الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Informatique académique

Thème :

Identification des services web à partir d'application orientée objet en utilisant les algorithmes de classification (étude comparative).

Encadré Par :

Mr Séridi Ali

Présenté par :

Nebili wafa,

Chelaghmia Rima.

Juin 2015

Remerciements

Nous voulons exprimer par ces quelques lignes de remerciements notre gratitude envers tous ceux en qui par leur présence, leur soutien, leur disponibilité et leurs conseils, nous avons eu le courage d'accomplir ce projet.

Nous commençons par remercier dans un premier temps, Dieu(Allah) qui nous a donné la volonté, la patience et la force pour réaliser notre travail.

Nos vifs remerciements à Monsieur Séridi Ali qui nous a fait l'honneur d'être notre encadrant. Nous le remercions profondément pour son encouragement continu et aussi d'être toujours là pour nous écouter, nous aider et nous guider pour retrouver le bon chemin par sa sagesse et ses précieux conseils. Ainsi que son soutien moral et sa preuve de compréhension, ce qui nous a donné la force et le courage d'accomplir ce projet.

Nos profonds remerciements pour les membres de jury qui ont accepté d'évaluer ce travail.

Nous exprimons également notre gratitude à tous les enseignants du département d'informatique, pour leur dévouement et leur assistance tout au long de nos études universitaires.

Merci finalement à tous ceux qui de près ou de loin ont eu à nous soutenir le long de nos parcours scolaire.

Chelaghmia Rima

Nebili Wafa

Dédicace

À

Mon père

L'épaule solide, l'œil attentif, compréhensif et la personne la plus digne de mon estime et de mon respect.

Aucune dédicace ne saurait exprimer mes sentiments, que Allah te présente et te procure santé et longue vie.

Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.

À

Ma mère

Affable, honorable, aimable : Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études.

Aucune dédicace ne saurait être assez éloquente pour exprimer ce que tu mérites pour tous les sacrifices que tu n'as cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte.

Je te dédie ce travail en témoignage de mon profond amour. Puisse Allah, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur.

À

Mon Frère : Abdesslam

Mon ange gardien et mon fidèle compagnon dans les moments les plus délicats de cette vie mystérieuse.

Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

À

Mes Chères Amies

Sana, Dounia, Wassila, Soumia, Imene, Sandra, Amina, Asma, Sana, Linda, Youssra, Khawla, Wafa, Samia

En témoignage de l'amitié qui nous uni et des souvenirs de tous les moments que nous avons passé ensemble, je vous porte toujours dans mon cœur, je vous souhaite tout le bonheur, la joie et le succès dans votre vie,

À

Toute ma famille

Petit ou grand, proche soit-elle ou lointaine.

A mes collègues et tous ceux qui m'ont aidée de près ou de loin, Je dédie ce travail avec hommage.

Chelaghmia Rima

Dédicace

C'est avec profonde gratitude et sincères mots, que je dédie ce travail

À

Mon père

Mon premier encadrant, depuis ma naissance, Rien au monde ne vaut les efforts fournis jours et nuits pour mon éducation et mon bien être.

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous.

À

Ma mère

Qui m'a comblée de son soutien et m'a vouée un amour inconditionnel. Tu es pour moi un exemple de courage et de sacrifice continu.

Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

À

Mes sœurs

Meriem, Abir

Vous m'avez toujours soutenue durant toutes mes études, je vous souhaite beaucoup de réussite et de bonheur.

À

Mon frère : Achraf

Mon cher petit frère présent dans tous mes moments par son soutien moral et ses belles surprises sucrées.
Je te souhaite un avenir plein de joie, de bonheur, de réussite et de sérénité.
Je t'exprime à travers ce travail mes sentiments de fraternité et d'amour.

À

Mes amies

Kawtar, Houda, Khawla, Samia, Rima, Asma

Je ne peux trouver les mots justes et sincères pour vous exprimer mon affection et mes pensées, vous êtes pour moi des sœurs et des amies sur qui je peux compter.
Je vous souhaite le courage et le succès dans votre vie.

À

Tous les membres de ma famille, petits et grands

Veuillez trouver dans ce modeste travail l'expression de mon affection

A Mes collègues et tous ceux qui m'ont aidée durant mon travail.

Nebili Wafa

Résumé

La variabilité importante des exigences des clients et la difficulté de répondre à leurs besoins avec des coûts de production acceptables ont poussé les entreprises à revoir en profondeur leurs processus de développement et d'adapter leur système d'information afin de devenir plus flexibles et plus agiles [11].

L'architecture orientée service (SOA) ne cesse d'attirer de nouveaux clients dans le monde de l'entreprise vu ses avantages d'agilité de réutilisabilité de composition et maitrise des coûts. Selon un rapport de 2007du Gartner Group, 50% des nouvelles applications à mission critique opérationnelles et des processus métier ont été conçus en 2007autour de SOA, et ce nombre sera plus grand que 80% en 2010. Le passage vers le SOA en utilisant le processus de réingénierie s'avère nettement moins chère qu'un redéveloppement d'un nouveau SI.

A travers notre projet nous avons essayé de proposer une solution à l'automatisation de la phase d'identification de service qui est une phase inévitable dans n'importe quelle approche de migration ver la SOA. Nous avons modélisé notre problème en un problème de regroupement et nous avons adapté et utilisé différents algorithmes de classification non supervisée (Soustractive clustring, IsoData algorithme …) pour atteindre notre but. Une phase d'évaluation et d'expérimentation a été menée afin de valider notre approche.

Les mots clé

Architecture Orienté Service, Service Web, Système Patrimonial, La Réingénierie, La Migration Vers SOA, Identification Des Services, Classification Non Supervisé, Algorithme Hiérarchique Ascendante, Algorithme Soustractive Clustering, Algorithme IsoData, Application Orienté Objet, Couplage, Cohésion.

Sommaire

Introduction générale.....1

Chapitre 1 : SOA et les services Web.

1 Introduction:.....3
2 Notion de base:.....3
2.1 Le service:.....3
2.1.1 Définition:.....3
2.1.2 Les caractéristiques des services:.....3
3 L'architecture orientée service:.....5
3.1 Définition:.....5
3.2 Les caractéristiques de l'architecture orientée service:.....5
3.3 Les acteurs de l'architecture orientée service:.....5
3.4 Les bénéfices de l'architecture orientée service:.....6
3.5 Les avantages de l'architecture orienté service:.....6
3.6 Les inconvénients de l'architecture orientée service:.....6
4 Les services web:.....7
4.1 Définition:.....7
4.2 Pourquoi les services web:.....7
4.3 Les principales technologies des services web:.....8
4.3.1 XML (Extensible Markup Language):.....8
4.3.1.1 Définition:.....8
4.3.1.2 Les caractéristiques d' XML :.....8
4.3.2 SOAP (Simple Object Access Protocol):.....9
4.3.2.1 Définition:.....9
4.3.2.2 Les caractéristiques de SOAP:.....9
4.3.3 WSDL (Web Services Description Language):.....10
4.3.3.1 Définition:.....10
4.3.3.2 Les caractéristiques de WSDL:.....11
4.3.4 UDDI (Universal Description, Discovery and Integration):.....11
4.3.4.1 Définition:.....11
4.4 L'architecture des services web.....12
4.5 Le fonctionnement global d'un échange des données grâce aux services web:.....12

5 Conclusion: 13

Chapitre 2 : Réingénierie et migration vers SOA.

1 Introduction: 14
2 L'ingénierie: 14
3 Les systèmes patrimoniaux: 15
4 La réingénierie: 15
4.1 Définition: 15
4.1.1 La réingénierie des systèmes d'informations: 15
4.1.2 La réingénierie des processus: 15
4.1.3 La réingénierie des logiciels: 15
4.1.3.1 Les étapes de réingénierie de logiciel: 16
4.2 Les avantages de la réingénierie: 17
4.3 Les risques reliés à la réingénierie: 17
5 La migration vers SOA: 17
5.1 L'approche orienté objet: 18
5.2 L'approche orienté service: 18
5.3 Les limites desarchitecture orientée objet par rapport à la SOA: 18
5.4 Définition de la migration: 19
5.5 Stratégies d'évolution vers une architecture orientée service: 19
5.5.1 Approches décisionnelles: 19
5.5.2 Approches partielles: 20
5.5.3 Approches techniques: 20
5.6 Les approches de migration vers SOA: 20
5.7 Les bénéfices de la migration: 21
5.8 Les difficultés de la migration: 21
6 Conclusion: 22

Chapitre 3 : Les algorithmes de classification non supervisé.

1 Introduction: 23
2 La classification: 23
2.1 Définition: 23
2.2 Les types de classification: 23
2.2.1 La classification supervisée: 23

2.2.2 La classification non supervisée: 24
2.3 La similarité et la dissimilarité: 24
2.4 Quelques algorithmes de classification non supervisée: 25
2.4.1 L'algorithme hiérarchique: 26
2.4.1.1 L'algorithme hiérarchique ascendant: 27
2.4.1.2 L'algorithme hiérarchique descendant: 28
2.4.2 L'algorithme Soustractive Clustering (SC): 28
2.4.3 L'algorithme de Kohonen: 29
2.4.4 L'algorithme IsoData: 30
2.4.5 L'algorithme EM (Expectation-Maximisation): 31
3 Conclusion: 32

Chapitre 4 : Conception.

1 Introduction: 34
2 L'objectif global: 34
3 Processus d'identification des services: 35
3.1 La mise en correspondance Objet-Service: 37
3.2 L'analyse du code source: 37
3.3 Le calcul des métriques: 38
3.3.1 Mesure de couplage: 39
3.3.2 Mesure de la cohésion: 41
3.4 La spécification de la fonction objectif: 45
3.4.1 L'Adaptation du calcul de couplage pour la fonction objectif: 46
3.4.2 Calcul de la fonction objectif: 48
3.4.3 Génération de la Matrice des similarités: 48
3.5 L'identification des services: 49
3.5.1 L'adaptation de l'algorithme hiérarchique: 49
3.5.1.1 Formation des services dans l'algorithme hiérarchique: 50
3.5.2 L'adaptation de l'algorithme Soustractive Clustering: 52
3.5.2.1 Discussion des choix d'adaptation: 54
3.5.3 L'adaptation de l'algorithme IsoData: 54
3.5.3.1 L'explication des choix d'adaptation: 56
4 Conclusion: 58

Chapitre 5 : Implémentation.

1 Introduction :.....59
2 Les outils de développement :.....59
2.1 Java :.....59
2.2 STAN :.....59
2.3 ASTParser :.....60
3 L'architecture globale de notre application:.....61
4 L'interface de l'application:.....63
4.1 Les options d'accueil :.....63
4.1.1 Charger un projet java :.....64
4.1.2 Récupérer une matrice de similarité à partir d'un fichier texte :.....64
4.1.3 Remplir une matrice de similarité :.....65
4.1.4 Le guide d'utilisation:.....65
4.1.5 Le calcul du temps d'exécution:.....66
4.1.6 Sauvegarder une matrice:.....66
4.1.7 Restaurer les fenêtres précédentes:.....66
4.1.8 Quitter l'application:.....66
4.2 Les algorithmes d'identification des services:.....66
5 Expérimentation:.....69
5.1 Présentation des applications de test:.....70
5.2 Le choix des paramètres :.....72
5.2.1 L'algorithme hiérarchique:.....72
5.2.1.1 Le choix du pourcentage de coupure pour l'algorithme hiérarchique :.....72
5.2.2 L'algorithme Soustractive Clustering:.....73
5.2.2.1 Le choix du centre potentiel et le rayon de la méthode Soustractive Clustering:.....73
5.2.3 L'algorithme IsoData:.....74
5.2.3.1 Le choix du nombre des clusters initiaux (K) et le seuil:.....74
5.2.3.2 Le choix de nombre des itérations:.....75
5.3 La comparaison entre les trois algorithmes:.....75
6 Conclusion:.....89
Conclusion générale et perspectives.....90
Références bibliographiques.....91

Liste des figures

Figure 1.1: Les caractéristiques de service.....4
Figure 1.2: Interaction entre les applications hétérogènes à travers XML 8
Figure 1.3: Structure générale d'un message SOAP 9
Figure 1.4: Document de WSDL.....11
Figure 1.5: Fonctionnement globale d'un échange de données grâce aux services Web.....12
Figure 2.1: Les étapes de réingénierie logicielle.....17
Figure 3.1 : Classification non supervisée selon différents critères de similarité.....24
Figure 3.2 : Partitions emboîtées d'un ensemble X à 6 éléments.....27
Figure 3.3 : L'algorithme hiérarchique ascendant.....27
Figure 3.4 : L'algorithme hiérarchique descendant.....28
Figure 3.5 : L'algorithme Soustractive Clustering.....29
Figure 3.6 : La distribution des données au tour d'un centre d'amas dans l'algorithme Soustractive Clustering.....29
Figure 3.7 : Le voisinage de rayon 2 pour une grille.....30
Figure 3.8 : Le voisinage de rayon 2 pour une ficelle.....30
Figure 3.9 : L'algorithme de Kohenon.....30
Figure 3.10 : L'algorithme IsoData.....31
Figure 3.11 : L'algorithme de K-means.....31
Figure 3.12 : L'algorithme EM.....32
Figure 4.1 : Processus général d'identification des services.....36
Figure 4.2 : La mise en correspondance entre l'orienté objet et l'orienté service.....37
Figure 4.3 : La structure des applications orientées objet.....38
Figure 4.4: Exemple de couplage entre quatre classes A, B, C, D..... 40
Figure 4.5 : Exemple qui explique la relation de cohésion d'une classe C.....42
Figure 4.6 : Cycle de calcul de couplage et cohésion d'une application OO.....44
Figure 4.7 : L'algorithme de transformation de couplage.....47
Figure 4.8 : Agrégation selon le lien minimale.....49
Figure 4.9 : Agrégation selon le lien moyenne.....49
Figure 4.10 : L'adaptation de l'algorithme hiérarchique.....49

Figure 4.11 : La sélection des services dans l'algorithme hiérarchique……..50

Figure 4.12 : Diagramme du processus d'identification de service de l'algorithme hiérarchique……51

Figure 4.13 : L'adaptation de l'algorithme Soustractive Clustering……52

Figure 4.14 : Diagramme du déroulement de l'algorithme Soustractive Clustering adapté....53

Figure 4.15: L'adaptation de l'algorithme IsoData……..55

Figure 4.16 : Diagramme du déroulement de l'algorithme IsoData adapté……56

Figure 4.17 : Algorithme de sélection des nouveaux centroides……57

Figure 5.1 : L'interface de STAN……60

Figure 5.2 : Les composantes d'une classe java……..60

Figure 5.3 : Architecture global de notre application……..62

Figure 5.4 : L'interface de notre application……63

Figure 5.5 : Fenêtre des résultats d'analyse d'un projet java……...64

Figure 5.6 : Récupération d'une matrice de similarité à partir d'un fichier texte……...64

Figure 5.7 : Matrice carrée de taille 4*4……..65

Figure 5.8 : Le guide d'utilisation……65

Figure 5.9 : Le temps d'exécution……....66

Figure 5.10 : Boite de confirmation de sauvegarde d'une matrice……..66

Figure 5.11 : Boite du choix du pourcentage de regroupement……....67

Figure 5.12 : Le choix du centre potentiel et du rayon pour Soustractive Clustering……....67

Figure 5.13 : Choix du nombre des centres initiaux et le seuil pour IsoData……..68

Figure 5.14: Le nombre des services……....68

Figure 5.15 : Le choix du type d'affichage pour les résultats obtenus……68

Figure 5.16 : Le résultat de L'algorithme IsoData sous forme d'un tableau……69

Figure 5.17 : Le résultat de L'algorithme IsoData sous forme graphique……69

Liste des tableaux

Tableaux 2.1: La déférence entre une architecture orientée objet et une SOA 18

Tableau 3.1: Taxonomie des méthodes de clustering 25

Tableau 4.1: Tableau des résultats de couplage de l'exemple1. 41

Tableau 4.2: Classification des caractéristiques des services. 45

Tableau 4.3: L'évaluation des caractéristiques d'un service..... 46

Tableau 4.4: Tableau de pourcentage de couplage. 48

Tableau 5.1: Quelques éléments en java et leur correspondance en ASTParser..... 61

Tableau 5.2: Les résultats attendus des deux exemples d'expérimentation. 70

Tableau 5.3: Le résultat de coupure de l'algorithme hiérarchiquepour des pourcentages différents..... 72

Tableau 5.4: Le résultat de l'algorithme SC pour des différents rayons et différents centres potentiels. 73

Tableaux 5.5: Le choix du seuil et de k cluster pour IsoData. 74

Tableaux 5.6: Tableau comparatif des nombres des itérations pour la méthode IsoData. 75

Tableau 5.7: Comparaison des résultats de l'algorithme hiérarchique pour l'exemple 1. 76

Tableau 5.8: Comparaison des résultats de l'algorithme hiérarchique pour l'exemple 2. 77

Tableau 5.9: Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre aléatoire)..... 78

Tableau 5.10: Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre est la classe dont leur similarité est le min). 79

Tableau 5.11: Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre est la classe dont leur nombre des fonctionnalités est le max). 80

Tableau 5.12: Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre est la classe dont leur nombre des fonctionnalités est le max). 81

Tableau 5.13: Comparaison des résultats de l'algorithme SC pour l'exemple 1 (la classe dont la moyenne des similarités est le min). 82

Tableau 5.14: Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre soit le min. 83

Tableau 5.15: Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre la classe dont leur similarité est le min. 84

Tableau 5.16: Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre la classe dont leur nombre des fonctionnalités est le max..... 84

Tableau 5.17: Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre c'est la classe dont la moyenne des similarités est le min..... 85

Tableau 5.18: Comparaison des résultats de l'algorithme IsoData pour l'exemple 1..... 86

Tableau 5.19: Comparaison des résultats de l'algorithme IsoData pour l'exemple 2..... 87

Tableau 5.20: Tableau comparatif de temps d'exécution écoulé dans quelque application. .. 88

Tableau 5.21: Tableaux comparatif pour les trois algorithmes. 89

Liste des abréviations:

SOA : Service Oriented Architecture.

XML : eXtensible Markup Language.

HTML : Hypertext Markup Language.

SOAP : Simple Object Access Protocol.

SMTP : Simple Mail Transfer Protocol.

POP : Post Office Protocol.

RPC : Remote Procedure Call.

WSDL : Web Services Description Language.

IDL : Interface Definition Langage.

UDDI : Universal Description Discovery and Integration.

OO : Orienté Objet.

OS : Orienté Service.

SI : Système d'Information.

SP : Système Patrimonial.

CAH : Classification Hiérarchique Ascendante.

CHD : Classification Hiérarchique Descendante.

EM : .Expectation Maximisation.

SC : Soustractive Clustering.

UA : Usage Attribut.

IM : Invocation Méthode.

JSP : Java Server Pages.

JDT : Java Development Tools.

PDE : Plug-in Développent Environnent.

IDE : Integrated Development Environment.

AST : **Abstract** Syntax Tree.

JAR : Java Archive.

Introduction générale

Introduction générale

De nos jours les systèmes informatiques sont développés de manière exponentielle, sous forme d'architectures logicielles qui relèvent une complexité considérable au niveau de la structuration et engendrent des coûts élevés pour l'entreprise [w13].

Les systèmes patrimoniaux contiennent souvent une grande quantité de données pertinente qui répond à une architecture bien définie, en effet les architectures traditionnelles atteignent leurs limites en termes de capacité, alors que les besoins traditionnels des organisations informatiques demeurent.

Afin de pallier à ce problème, il existe une nouvelle architecture, nommé architecture orientée service (SOA) qui s'avère être une solution adéquate.

La notion de SOA renvoie à une nouvelle manière d'intégrer et de manipuler les différentes briques et composants applicatifs d'un système informatique (comptabilité, gestion de la relation client, production, etc.) et de gérer les liens qu'ils entretiennent. Cette approche repose sur la réorganisation des applications en un ensemble fonctionnel appelé service. Un service est un composant logiciel fonctionnant de manière autonome et offrant des fonctionnalités métiers à d'autres applications ou d'autres services au travers d'une interface standard bien défini. Idéalement chaque service doit être indépendant des autres afin de garantir sa réutilisabilité et son interopérabilité.

L'ensemble des services dialoguent entre eux au travers de bus ou par Internet, on parle alors de Web Service (WSOA). Les échanges des données peuvent se faire de manière synchrone ou asynchrone [w38].

Notre étude s'inscrit dans le cadre de la réingénierie des systèmes patrimoniaux vers de nouvelles architectures et plus particulièrement l'architecture orientée service.

Allons de ce principe, nous avons essayé de transformer les applications orientées objet vers des applications orientées service afin de bénéficier des avantages de cette nouvelle architecture, à savoir, l'agilité, l'interopérabilité, la réutilisabilité, minimisation de cout et facilité de maintenance.

Nous avons choisi les applications orienté objet puisque d'une part, elles sont devenues la partie dominante des applications patrimoniales disponible sur le marché, et d'autre part elles sont plus disponible à être migré vers une SOA.

La tendance à migrer vers la SOA a suscité l'intérêt d'un nombre important de chercheurs ces dernières années, ainsi plusieurs approches de migration vers cette architecture sont apparues. Bien que les stratégies et approches de migration vers la SOA soient multiples et différentes, toutes s'accordent sur l'importance d'une phase essentielle qui est la phase d'identification de service.

Nous nous sommes fixé l'objectif d'automatiser cette phase cruciale qui est l'identification de service à partir d'applications orientée objet.

Afin d'atteindre ce but nous avons exploré les algorithmes de classification non supervisés, à savoir (regroupement hiérarchique, Soustractive Clustering, IsoData) pour regrouper l'ensemble des classes homogènes pour former un service tout en respectant les caractéristiques de la SOA (l'autonomie, la composabilité, le couplage lâche…).

Notre mémoire est décomposé en cinq chapitres organisés de la manière suivante :

Le premier chapitre : est consacré à un état de l'art de l'architecture orienté service où nous parcourons brièvement les concepts de la SOA et la nouvelle technologie de l'implémentation de la SOA qui est les services web.

Le deuxième chapitre : porte sur le processus de réingénierie et la technique de la migration vers le SOA avec ces différentes approches.

Le troisième chapitre: présente quelques algorithmes de classification non supervisées afin de choisir quelque uns pour identifier les services.

Le quatrième chapitre : c'est la conception de notre projet, dans cet étape nous allons expliquer le processus que nous avons suivi pour concevoir notre objectif et comment nous avons adapté les algorithmes choisis afin d'identifier les services.

Le cinquième chapitre : comportera les différents détails de l'implémentation où nous définirons les outils utilisés et présenterons les différentes interfaces ainsi que les résultats obtenu.

Enfin, nous clôturons ce mémoire par une conclusion générale et quelques perspectives.

Chapitre 1 :

SOA et les services web

1 Introduction:

Nous vivons actuellement dans une société de consommation orientée vers la technologie. Cette dernière est présente dans la vie de tous les jours autant pour les particuliers que pour les entreprises. Un des concepts clé de cette mutation technologique est l'informatique, qui est devenu un des piliers d'une bonne gestion organisationnelle [1].

Un des pas géants qui a vulgarisé l'utilisation de l'informatique est l'apparition de l'internet.

Cette vulgarisation du monde informatique a entraîné le développement de nouvelles approches architecturales tel que l'approche orientée service qui est une approche architecturale de dernière génération [w1].

L'architecture orientée service ou SOA (Service Oriented Architecture) est aujourd'hui envisagée par de nombreuses entreprises dans le cadre de l'évolution de leur système d'information.

A travers ce chapitre nous essayerons de présenter les notions relatives à la SOA puis à une des technologies d'implémentation qui est les services web.

2 Notion de base:

2.1 Le service:

2.1.1 Définition:

On distingue plusieurs définitions pour le service :

Définition 1: Un Service est un composant logiciel distribué, exposant les fonctionnalités à forte valeur ajoutée d'un domaine métier [w2].

Définition 2: Un service est une ressource abstraite qui effectue une tâche cohérente et fonctionnelle [2].

Définition 3: Un service est considéré comme un processus qui a une interface ouverte, et une granularité grossière. Il peut être facilement décomposé et composé pour mettre en œuvre divers flux de travail [3].

2.1.2 Les caractéristiques des services:

Chaque service est caractérisé par les caractéristiques suivantes:

- **Contrat standardisé:** L'ensemble des services d'un même système technique sont exposés au travers des contrats respectant les mêmes règles de standardisation (l'interface) [w2].

- **Couplage lâche:** Le couplage faible (loosely-coupled) est une propriété qui montre le degré de liaison des services. Les services sont connectés aux autres services ou aux clients à travers des documents standards, ces documents sont réalisés en XML de la même manière que les web services et ils assurent le découplage ou autrement dit la réduction des dépendances [w2].
- **Abstraction:** Le contrat d'un service ne doit contenir que les informations essentielles à son invocation. Seules ces informations doivent être publiées [w2].
- **Réutilisabilité:** Les services doivent encapsuler une logique de traitement suffisamment générique pour être utilisés dans des contextes d'utilisation différents [w3].
- **Autonomie:** Un service doit être totalement autonome. On doit pouvoir le remplacer ou le déplacer sans que cela affecte d'autres services. Un service implémente ses propres composants et ses propres méthodes d'accès aux données, il ne doit dépendre d'aucun élément externe [w4].
- **Sans état (stateless):** Son exécution ne dépend pas de l'invocation préalable d'autres services, l'ensemble du contexte qui est fourni lors de l'invocation par le consommateur suffit au service pour exécuter ses actions [w5].
- **Découvrabilité:** Un service est complété par un ensemble de métas donnés de communication au travers desquelles il peut être découvert et interprété de façon effective [w2].
- **Composabilité:** Un service doit être conçu de façon à participer à des compositions de services [w2].
- **Synchrone ou asynchrone:** Attente de réponse après l'utilisation d'un service ou non [w6].

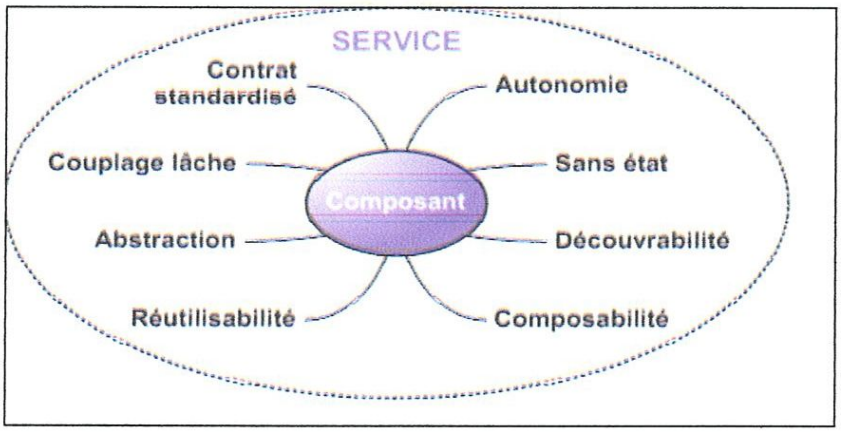


Figure 1.1 : *Les caractéristiques de service [w2].*

3 L'architecture orientée service:

3.1 Définition:

La SOA est un ensemble de principes architecturaux qui permettent de développer des systèmes modulaires basés sur des « services » ou des unités de fonctionnalités informatiques. Ces services, qu'ils soient métier ou techniques, sont offerts par une entité, le prestataire de services, et consommés par une autre [4].

3.2 Les caractéristiques de l'architecture orientée service:

SOA possède les caractéristiques suivantes:

- **Réutilisation et composition:** Ceci est particulièrement puissant pour créer des nouveaux processus d'affaires rapide et fiable.
- **Recomposition:** La capacité de modifier les processus d'affaires existants ou d'autres applications basées sur l'agrégation des services.
- **La capacité à progressivement changer le système:** Commutation des prestataires de services, l'extension des services, en modifiant les fournisseurs de services et les consommateurs. Tous ces éléments peuvent être faits en toute sécurité, grâce au couplement bien contrôlé.
- **La capacité à construire progressivement le système:** Cela est particulièrement vrai pour toute intégration basée sur SOA [w7].

3.3 Les acteurs de l'architecture orientée service:

L'architecture orienté service se représente en faisant intervenir trois acteurs: le consommateur, le service, et le répertoire de services.

Le consommateur correspond à l'application cliente (ou à un autre service), qui fait appel au service pour une tâche précise. Ce consommateur trouvera les informations à propos du client au sein du répertoire de services, où sont enregistrés et triés un grand nombre de ceux-ci. Un répertoire peut être privé, c'est-à-dire interne à l'entreprise, ou public. Le service fournit une fonctionnalité bien définie et répond à trois fonctionnalités caractéristiques : il est indépendant, il peut être découvert et appelé de manière dynamique et il fonctionne seul.

Le répertoire de services a un rôle primordial dans la SOA. C'est lui qui reçoit la requête du consommateur, lui qui découvrira le service idoine, et lui qui agira en tant que proxy (intermédiaire) entre consommateur et service. En s'assurant que les fournisseurs

de services informent régulièrement les répertoires de leurs nouveautés, le consommateur peut constamment profiter de celles-ci sans pour autant devoir mettre à jour ses méthodes [w8].

3.4 Les bénéfices de l'architecture orientée service:

SOA avec sa nature faiblement couplée permet aux entreprises de :

- Brancher de nouveaux services.
- Améliorer les services existants de façon granulaire pour répondre aux nouveaux besoins.
- Offrir la possibilité de rendre les services consommables à travers différents canaux.
- Exposer les applications existantes de l'entreprise comme des services, tout en préservant les investissements d'infrastructure informatique existante [w7].

3.5 Les avantage de l'architecture orienté service:

- Une modularité permettant de remplacer facilement un composant ou service par un autre.
- Une réutilisabilité possible des composants par opposition d'un système tout en un fait sur mesure pour une organisation.
- De meilleures possibilités d'évolution, ici il suffit de faire évoluer un service ou d'ajouter un nouveau service.
- Une plus grande tolérance aux pannes.
- Une maintenance facilitée.
- Obligation d'avoir une modélisation poussée.
- Possibilité de découpler les accès aux traitements.
- Localisations et interfaçage transparents.
- Possibilité de mise en place facilitée à partir d'une application objet existante.
- Réduction des coûts en phase de la maintenance et d'évolution.
- Facilité d'amélioration des performances pour des applications importantes (répartition de traitements facilités) [w7].

3.6 Les inconvénients de l'architecture orientée service:

- Nécessité d'appréhender de nouvelles technologies.
- Performances réduites pour les traitements simples (couche supplémentaire) [w11].
- Tendance à la multiplication des couches et des messages.
- Problème de coordination ou orchestration entre les divers services.
- Problème de sécurité (plus grande surface exposée).

- Problème de réelle interopérabilité.
- On constate une certaine lourdeur et de la complexité [w12].

4 Les services web:

Le service web est une des technologies d'implémentation de l'architecture orientée service la plus mur, nous pouvons le définir comme suit:

4.1 Définition:

On distinct deux définition:

Définition 1: Un service web est une application accessible à partir du web, il utilise les protocoles internet pour communiquer et utilise un langage standard pour décrire son interface [w9].

Définition 2: Les services web sont des compléments aux programmes et applications existantes, développées à l'aide de langages tel que Visual Basic, C, C++, C# (C sharp), Java ou autre, et servent de pont pour que ces programmes communiquent entre eux.

Les services web définissent non seulement les données transmises entre deux applications, mais aussi comment traiter ces données et les relier à l'interne et à l'externe d'une application logicielle sous-jacente [w10].

4.2 Pourquoi les services web:

- Les services web permettent d'interconnecter:
  - ✓ Différentes entreprises.
  - ✓ Différents matériels.
  - ✓ Différentes applications.
  - ✓ Différents clients.
- Réutilisation dans un environnement ouvert (run time).
  - ✓ Distribuer et intégrer des logiques métiers vers le web sémantique (pas uniquement le web purement interactif).
- Les services web sont faiblement couplés [w9].

4.3 Les principales technologies des services web:

4.3.1 XML (Extensible Markup Language):

4.3.1.1 Définition:

XML (Extensible Markup Language, ou Langage Extensible de Balisage) est le langage destiné à succéder à HTML. Comme HTML (Hypertext Markup Language) c'est un langage de balisage (markup): il représente de l'information encadrée par des balises. XML est un métalangage, ce qui veut dire que contrairement à HTML qui possède un ensemble de balises de présentation prédéfinies, il va permettre d'inventer de nouvelles balises d'isolement d'informations ou d'agrégats élémentaires que peut contenir une page web [5].

XML permet donc de transformation internet d'un univers d'information et de présentation de sites web statiques à un univers web programmable et dynamique, centré sur les données [w10].

4.3.1.2 Les caractéristiques d' XML:

- Il est indépendant des plates-formes informatiques.
- Il est lisible par l'humain mais est destiné à être lu par la machine.
- Il est flexible en ce sens que vous pouvez définir d'autres langages à partir d'XML.
- XML permet aux données d'êtres universellement navigables.
- XML permet l'interopérabilité entre déférents langages (figure 1.2) [w10].

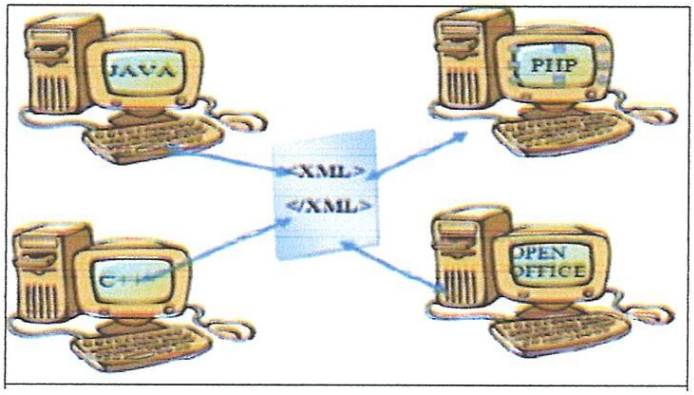


Figure 1.2 : *Interaction entre les applications hétérogènes à travers XML [w10].*

4.3.2 SOAP (Simple Object Access Protocol):

4.3.2.1 Définition:

C'est un protocole de dialogue par appels de procédures à distance entre objets logiciels. Sa syntaxe d'utilisation est fondée sur XML et ses commandes sont envoyées sur Internet par l'intermédiaire du protocole HTTP mais aussi SMTP et POP sous forme de texte structuré.

Il permet aux systèmes objets distribués de solliciter et d'obtenir des services rendus par d'autres objets, il est moins lourd à mettre en œuvre que d'autres protocoles et c'est pour cela qu'il est de plus en plus adopté [5]. SOAP est le protocole de communication standard des services web [6].

Le standard SOAP définit trois éléments composant un message : l'enveloppe, l'entête du message et le corps du message.

L'enveloppe définit le cadre pour décrire ce qui est dans le message et comment le traiter.

Les règles d'encodages sont placées dans ***l'en-tête*** et servent à exprimer et définir le mécanisme de représentation des données.

Le ***corps*** du message permet de transmettre les requêtes et les réponses entre les systèmes [w10].

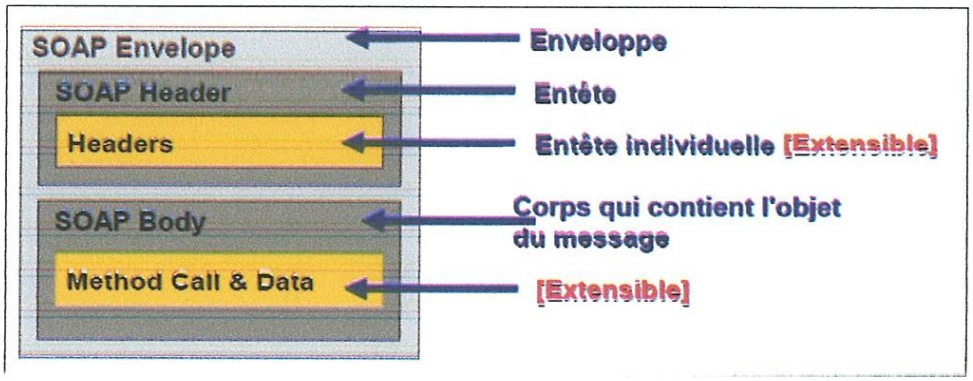


Figure 1.3 : *Structure générale d'un message SOAP [w9].*

4.3.2.2 Les caractéristiques de SOAP:

- Par rapport à tous les autres protocoles de RPC, SOAP est inter opérable.
- SOAP est indépendant des plates-formes et langages de programmation [6].

4.3.3 WSDL (Web Services Description Language):

4.3.3.1 Définition:

WSDL est un langage permettant de décrire les services web et les données attendues par ces services web. L'utilité de WSDL est donc de décrire et publier le format et les protocoles d'un service web de manière homogène par l'utilisation de XML.

Cela permettra au requérant et à l'émetteur d'un service de comprendre les données qui seront échangées.

Ce langage possède une structure, appelé structure de document WSDL qui est divisée en deux parties: le groupe du haut constitué des définitions abstraites et le groupe du bas composé des descriptions concrètes [5].

- **Définition abstraite:**
  - ✓ Les types (XML Schéma) des paramètres et des résultats des messages.
  - ✓ **Les messages:** Un ensemble des données.
  - ✓ **Opération:** Une action proposée par un service web, décrite par ses messages (proche d'une méthode au sens Java).
  - ✓ **Type de port:** Un ensemble d'opérations (proche d'une interface au sens Java)
  - ✓ **Donnée:** Une information typée selon un système de type comme celui des schémas du W3 [5].
- **Définition concrète:**
  - ✓ **Binding:** Un protocole et un format de données associé à un type de port (port type).
  - ✓ **Port:** Une adresse réseau et un binding.
  - ✓ **Service:** Une collection de port (port ou end points) [5].

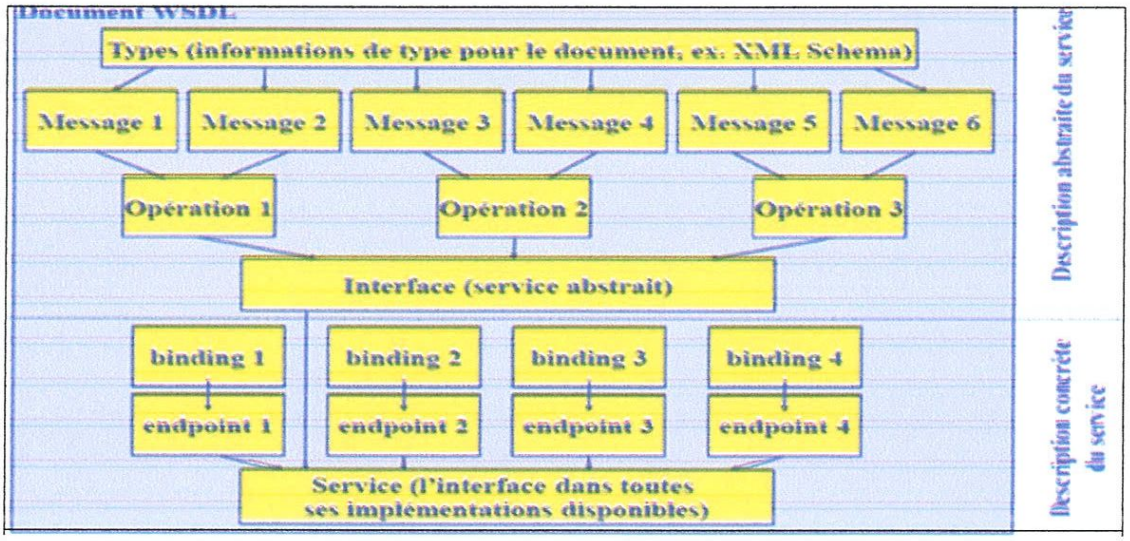


Figure 1.4 : *Document de WSDL [7].*

4.3.3.2 Les caractéristiques de WSDL:

- Ce langage est indépendant par rapport au langage de programmation et à la plateforme utilisée.
- WSDL est la représentation XML du langage IDL (Interface Définition langage) [w1].

4.3.4 UDDI (Universal Description, Discovery and Integration):

4.3.4.1 Définition:

UDDI définit les mécanismes permettant de répertorier des services web. Ce standard régit donc à l'information relative à la publication, la découverte et l'utilisation d'un service web. En d'autres mots, UDDI détermine comment nous organiser l'information concernant une entreprise et les Services web qu'elle offre à la communauté afin de permettre à cette communauté d'y avoir accès. En fait, UDDI définit un registre des services web sous un format XML. Ce registre peut être public, privé ou partagé [w10].

L'annuaire UDDI possède une structure de données bien spécifique qui contient l'ensemble des informations divisées en trois parties (les pages blanches, les pages jaunes et les pages vertes).

- **Pages blanches:** Données sur le fournisseur du service (nom, adresse, …).
- **Pages jaunes:** Classification du type de service, basée sur des standards.
- **Pages vertes:** Information technique sur l'utilisation du service (Pointeurs sur les descriptions WSDL, qui ne font pas partie de l'annuaire) [7].

4.4 L'architecture des services web:

Il s'agit d'identifier chaque acteur de ses services web et de comprendre comment ils interagissent les uns avec les autres. Les trois éléments les plus importants des services web sont les fournisseurs de service, les annuaires de services et les consommateurs de service. Le fournisseur (ou serveur) crée le service web et publie toutes ces caractéristiques dans l'annuaire de service.

L'annuaire rend disponible les interfaces d'accès aux services et donnant le contrat et l'architecture employée pour permettre les interactions.

Le consommateur (ou client) quant à lui, accède à l'annuaire pour rechercher les services web dont il a besoin et avec lui les normalisations à obtenir.

Il peut ainsi envoyer ses requêtes au service désiré et obtenir les réponses qu'il pourra analyser [5].

4.5 Le fonctionnement global d'un échange des données grâce aux services web:

Elle s'agit d'identifier chaque acteur de l'architecture orienté service et de comprendre comment ils interagissent les uns avec les autres.

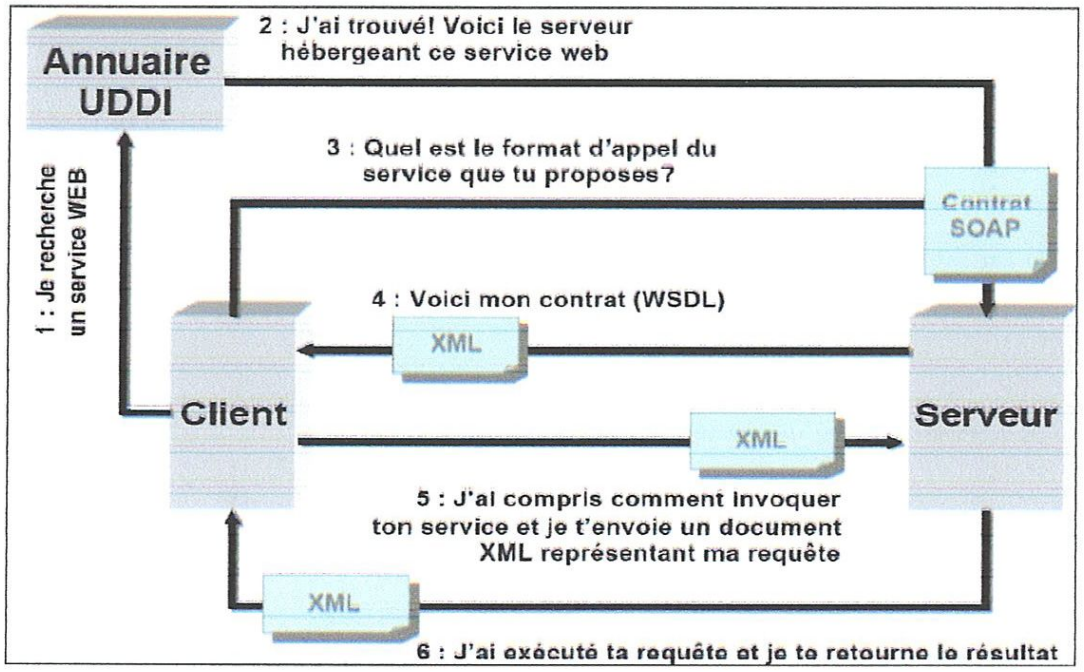


Figure 1.5 : *Fonctionnement globale d'un échange des données grâce aux services web [5].*

D'après la figure 1.5 le fonctionnement se fait comme ce suit:

- Le client envoie une requête à l'annuaire de service pour trouver le service web dont il a besoin.
- L'annuaire cherche pour le client, trouve le service web approprié et renvoie une réponse au client en lui indiquant quel serveur détient ce qu'il recherche.
- Le client envoie une deuxième requête au serveur pour obtenir le contrat de normalisation de ses données.
- Le serveur envoie sa réponse sous la forme établie par WSDL en langage XML.
- Le client peut maintenant rédiger sa requête pour traiter les données dont il a besoin.
- Le serveur fait les calculs nécessaires suite à la requête du client, et renvoie sa réponse sous la même forme normalisée [5].

5 Conclusion:

Dans ce chapitre nous avons illustré les concepts de base de la SOA qui est un style architectural moderne qui suit des principes de modélisation modernes permettant la réutilisation des services ainsi que l'exécution des processus d'entreprise.

Le service web qui est une nouvelle technologie d'implémentation de la SOA offre des applications accessible à partir du web ce qui donne plus de flexibilité et de souplesse aux utilisateurs.

Dans Le chapitre suivant nous présenterons la notion de réingénierie et les techniques de migration des systèmes patrimoniaux vers la SOA.

Chapitre 2 :

Réingénierie et migration vers SOA

1 Introduction:

Le secteur d'activité des logiciels a connu de nombreuses architectures conçues pour permettre un traitement des problèmes à un domaine bien précis.

Les architectures traditionnelles ont atteint leurs limites en termes de capacité, alors que les besoins traditionnels des organisations informatiques demeurent [w13].

Dans ce cadre l'évolution et la modernisation du logiciel patrimonial est nécessaire, puisque un bon logiciel doit être compatible avec les technologies disponibles ainsi que les compétences des équipes de maintenance.

La réingénierie logicielle est l'approche qui permet l'adaptation du logiciel traditionnel, puisque elle est basée sur la remise en cause fondamentale et la reconception radicale des processus organisationnels, afin de réaliser des améliorations spectaculaires des performances courantes [w14].

L'architecture orientée service est une architecture logicielle s'appuyant sur un ensemble des services simples. Son objectif est de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services [w13].

Dans ce chapitre nous allons présenter le processus de réingénierie, qui permet de faire évoluer un ancien système vers une nouvelle architecture plus récente ainsi que les notions relatives. L'accent sera mis sur la modernisation des systèmes orientés objet vers une architecture SOA.

2 L'ingénierie:

L'ingénierie (forward engineering) est le processus traditionnel de passage d'un haut niveau d'abstraction et de conception détaillée vers le développement physique du système, Le processus implique généralement les étapes suivantes:

- L'analyse.
- La conception.
- Le codage.
- Test et maintenance [w20].

3 Les systèmes patrimoniaux:

L'expression « systèmes patrimoniaux » s'est largement répandue pour désigner les systèmes informatiques issus d'une génération précédente et nécessitante une mise à niveau [w20].

4 La réingénierie:

4.1 Définition:

La réingénierie consiste à reconcevoir, repenser ce qui a été conçu dans une démarche d'ingénierie. Il est difficile, voire impossible de faire de la réingénierie si ce qui a été conçu précédemment n'a suivi aucune règle, aucune norme.

Dans le secteur de l'informatique il existe plusieurs types de réingénierie:

- Réingénierie des systèmes d'informations.
- Réingénierie des processus.
- Réingénierie logicielle [w15].

4.1.1 La réingénierie des systèmes d'information:

Il s'agit d'une remise à plat de tout ou partie d'un système d'information, afin d'atteindre de meilleurs niveaux de performances globales. Cela peut par exemple consister à transformer une architecture informatique initialement tournée vers la production en un dispositif orienté vers le client [w16].

4.1.2 La réingénierie des processus:

La réingénierie des processus traite la transformation des processus de l'entreprise, et ce, essentiellement au sein des grandes entreprises. Cette transformation est toujours le fruit d'un processus ou d'un plan de gestion du changement maîtrisé, dans le cadre duquel une organisation audite et compare les processus existants puis tente de les optimiser [w17].

4.1.3 La réingénierie des logiciels:

La mission de la réingénierie des logiciels est l'amélioration et la transformation d'un logiciel existant de telle façon que la compréhension, le contrôle et l'utilisation de ce dernier puissent de nouveau être garantis. La demande en réingénierie de logiciels a connu une croissance spectaculaire liée à l'obsolescence des systèmes de logiciels patrimoniaux, l'obsolescence portant non seulement sur leur architecture, mais également sur les plateformes les supportant ainsi que sur leur adaptabilité et leur stabilité face aux développements techniques requis par l'évolution des besoins.

La réingénierie des logiciels s'avère nécessaire afin de rétablir et de réutiliser les ressources des logiciels existants, de contrôler davantage les coûts élevés de maintenance des logiciels et de créer une base solide à même de supporter les évolutions à venir [w18].

4.1.3.1 Les étapes de réingénierie de logiciel:

Les grandes étapes de la réingénierie de logiciel sont :

- **La rétro-ingénierie (reverse engineering):**

C'est le processus d'analyse d'un système en vue d'identifier ses composantes, leurs corrélations et de créer des représentations du système sous une autre forme ou d'un niveau plus élevé d'abstraction.

Dans cette étape, il existe un sous-ensemble qui permet de comprendre la fonction d'un programme grâce à l'interprétation de code et de documentation de l'application, ce sous-ensemble s'appelle rétro-conception (Design Recovery) [8].

- **La recomposition (Forward Engineering):**

C'est le processus qui permet la conception physique d'un système informatique à partir de sa définition logique c'est à dire le haut niveau d'abstraction [8].

- **La redocumentation (Redocumentation):**

Est la création ou la révision d'une représentation sémantiquement équivalente dans le même niveau relatif d'abstraction. Les formes de représentation résultantes sont habituellement considérées des vues alternatives (par exemple, flux de données, structures de données, et flux d'informations) destinées à une compréhension humaine [w20].

- **La restructuration (Restructuring):**

Est la transformation d'un logiciel en un autre logiciel ayant le même niveau d'abstraction tout en préservant le comportement externe du logiciel (fonctionnalité et sémantique) [w20].

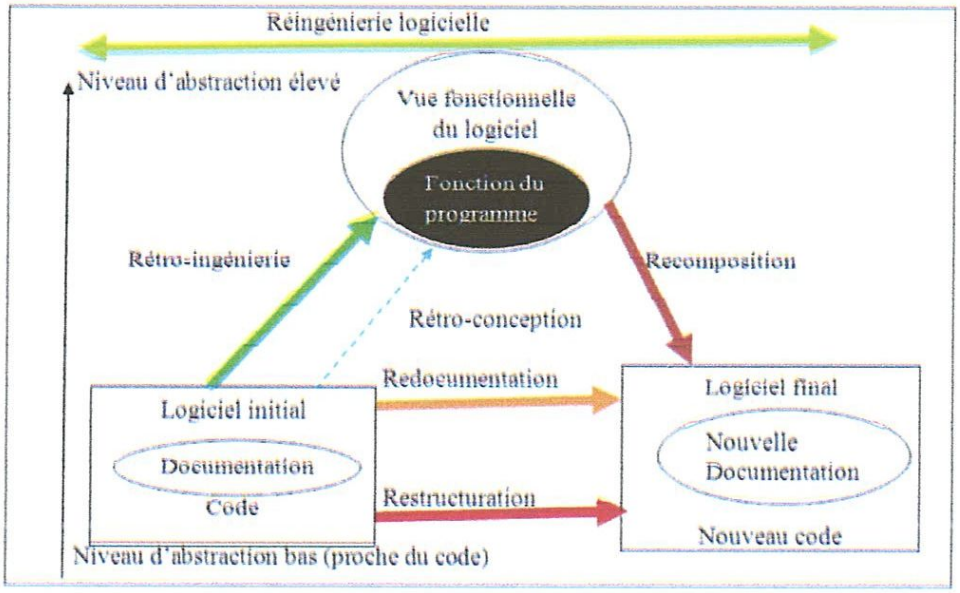


***Figure 2.1** : Les étapes de la réingénierie des logiciels [8].*

4.2 Les avantages de la réingénierie:

Les avantages de la réingénierie sont :

- Réduire les risques: redévelopper un système critique introduit beaucoup de nouveaux risques.
- Réduire les coûts: redévelopper un nouveau système coûte plus cher que la modification d'un système existant [w19].
- Réduire la perte de données et la fuite d'informations.
- Améliorer l'efficacité opérationnelle et le service à la clientèle [w19].

4.3 Les risques reliés à la réingénierie:

On désigne plusieurs facteurs de risque lié à la réingénierie :

- La capacité à éliminer ou à réduire les propriétés indésirables du système existant.
- L'introduction des nouvelles technologies n'est pas supportée par les programmeurs.
- L'impact de l'introduction des nouvelles parties au système existant.
- L'impact des changements sur la performance et la robustesse du système existant [w20].

5 La migration vers SOA:

La SOA permettant plus de souplesse dans la gestion du système d'information. En effet, les systèmes d'informations des entreprises sont composés d'application et des données

constituants leurs héritages [w25]. Nous nous somme intéressé dans notre étude au côté applicatif qui concerne la modernisation (ou bien la migration) des applications orientées objet vers des applications orientées service.

Avant d'entamer la présentation du processus de migration, nous avons choisi, tout d'abord, d'exposer quelques notions importantes sur l'approche orientée objet et ses limites par rapport à la SOA, puis nous présenterons le terme de migration, ses différentes approches, ses avantages et ses limites.

5.1 L'approche orienté objet:

L'approche orientée objet considère le logiciel comme une collection d'objets dissociés, identifiés et possédant des caractéristiques. Une caractéristique est soit un attribut, soit une entité comportementale de l'objet (i.e. une fonction). La fonctionnalité du logiciel émerge alors de l'interaction par l'envoie des messages entre les différents objets qui le constituent. L'une des particularités de cette approche est qu'elle rapproche les données et leurs traitements associés au sein d'un objet unique [w26].

5.2 L'approche orienté service:

La notion de SOA (pour architecture orientée service) définie un modèle d'interaction applicative mettant en œuvre des connexions en couplage lâche entre divers composants logiciels (ou agents) [w21].

5.3 Les limites des architectures orientées objet par rapport à la SOA:

Cette partie exprime la différence entre les deux concepts logiciels clés (SOA et Orienté Objet), qui concernent à la fois les méthodes et l'architecture des systèmes [w22].

Nous avons résumé la déférence dans les points cités aux tableaux suivant :

L'architecture orientée objet (OO)	L'architecture orientée service (OS)
Au sein d'une architecture orientée objet, les données manipulées sont directement associées au mode de traitement qui leur est appliqué (cf. les méthodes et les classes du langage Java) [w21].	Ce n'est pas le cas au sein d'une architecture de SOA dans laquelle ces deux éléments sont dissociés [w21].
L'orientée objet comme une approche d'ingénierie logicielle pour viser l'agilité du	La SOA doit être employée comme un moyen d'architecture technique [w22].

SI [w22].	
Dans l'orienté objet le terme de réutilisation vue comme l'héritage entre les classes [w23].	réutilisation d'outil (ou produit) des entreprises. C'est une approche permettant de réutiliser et d'organiser des ressources existantes, dans une solution autorisant une interopérabilité entre plateforme et environnement, une évolutivité des modules applicatifs et une flexibilité autorisant l'utilisation dynamique d'application [w24].

Tableaux 2.1: *La différence entre l'orientée objet et la SOA.*

La SOA fournit une meilleure base d'intégration des nouveaux systèmes d'application.

5.4 Définition de la migration:

La migration ou la modernisation logicielle implique des changements plus étendus qu'une simple maintenance, mais une partie significative du système est conservée. Ces changements incluent souvent la restructuration du système, l'amélioration des fonctionnalités ou la modification des attributs du logiciel [9].

5.5 Stratégies d'évolution vers une architecture orientée service:

Plusieurs travaux ont été proposés pour faire évoluer les systèmes patrimoniaux, vers une architecture SOA, la majorité tourne autour de la réingénierie. Différentes classification ont été discutées dans [12]. Puis une nouvelle classification qui répartit les différentes stratégies de migrations vers SOA en trois catégories a été proposée à savoir :

5.5.1 Approches décisionnelles:

Ce sont des approches qui aident les entreprises à prendre des décisions, après étude des coûts, du contexte, de la faisabilité pour décider est ce qu'il est rentable ou faisable de migrer, d'intégrer, de remplacer ou de ne rien faire. Puis si on opte pour une des solutions, comment l'appliquer, quelles techniques utiliser. La critique qui peut être faite à cette catégorie, c'est qu'elle ne comporte pas de phase implémentation et reste finalement une approche d'aide à la décision [12].

5.5.2 Approches partielles:

Elle englobe des approches qui proposent des solutions partielles en utilisant les techniques de réingénierie pour analyser et récupérer l'architecture et les fonctionnalités du système patrimonial pour enfin avoir comme résultat un nouveau système orienté service, en utilisant l'intégration ou la migration. Mais sans faire une analyse préalable de la faisabilité de l'évolution [12].

5.5.3 Approches techniques:

Elles se basent sur l'aspect technique tel que les méthodes d'analyse de code, d'identification et d'extraction des services, comment les intégrer avec le SP (Système Patrimonial), comment tester les nouveaux services,….etc [12].

5.6 Les approches de migration vers SOA:

Au niveau des approches, il existe trois angles différents pour la mise en place d'une Architecture orientée services qu'on va présenter dans le paragraphe suivant.

- **L'approche bottom up:**

Le premier angle d'attaque part de la couche technique de l'entreprise. Il s'agit dans ce cas d'extraire les services à partir des applications existantes. Le retour de cette démarche est assez clair et bien établi. On aboutit à un ensemble de services techniques dont le processus d'identification est assez facile, avec par contre un niveau de granularité assez fin et une profusion de service trop importante ce qui engendre une grande difficulté pour s'aligner avec le métier.

- **L'approche top down:**

Le deuxième angle d'attaque est relié directement au métier de l'entreprise. Il consiste à identifier des services réutilisables en se basant sur les descriptions du niveau métier de l'entreprise. Le résultat est un ensemble des services réutilisables de point de vue métier. Cependant, cette méthode néglige l'aspect architectural de service. En effet, un service est une notion d'architecture et le fait de définir des services à partir du niveau métier n'implique pas forcément que ces derniers peuvent être projetés sur le niveau technique afin d'assurer leur fonctionnement.

Les méthodes top-down et bottom-up ne nous permettent pas d'aboutir à une identification de services fiables répondant à la finalité de l'entreprise orientée service. La première aboutit à un ensemble de services très abstraits avec aucune garantie pour leur réalisation du point de

vue technologique, quand à la deuxième, elle concourt à un ensemble des services très techniques qui n'implique pas forcément un alignement avec les besoins métier.

- **L'approche meet in the middle:**

Le troisième angle d'attaque est le "meet in the middle" qui mène en parallèle une approche top down pour définir des services de haut niveau nécessaires à la réalisation des processus métiers, ainsi qu'une approche bottom-up dans le but de cartographier l'existant applicatif de l'entreprise pour supporter les services métiers. Ensuite, un croisement entre les deux résultats est fait afin de déterminer comment seront réalisés les processus métiers. Cette démarche paraît la plus intéressante et la plus concrète [11].

5.7 Les bénéfices de la migration:

La migration vers la SOA offre plusieurs bénéfices aux entreprises, parmi ces bénéfices nous citons :

- L'adaptation aux nouveaux besoins.
- L'amélioration des services clients.
- Une intégration plus étroite avec les partenaires et les fournisseurs.
- La réduction du coût d'usage des systèmes d'information.
- L'amélioration de la qualité des données.
- L'amélioration du contrôle et de la gestion de sécurité.
- L'augmentation de la flexibilité et la réactivité des systèmes d'information [10].

5.8 Les difficultés de la migration:

Les efforts de la modernisation des systèmes d'information des entreprises ont échoué dans la plupart du temps à cause de plusieurs facteurs, nous citons les suivants :

- **La complexité des systèmes patrimoniaux:** la complexité est considérée comme le plus grand limiteur du processus de migration, elle est produite à cause de la taille énorme de système, de l'incompréhensibilité des systèmes à migrer et des phases successives de maintenance.
- **Les risques de migration:** les risques de migration ne sont pas majeurs, il est possible d'accepter un certain risque si nous devons accomplir une tâche de migration. Malheureusement, beaucoup d'entreprises sont incapables ou ne veulent pas contrôler le risque correctement. Ceci peut également provenir de l'insuffisance de compréhension de la gestion des risques et des techniques de réduction du risque [10].

6 Conclusion:

Nous avons traité dans ce chapitre les concepts de base de la réingénierie et la migration, la première s'applique aux systèmes d'information, les processus des entreprises ainsi que les logiciels afin de les améliorer.

La transformation des systèmes patrimoniaux se fait en quatre phases : une phase d'analyse (La rétro-ingénierie), une phase de conception (Recomposition), une phase de création d'une représentation sémantique (Redocumentation) et enfin une phase de transformation (Restructuration).

La migration vers une architecture SOA permet de garder et réutiliser les parties intéressantes qui supportent les besoins métiers du système intégré et en même temps d'accompagner l'évolution technologique.

Plusieurs stratégies et approches existent dans la littérature, elles peuvent converger ou diverger, mais toutes ont une étape commune et essentielle dans le processus de migration qui est l'identification des services. L'identification peut se faire de façon manuelle ou automatique. Dans notre travail, nous avons essayé de focaliser notre effort pour automatiser cette phase d'identification.

Le prochain chapitre a pour objectif de présenter une étude sur quelques algorithmes qui nous permettent d'automatiser l'identification.

Chapitre 3 :

Les algorithmes de classification non supervisée

1 Introduction:

Dans notre travail nous nous sommes concentrés sur la phase d'identification des services qui est une phase primordiale dans le processus de migration vers une architecture orientée service. Cette identification se fait généralement de façon manuelle. Dans notre cas nous avons fait le choix d'automatiser cette étape. Afin de le faire nous avons utilisé des algorithmes de classifications qui permettent de regrouper de façon automatique les classes qui sont les composants de base d'une application orientée objet dans des groupes qui constituerons par la suite des services homogènes.

Nous avons dû chercher parmi les algorithmes existants celui qui convient le mieux à notre objectif.

Dans ce qui suit nous présenterons une famille d'algorithme de classification non supervisé.

2 La classification:

De nos jours, la classification est une démarche qui est appliquée dans d'innombrables domaines. Elle consiste à regrouper les objets d'un ensemble de données X de nature quelconque en un nombre restreint de classes homogènes.

Il existe deux types de classification : la classification supervisée et la classification non supervisé. Nous parlerons de classification non supervisée, ou regroupement, lorsque l'on ne dispose d'aucune information à priori sur les objets à traiter, et de classification supervisée, ou classification tout court, dans le cas contraire. Il arrive parfois, que l'information disponible soit incomplète, et on parle alors d'environnement partiellement supervisé [13][w27].

2.1 Définition:

C'est une discipline scientifique qui trouve dans un ensemble d'objet des groupes homogènes (classes) et bien distincts les uns des autres [w28].

2.2 Les types de classification:

On distingue essentiellement deux types de classifications : supervisée et non supervisée.

2.2.1 La classification supervisée:

Consiste à définir une fonction qui attribue une ou plusieurs classes à chaque donnée. Dans cette approche on suppose qu'un expert fournit auparavant les étiquettes pour chaque donnée, les étiquettes sont des classes d'appartenance [w32].

Selon [18] : la classification supervisée (appelée aussi classement ou classification inductive) a pour objectif « d'apprendre » par exemple.

Elle cherche à expliquer et à prédire l'appartenance de documents à des classes connues a priori.

Leur algorithmes reposent sur des arbres de décision, des réseaux de neurones, la règle de Bayes, les k plus proches voisins…

2.2.2 La classification non supervisée:

La classification non supervisée (clustering, segmentation) est un outil très performant pour la détection automatique de sous-groupes pertinents (ou clusters) dans un jeu de données ou d'objet [w30]. C'est un traitement sur un ensemble d'objets qui n'ont pas été étiquetés par un superviseur (expert humain, caractère répétitif, le système visuel humain . . .)[w32].

Les membres d'un même cluster doivent êtres similaires entre eux, contrairement aux membres des clusters différents (homogénéité interne et séparation externe) [w30].

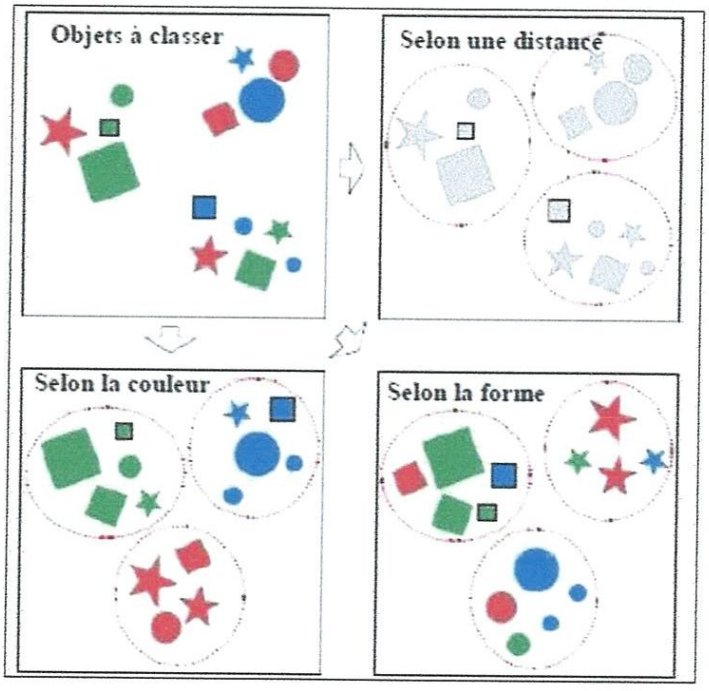


Figure 3.1 : *Classification non supervisée selon différents critères de similarité [21].*

2.3 La similarité et la dissimilarité:

Un calcul de proximité peut mesurer la similarité ou la dissimilarité : plus deux objets se ressemblent, plus leur similarité est grande et plus leur dissimilarité est petite.

La ressemblance peut être mesurée par la distance qui existe entre deux objets ou par la relation entre les attributs de ces objets [14]. Dans la similarité, et selon le type et la représentation des objets, on retrouve plusieurs distances.

Dans ce qui suit, nous présentons quelques-unes entre deux objets d(x1 ; x2).

Distance de Minkowski:

$$d(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^q\right)^{\frac{1}{q}}$$

Distance euclidienne:

$$d(x,y)=\sqrt{\left(\sum_{i=1}^{n}|x_i-y_i|^2\right)}\,(\text{Minkowski. } q=2)$$

Distance de Manhattan:

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|\,(\text{Minkowski. } q=1)$$

Distance maximum:

$$d(x,y)=\max_{i=1}^{n}|x_i-y_i|\,(\text{Minkowski. } q\to\infty)$$

Les distances citées ci-dessus sont exposées en détail dans [15].

2.4 Quelques algorithmes de classification non supervisée:

Les méthodes de classification non supervisée peuvent être regroupées en cinq grandes catégories : les méthodes hiérarchiques, les méthodes de clustering par partitionnement, les méthodes basées sur la densité des objets dans leur espace de représentation, les méthodes basées sur le réseau de neurone et les méthodes probabilistes. Cette taxonomie est présentée dans le tableau suivant [19] [20]:

Les approches de classification non supervisée	Les algorithmes
Approche hiérarchique.	Classification hiérarchique ascendante CAH, Classification hiérarchique descendante CURE,

	BIRCH, l'algorithme deWard, etc.
Approche par partitionnement.	IsoData, Fuzzy C-Means, Fast Global K-Means, K-Means++, etc.
Approche de clustering basé sur la densité.	Soustractive clustering, Denclust, Mean-scihft, Density based spatial clustering of applications with noise (DBSCAN), etc.
Approche de clustering basé sur le réseau de neurone.	Kohonen, Adaptive Resonance, etc.
Approche de clustering probabiliste.	Expectation maximisation (EM).

Tableau 3.1: *Taxonomie des méthodes de clustering [19][20].*

Dans ce qui suit nous allons présenter quelques algorithmes de ces catégories.

2.4.1 L'algorithme hiérarchique:

La méthode hiérarchique se présente comme la succession des partitions emboîtées [16]. Elle consiste à calculer une matrice exprimant les distances mutuelles entre les points à classer, puis, selon le type de l'hiérarchiquement choisis (ascendant ou descendant) soit on supposé chaque individué comme un cluster et on fusionne à chaque étape les deux clusters les plus proches jusqu'à l'obtention d'un seul cluster. Ou bien on procède de façon inverse. On considère l'ensemble des données comme un gros cluster unique, et le scinde en deux clusters "descendants", tel que la distance entre les deux clusters soit le plus grande possible jusqu'à ce qu'il ne reste plus que des clusters qui contient un seul individu.

Le résultat de cette méthode est un arbre de partition successive appelé dendrogramme [w32], comme celui de la figure 3.2.

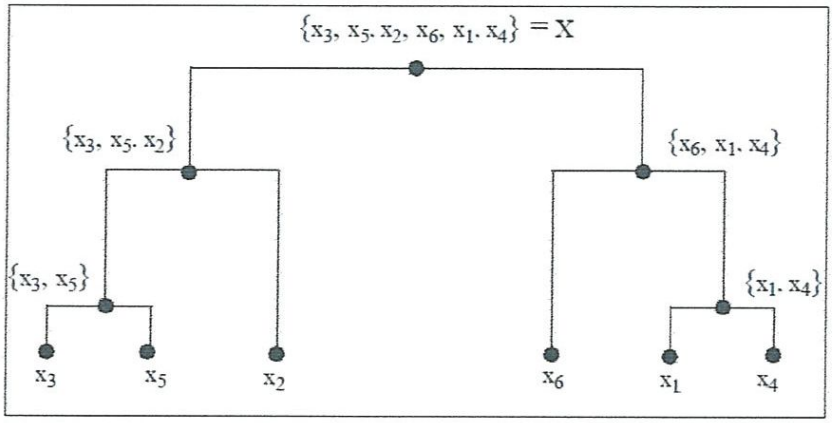


Figure 3.2 : *Partitions emboîtées d'un ensemble X à 6 éléments [13].*

2.4.1.1 L'algorithme hiérarchique ascendant:

L'algorithme hiérarchique ascendant est décrit dans les étapes suivantes :

Au départ, on dispose de n objets à classer.

1. Associer chaque objet à classer à un nouveau cluster.
2. Calculer la matrice des distances qui sépare toutes les clusters deux à deux.
3. Chercher les deux clusters les plus proches.
4. Réunir les deux clusters les plus proches dans un seul nouveau cluster.
5. Mettre à jour la matrice des distances.
6. Réitère les étapes (3-5) jusqu'à ce que tous les clusters aient été regroupés.

Figure 3.3 : *L'algorithme hiérarchique ascendant [13].*

Selon la façon avec laquelle les clusters sont fusionnés, plusieurs algorithmes ont été réalisés.

Les algorithmes les plus utilisés dans la plupart des méthodes hiérarchiques ascendantes sont les algorithmes du lien simple ou saut minimum (single link), les algorithmes du lien ou diamètre complet ou maximal (complete link), les algorithmes du lien moyen (average link).

- Dans l'algorithme du lien simple, la distance entre deux clusters est la valeur minimum des distances entre toutes les paires d'objets, l'un du premier cluster, l'autre du deuxième.
- Dans l'algorithme du lien complet, la distance entre deux clusters est la valeur maximale des distances entre toutes les paires d'objets.

- Dans l'algorithme du lien moyen, la distance entre deux clusters est la valeur moyenne des distances entre toutes les paires d'objet, l'un du premier cluster, l'autre du deuxième [21].

2.4.1.2 L'algorithme hiérarchique descendant:

On procède selon les étapes suivantes:

Soit I l'ensemble des objets

1. Calculer les distances des éléments de I deux à deux et trier les valeurs par ordre décroissant.
2. Evaluer la distance max $d(i_1, i_0)$ ou chaque élément de I est associé à un cluster C_0 représenté par i_0 ou C_1 représenter par i_1.
3. On considère le cluster C_i qui possède la distance maximale et on divise C_i en C_i^a et C_i^b puis on attribuant chaque élément de C_i à C_i^a ou C_i^b.
4. Recalculer les distances par ordre décroissant on s'arrête dès que le nombre des clusters égal la cardinalité de I.

Figure 3.4 : *L'algorithme hiérarchique descendant [21].*

2.4.2 L'algorithme Soustractive Clustering (SC):

Cette technique est appliquée lorsqu'il n'y a pas une idée claire sur le nombre des centres pour la répartition des données.

La méthode Soustractive Clustering est une extension de la méthode de classification proposée par Yager [17]. Il suppose que chaque point de données est un centre potentiel de l'amas et calcule le potentiel de chaque point de données par la mesure de la densité des points de données l'entourant.

L'algorithme sélectionne d'abord le point de données avec le plus grand potentiel en tant que centre de l'amas, puis évince les points de données à proximité du centre du premier groupe (déterminer par un rayon), puis calculer le centre prochain et ainsi de suite [22].

L'algorithme est fonctionné comme se suit:

1. Choisir le point de donnée avec le plus grand potentiel d'être le centre du premier groupe.
2. Évincer tous les points dans le voisinage du 1er centre de l'amas (déterminé par le rayon).
3. Déterminer le prochain centre.
4. Itérer le processus jusqu'à ce que toutes les données soient dans un rayon du centre d'un amas.

Figure 3.5 : *L'algorithme Soustractive Clustering [22].*

Nous pourrons dire que l'algorithme SC réduit la complexité de calcul et donne une répartition des centres de cluster en fonction de la mesure de densité ainsi que du rayon (voir figure 3.6), sachant que chaque centre est un point de données avec le plus grand potentiel [22].

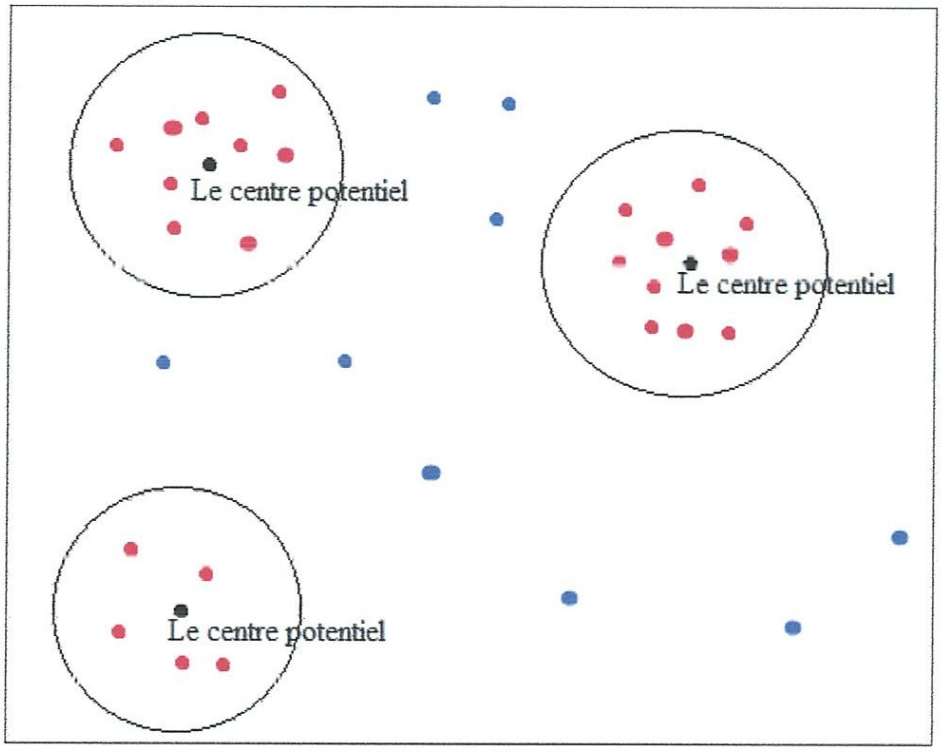


Figure 3.6 : *La distribution des données autour des centres d'amas dans l'algorithme Soustractive Clustering.*

2.4.3 L'algorithme de Kohonen:

L'algorithme regroupe les observations en classes, en respectant la topologie de l'espace des observations. Cela veut dire qu'on définit à priori une notion de voisinage entre classes et que des observations voisines dans l'espace des variables (de dimension p) appartiennent (après classement) à la même classe ou à des classes voisines. Les voisinages entre les classes peuvent être choisis de manière variée, soit en suppose que les classes sont disposées sur une grille ou soit sur une ficelle qui définit naturellement les voisins de chaque classe [w31].

Le voisinage pour une grille :

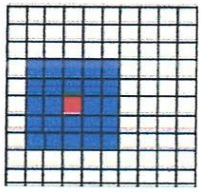


Figure 3.7 : *Le voisinage de rayon 2 pour une grille [w31].*

Le voisinage pour une ficelle :

Figure 3.8 : *Le voisinage de rayon 2 pour une ficelle [w31].*

On résume le déroulement de l'algorithme selon les étapes suivantes :

1. L'initialisation: associer à chaque classe un vecteur code dans l'espace des observations choisi de manière aléatoire.
2. Ensuite, à chaque étape, on choisit une observation au hasard, on la compare à tous les vecteurs codes, et on détermine la classe gagnante, c'est-à-dire celle dont le vecteur code est le plus proche au sens d'une distance donnée à priori.
1. On rapproche alors de l'observation les codes de la classe gagnante et des classes voisines.
2. Répéter l'étape 2, 3 jusqu'à que toutes les observations soient affectées à une classe.

Figure 3.9 : *L'algorithme de Kohonen [w31].*

2.4.4 L'algorithme IsoData:

Dans la méthode K-means, le nombre de clusters K reste le même pendant toutes les itérations. Cet inconvénient peut être surmonté dans l'algorithme IsoData (Itératif auto-organisation Analyse Technique Algorithm), qui permet de régler automatiquement le nombre des clusters lors de l'itération.

L'algorithme utilise également un certain nombre des paramètres pour déterminé les clusters à fusionner ou à diviser [w34].Ces paramètres sont :

K : Le nombre initial des clusters.

I_{max}: Le nombre maximal des itérations.

σ_{max} : L'écart type maximum.

L_{min}: La distance minimale entre deux centres de clusters [w32].

On résume l'algorithme suit les étapes suivantes :

1. Appliquer l'algorithme de k-means.
2. Si l'écart type d'un segment dépasse le maximum spécifié (σ_{max}), le segment est éclaté en deux segments.
3. Si la distance entre deux clusters (entre les centres des clusters) inferieur a L_{min} alors fusionner les deux clusters.
4. Aller à l'étape 1 jusqu'à I_{max} ou attendre quand les membres des groupes ne change plus.

Figure 3.10 : *L'algorithme IsoData [w33].*

Sachant qu'on peut utiliser un seul seuil tel que si la distance entre deux clusters inferieur à celle on doit la fusionner et dans le cas contraire on le segmenter.

Et voici l'algorithme k-means :

1. Choisir au hasard les centre de k clusters.
2. Attribuer chaque donner au centre le plus proche.
3. Recalculer les positions des nouveaux centroïdes.
4. Répéter les étapes 2 et 3 jusqu'à convergence, c'est-à-dire jusqu'à ce que les centroïdes ne bougent plus.

Figure 3.11 : *L'algorithme de K-means [w33].*

2.4.5 L'algorithme EM (Expectation-Maximisation):

En français « L'algorithme d'espérance-maximisation », souvent abrégé par EM. Son objectif est de trouver le maximum de vraisemblance des paramètres de modèles probabilistes (comme le modèle gaussien). On compte une multitude de domaines d'applications de cet algorithme, à titre d'exemple : il est usuellement utilisé dans la vision artificielle ou encore dans le traitement d'images, plus spécifique en ce qui concerne la segmentation (image médicale, satellitaire . . . etc.), ou tout simplement en clustering pour regrouper les données homogènes dans un groupe, . . . etc.

Cet algorithme comprend deux étapes essentielles :

1. E-steps (E) : une étape d'évaluation de l'espérance, c'est dans cette étape qu'on calcule l'espérance de la vraisemblance en tenant compte des dernières variables observées.

2. M-steps (M) : une étape de maximisation de vraisemblance qu'on a trouvée à l'étape(E), en tentant d'estimer le maximum de vraisemblance des paramètres [w32].

Le déroulement de l'algorithme est le suivant:

1. Initialisation au hasard de $\Phi^{(0)}$.
2. M=0.
3. Tant que EM n'a pas convergé faire :
   - Evaluer l'espérance (E-steps) : $Q(\Phi, \Phi^{(M)})$.
   - Maximisation (M-steps): $\Phi^{(M+1)}=\text{argmax}_{\Phi}\Phi(Q(\Phi,\Phi^{(M)}))$.
   - M=M+1.

Figure 3.12 : *L'algorithme EM [w32].*

Φ : représente les paramètres à estimer.

3 Conclusion:

La classification est un moyen utile d'organisation et de hiérarchisation des données. On constate deux méthodes de classification une classification supervisée qui consiste à analyser des nouvelles données et à les affecter, en fonction de leurs caractéristiques ou attributs, à telle ou telle classe prédéfinie. Et une classification non supervisée où le but est de trouver des groupes compacts et bien séparés pour un ensemble de données, donc d'affecter à chaque

observation une étiquette de classe qui matérialise l'appartenance de celle-ci aux classes dégagées.

Dans ce chapitre nous avons présenté quelques algorithmes de classification non supervisée qui correspondent à nos besoins en présentant leurs principes. Le choix des algorithmes de classification non supervisée est motivé par le fait que dans notre travail nous démarrons avec un nombre de classes d'une application orientée objet que nous voulons répartir sur un ensemble des groupes non défini au préalable. La répartition des classes est guidée par les caractéristiques de qualité d'un service. Ainsi nous obtiendrons en final un nombre des services candidats.

Dans le chapitre suivant nous donnerons les détails de conception de notre solution pour l'identification des services à partir d'une application orientée objet en utilisant les algorithmes de classification.

Chapitre 4 :

Conception

1 Introduction:

La réalité est que l'architecture orientée services (SOA) est actuellement la meilleure option disponible pour l'intégration et l'optimisation des systèmes patrimoniaux (existants). Selon un rapport de 2007 du Gartner Group, 50% des nouvelles applications à mission critique opérationnelles et des processus métier ont été conçus en 2007 autour de SOA, et ce nombre sera plus grand que 80% en 2010. Bien que les technologies de mise en œuvre de la SOA aillent probablement changer au fil du temps, un seul concept restera: SOA promet une façon de concevoir des systèmes qui permet l'efficacité des coûts, l'agilité, l'adaptabilité et la promotion des investissements hérités.

Notre travail se situe dans le cadre de la réingénierie et de la modernisation des applications patrimoniales (dans notre cas orientée objet) vers une nouvelle architecture orientée service.

Bien que les stratégies et approches de migration vers la SOA soient multiples et différentes, toutes s'accordent sur l'importance d'une phase essentielle qui est la phase d'identification de service.

2 L'objectif global:

Nous nous sommes fixé l'objectif d'automatiser cette phase cruciale qui est l'identification de service à partir d'application orientée objet.

Une application orientée objet est constituée d'un ensemble des classes, pour migrer vers une architecture orientée service nous avons besoin de regrouper les classes qui sont fortement couplées et fortement cohésives pouvant constituer un module autonome, réutilisable et distribuable, donc un groupe des classes qui respecte la définition d'un service de qualité.

Afin d'atteindre ce but nous avons explorer les algorithmes de classification qui pourraient nous permettre d'atteindre ce but, et nous nous somme intéressé plus particulièrement aux algorithmes de classification non supervisées puisque nous ne pouvons savoir à l'avance le nombre de groupe des classes constituant des services à l'avance.

Afin de faciliter et automatiser l'identification des services à partir du code source, nous avons choisi d'adapter quelque algorithme décrit dans le chapitre précédent (Soustractive Clustering, IsoData) pour regrouper l'ensemble des classes homogènes pour former un service tout en respectant les caractéristiques de la SOA (l'autonomie, la composabilité, le couplage lâche…).

3 Processus d'identification des services:

Notre objectif est d'automatiser la transformation des applications orientées objet vers des applications orientées service, donc il faudrait bien décomposer l'ensemble des fonctionnalités fournies par une application orientée objet global en un ensemble de fonctionnalités cohérentes basics appelées services.

Allons de ce principe nous avons fixé les étapes suivantes :

- **La mise en correspondance Objet-Service:** Un travail préliminaire avant de définir le processus d'identification de service s'est imposé, c'est de recenser les caractéristiques des éléments de base des deux formalismes qui sont la classe (ou objet) et le service afin de faire une correspondance entre les deux.
- **Analyse du code source:** dans cette étape nous avons analysé le code source d'une application orienté objet afin d'extraire toutes les expressions les appels des méthodes, les utilisations des attributs...etc., afin dc récupérer les caractéristiques à travers lesquelles nous identifierons les services.
- **Identification des services:** décomposé une application en un ensemble de services, en regroupant les classes similaires à l'aide des algorithmes hiérarchique, Soustractive Clustering et IsoData afin de construire un service.

Pour passer de la phase d'analyse à la phase d'identification nous avons suivi les étapes suivantes.

- ✓ **Calcul des métriques:** Il s'agit de quantifier et de mesurer les propriétés des services, dans notre cas le couplage et la cohésion entres les classes candidates pour former un service.
- ✓ **Définition de la fonction objectif:** Combiner les métriques proposées pour avoir une fonction globale qui mesure les propriétés d'un service.
- ✓ **La matrice symétrique:** C'est le résultat de l'application de la fonction objectif sur les différentes classes de l'application orientée objet.

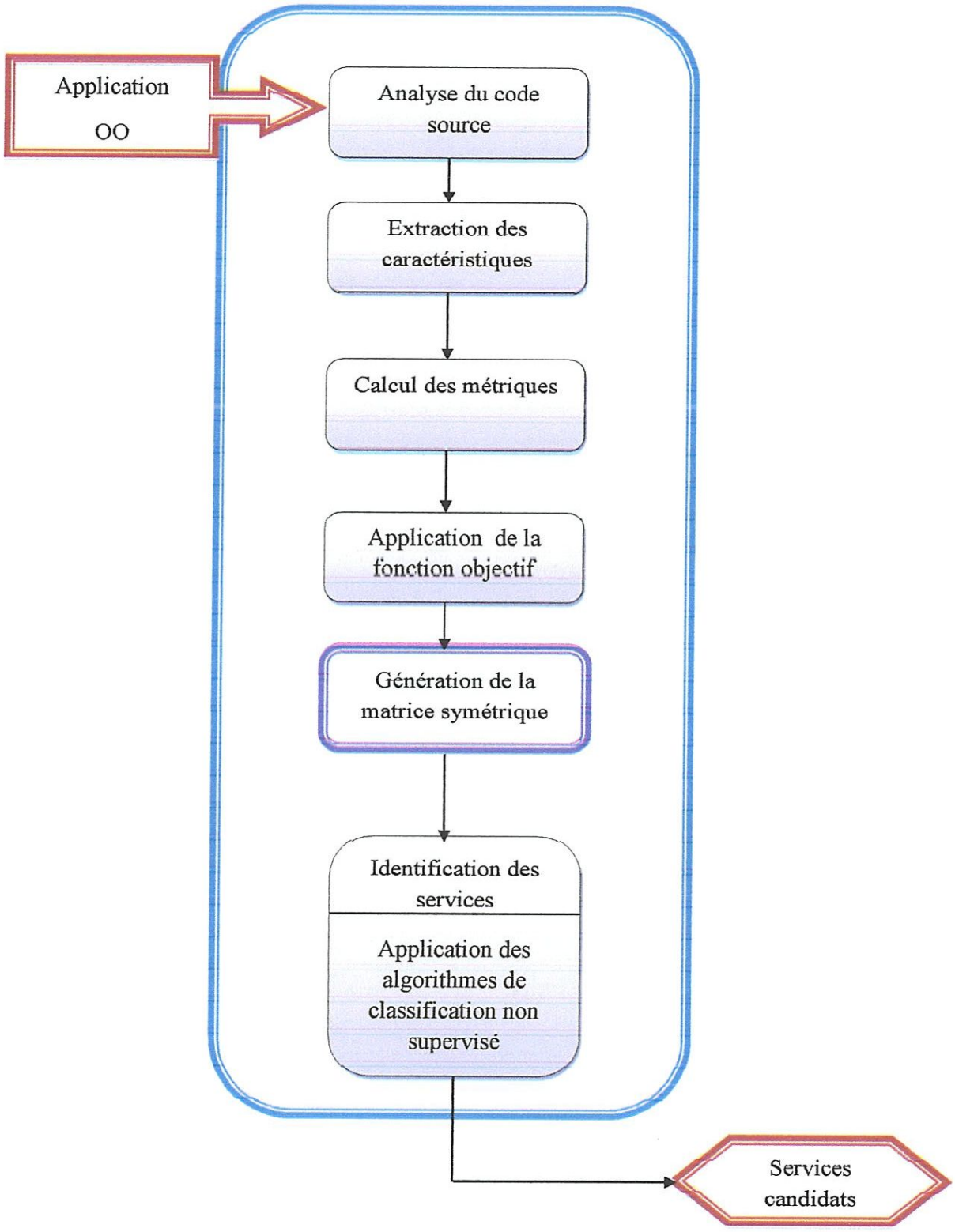


Figure 4.1 : *Processus général d'identification des services.*

3.1 La mise en correspondance Objet-Service:

Afin de pouvoir passer de l'objet vers le service nous avons étudié les éléments correspondance entre les deux paradigmes.

Nous considérons un service comme un groupe de classes définies dans l'orienté objet. Parmi ces classes il y'a celles qui définissent des opérations (classes interface), l'ensemble des classes interface forment l'interface du service.

L'interface du service expose un ensemble d'opérations correspondant à des méthodes publiques d'une classe OO.

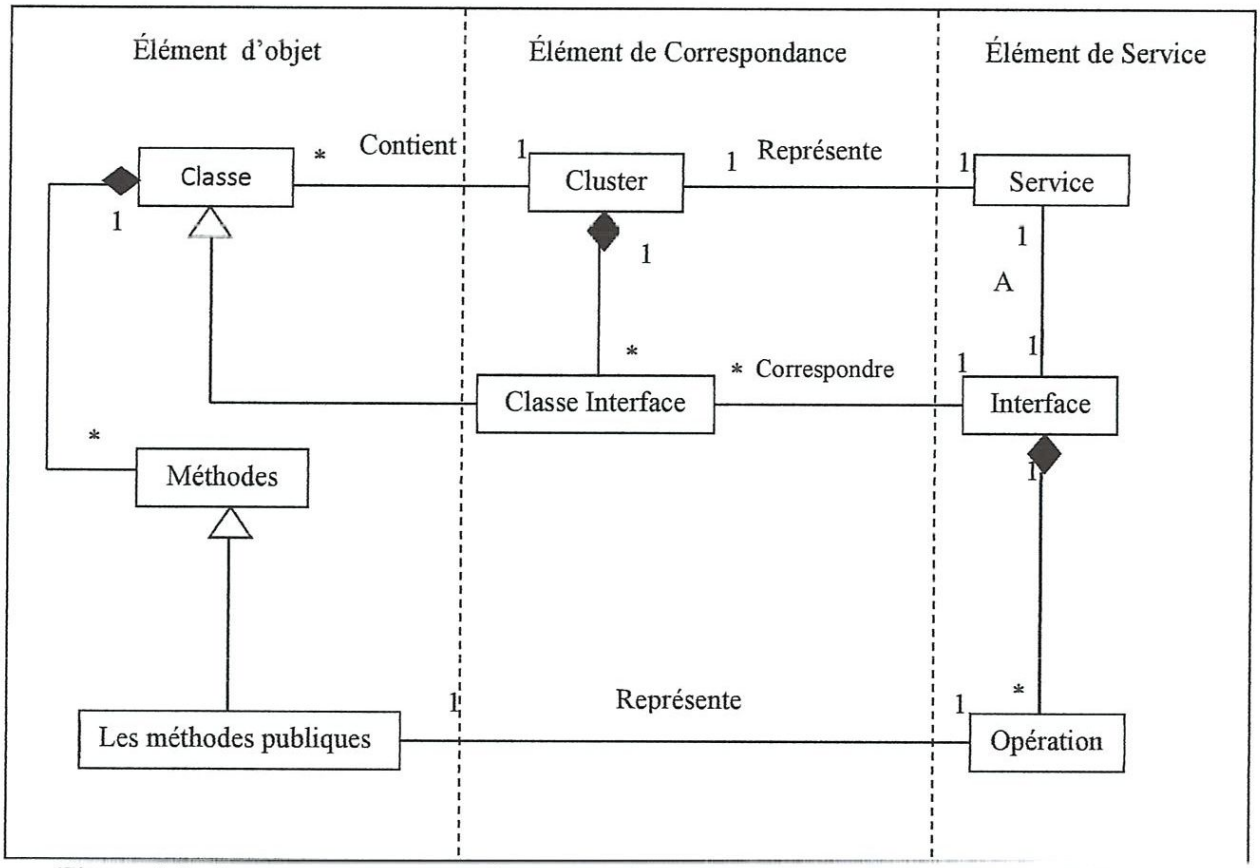


Figure 4.2 : *La mise en correspondance entre l'orienté objet et l'orienté service [31].*

3.2 L'analyse du code source:

L'étape d'analyse permet d'extraire les informations d'une application donnée. Ces informations concernant les classes qui composent un projet, et qui nécessite un parcours de la totalité du projet pour atteindre ces derniers.

A partir du code source, nous parcourons les informations utiles dans les instructions simples (l'affectation, les appels des méthodes, la création des instances...etc) et composées (les

structures conditionnel {si ...sinon}, les boucles {pour, tant que et répéter}, les structures de choix ...etc) qui comportent des blocs d'instruction.

De plus nous extrayons les signatures des méthodes (les noms, les paramètres et les types retournés) pour distinguer les cas des méthodes concernées par le polymorphisme et la surcharge. Toutes ces informations vont être utilisées plus tard dans le calcul des métriques.

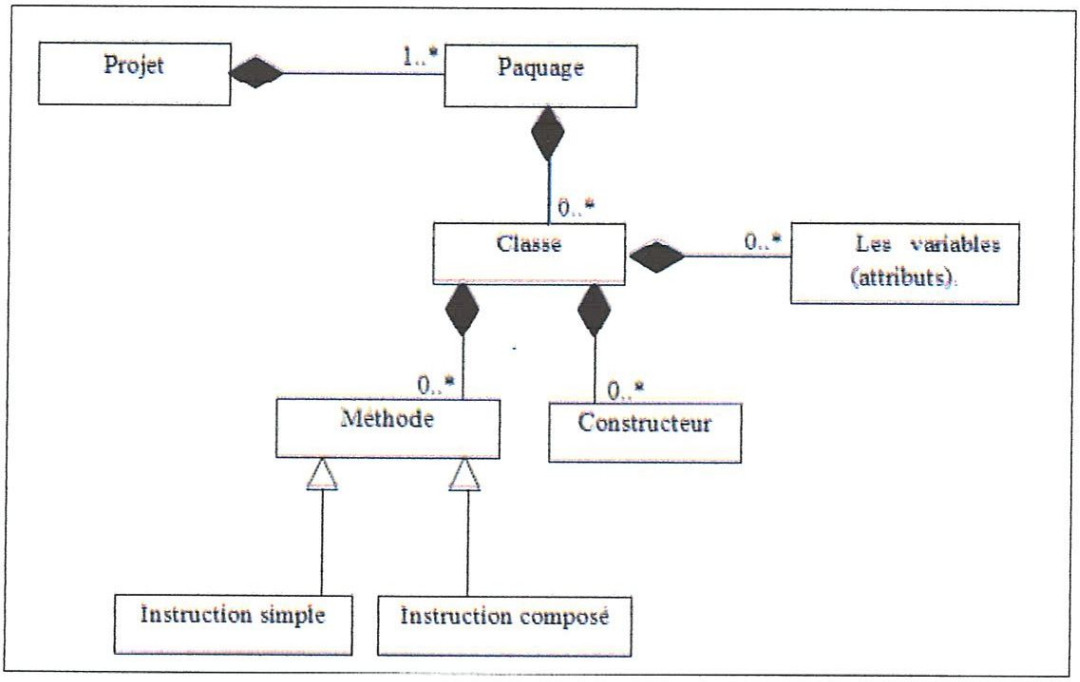


Figure 4.3 : *La structure des applications orientées objet.*

3.3 Le calcul des métriques:

Cette partie est basée sur l'étude des éléments de base de notre application (les classes) afin de chercher les métriques qui mesurent la similarité entre ces éléments.

Puisque le résultat attendu est d'identifier des services à partir d'un ensemble de classes, nous devons regrouper les classes qui, ensemble soient caractérisées par les caractéristique des services. Ces caractéristiques doivent être mesurées à travers des métriques.

Dans notre cas la similarité ne s'agit pas de la distance traditionnelle entre les classes mais il s'agit de l'homogénéité des fonctionnalités offertes par ces classes et de l'interdépendance entre ces dernières.

Nous voulons avoir à la fin des services composés d'un ensemble de classes qui partagent des fonctionnalités homogènes, des groupes de classes qui soient fortement dépendantes entre

elles, des groupes de classes qui peuvent être aussi autonomes, réutilisables, distribuables, accessibles à distance ,..etc.

Afin de mesurer ces caractéristiques, nous avons trouvé que les métriques de couplage et de cohésion conviennent bien à nos besoins.

3.3.1 Mesure de couplage:

C'est le degré d'interdépendance entre modules. Cette notion est très utilisée dans le génie logiciel et en particulier dans le paradigme objet [24].

Il existe de nombreuses formules qui permettant de mesurer le couplage des classes avec l'extérieur [25, 26, 27, 28, 29, 30]. Une classe est couplée avec une autre classe si elle est en relation avec une ou plusieurs parties de code source de celle-ci.

Voici quelques mesures de couplage :

- NIH ICP [29]: mesure le nombre d'appels de méthodes des classes avec lesquelles une classe n'a pas de relation d'héritage.
- DAC [30]: mesure le nombre d'attributs de la classe qui ont pour type une autre classe.
- OCMIC [25]: mesure le nombre des paramètres qui ne sont pas de type a ou primitifs dans les méthodes de la classe a.

Afin de couvrir la métrique de couplage, nous avons combiné ces trois formules dans la formule suivante, étant donné que le couplage est une relation binaire entre deux classes (A, B) :

$$\text{Couplage}(A, B) = \text{NIH\_ICP}(A, B) + \text{DAC}(A, B) + \text{OCMIC}(A, B).$$

Nous avons aussi constaté que l'héritage représente une relation de couplage très forte c'est-à-dire:

Si la classe A est une sous classe de la classe B alors :

Couplage (A, B) : doit avoir la plus grande valeur possible.

Exemple 1 :

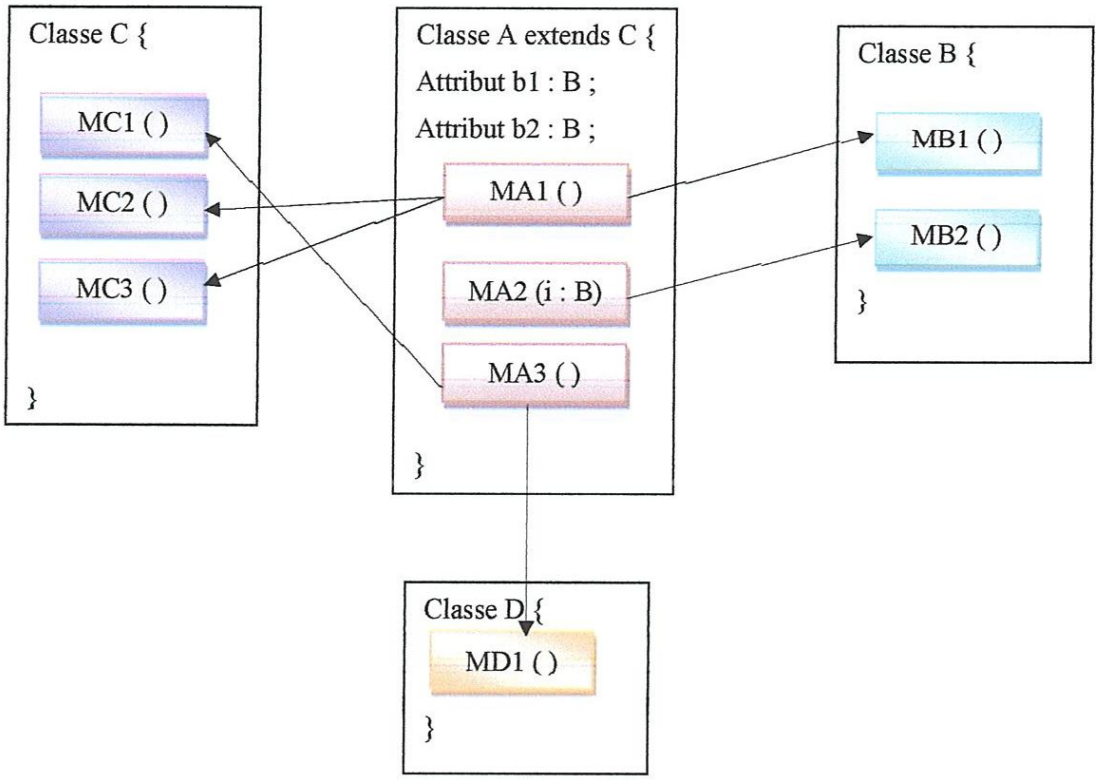


Figure 4.4: *Exemple de couplage entre quatre classes A, B, C, D.*

Les flèches indiquent qu'une méthode qui fait appel à une autre méthode ou bien une méthode qui utilise un attribut d'une autre classe, la direction de la flèche indique la direction d'appel.

Le couplage (A, B) diffère du couplage (B, A) et il peut être calculé comme suit :

Le couplage pour les classes (A, B) est:

NIH_ICP (A, B) =2.

DAC (A, B) =2.

OCMIC (A, B) =1.

Couplage (A, B) = 2+2+1 => Couplage (A, B) =5.

Le couplage pour les classes (A, D) est:

NIH_ICP (A, D) =1.

DAC (A, D) =0.

OCMIC (A, D) =0.

Le couplage (A, D) = 1+0+0 => Couplage (A, D)=1.

Le tableau suivant résume le résultat de calcul :

Les classes	A	B	D
A	0	5	1
B	0	0	0
C	0	0	0
D	0	0	0

Tableau 4.1: *Tableau des résultats de couplage de l'exemple1.*

Vu que le couplage se calcule entre deux classes distinctes, sa valeur sera nulle pour la classe elle-même.

$$\text{Couplage (A, B)} = \begin{cases} >0 \text{ si A dépend de B sur un ou plusieurs détaille .} \\ 0 \text{ Sinon .} \end{cases}$$

Le résultat de couplage pour une classe est un vecteur de taille n (n c'est le nombre des classes d'une application) et le couplage totale d'une application est une matrice de taille n*n.

3.3.2 Mesure de la cohésion:

La cohésion mesure la force de la collaboration au sein d'un ensemble d'éléments. Elle doit mesurer la cohésion d'un ensemble de classes ainsi que la cohésion d'une classe [24].

La métrique de cohésion prend en considération les relations directes et les relations indirectes.

- **Cohésion basée sur la relation directe:**

Deux méthodes Mi et Mj peuvent être directement connectées de différentes manières: Elles partagent au moins une variable d'instance en commun (relation UA : usage d'attributs), ou interagissent au moins avec une méthode de la même classe (relation IM invocation de méthodes). Donc, deux méthodes peuvent être directement connectées par un ou plusieurs critères.

$$UA_{Mi} \cap UA_{Mi} \neq \emptyset \text{ ou } IM_{Mi} \cap IM_{Mj} \neq \emptyset.$$

Considérons un graphe non dirigé G_D où chaque nœud représente une méthode de la classe. Il y'a un arc entre deux méthodes Mi et Mj si elles sont directement reliées. Soit E_D le nombre d'arcs dans le graphe G_D.

Le degré de cohésion dans la classe C basé sur la relation directe entre ses méthodes est défini par:

$$DC_D = |E_D| / [n*(n-1)/2].$$

Tel que DC_D : donne le pourcentage de paires de méthodes publiques qui sont directement reliées, et n c'est le nombre des méthodes [23].

Exemple 2 :

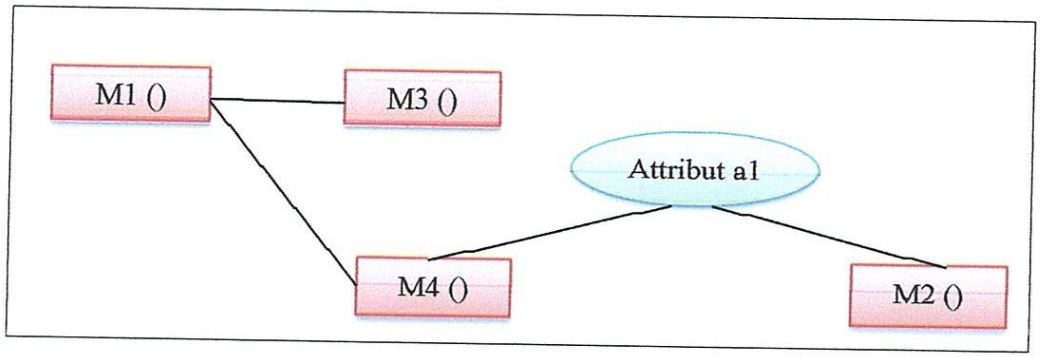


Figure 4.5 : *Exemple qui explique la relation de cohésion d'une classe C.*

Le nombre d'UA=1.

Le nombre d'IM = 2.

Le nombre total des méthodes =4.

$$\text{Donc } DC_D = 3/(4*3/2) \Rightarrow DC_D = 3/6.$$

- **Cohésion basée sur la relation indirecte:**

Deux méthodes Mi et Mj peuvent être indirectement reliées si elles sont directement ou indirectement reliées à unc méthode Mk.

Une méthode est indirectement reliée par une autre méthode s'il existe une séquence de méthodes M1, M2,...Mk telle que Mi est directement connectée à Mi+1 (i=1, k-1).

Considérons maintenant un graphe non orienté G_I où les sommets sont les méthodes publiques d'une classe C. Il y'a un arc entre deux sommets si les méthodes correspondantes sont directement ou indirectement reliées.

Soit E_I, le nombre d'arcs du graphe G_I, Alors, le degré de cohésion dans la classe C est défini par:

$$DC_I = |E_I| / [n*(n - 1) / 2].$$

DC_I : Donne le pourcentage de paires de méthodes publiques qui sont directement ou indirectement reliées [23].

Donc la cohésion d'une classe C est donnée par la formule suivante :

$$DC=DC_D+DC_I=>DC=|E_D|+|E_I| / [n*(n - 1) / 2].$$

Pour l'exemple 2 :

Puisque M3 en relation avec M1 qui est en relation avec M4 donc M3 est en relation indirecte avec M4 et M2 en relation indirecte avec M1 et M3 alors :

$$DC_I=3/ (4*3/2) => DC_I=3/6.$$

Par l'appliquant de DC sur l'exemple précédent:

$$DC=3+3/ (4*3/2) => DC=1.$$

Le résultat de la cohésion d'une classe est un vecteur de taille n (tel que n est le nombre des classes d'une application).

La cohésion interne concerne les attributs et les appels des méthodes entre les classes candidates à former un service. Puisque nous avons calculé la cohésion interne entre deux classes leur résultat est une matrice symétrique de taille n*n (n étant le nombre des classes).

Remarque:

$$\text{Cohésion } (A, B) = \text{Cohésion } (B, A).$$

$$0 \leq \text{Cohésion } (A_i, A_j) \leq 1.$$

Le calcul du couplage et de la cohésion peut être résumé par la figure 4.6.

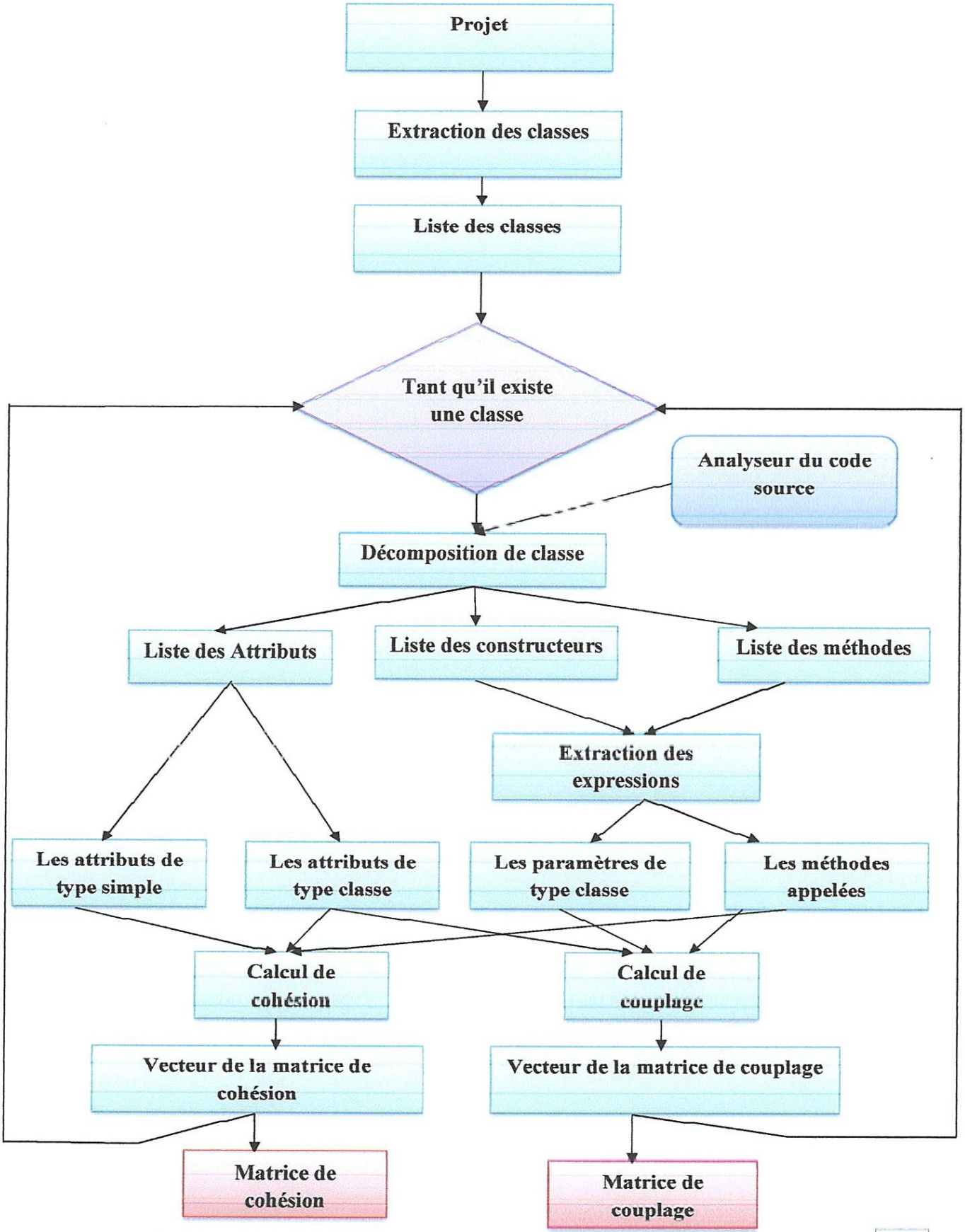


Figure 4.6 : *Cycle de calcul de couplage et cohésion d'une application OO.*

3.4 La spécification de la fonction objectif:

La spécification de la fonction objectif c'est l'étape la plus importante dans notre travaille, elle consiste à proposer une formule globale qui permet de combiné toutes les caractéristiques d'un service.

Un service doit être :

Autonome.

Sans état.

Découvrable.

Composable.

Abstrait.

Faiblement couplé et faiblement cohésif avec l'extérieure (avec les autres services).

Asynchrone.

Communiquer par des messages.

Avec un contrat bien précis, et acccssible par une interface.

Parmi ces caractéristiques ils y'a celles qui ne sont pas mesurable (concerne la plateforme de SOA) et celle qui sont mesurable (voir le tableau ci-dessus).

Les caractéristiques	Mesurable	Non mesurable (plateforme de SOA).
Autonomie	✓	
Sans état		✓
Découvrabilité		✓
Abstraction		✓
Couplage	✓	
Cohésion	✓	
Asynchronisation		✓
Communication par messages		✓
Interfaçage et accessibilité	✓	
Réutilisabilité	✓	

Tableau 4.2: *Classification des caractéristiques des services.*

Le tableau suivant montre comment évaluer les caractéristiques mesurables:

Les caractéristiques mesurables	Evaluation des caractéristiques
L'autonomie	Couplage inter service ↓
La réutilisabilité	Couplage inter service ↓
Le couplage	Couplage au sein de service ↑ Couplage entre les services ↓
La cohésion	Cohésion au sein de service ↑ Cohésion entre les services ↓
L'interface	Le nombre des fonctionnalités public n'étant pas appelé à l'intérieur du service.

Tableau 4.3: *L'évaluation des caractéristiques d'un service.*

3.4.1 L'Adaptation du calcul de couplage pour la fonction objectif:

La formule précédente de couplage ne permet pas de déterminer le dégrée de dépendance d'une classe envers son environnement c'est pour cette raison que nous avons calculé le couplage par rapport au nombre totale des relations d'une classe sans compter les relations d'héritage (voir la formule ci-dessous).

L'adaptation de la formule permet une mise à l'échelle des valeurs du couplage afin qu'ils soient homogènes avec les valeurs de la cohésion, ainsi la valeur max des deux métriques sera bornée entre 0 et 1.

$$\text{Couplage}(A, B) = \frac{\text{Couplage}(A,B)}{\sum_{i=1}^{n} \text{Couplage}(A,Ai)}$$

Ai représente une classe.

L'algorithme suivant explique comment nous avons calculé exactement la nouvelle formule de couplage :

```
Fonction Couplage (M : tableau (1..n, 1..n) d'entier) : tableau de réel.
Var Somme : entier ; M1 : tableau (1..n, 1..n) de réel.
Début
Pour i allant de 1 à n faire
        Début
        Somme ← Σ M[i][.]; //La somme de chaque ligne de la matrice.
        Pour j allant de 1 à n faire
        M1[i][j] ← M[i][j]/Somme;
        Fin;
Couplage ←M1;
Fin;
```

Figure 4.7 : *L'algorithme de transformation de couplage.*

Remarque:

Nous n'avons appliqué cet algorithme que pour les classes qui ne possède pas une relation d'héritage, les classes ayant une relation d'héritage leur valeur est le max =1.

Donc la nouvelle formule de couplage est calculée de la manière suivante:

$$\sum_i \text{Couplage (A, Ai)} = \text{Couplage (A, B)} + \text{Couplage (A, D)}.$$

$$\Leftrightarrow \sum_i \text{Couplage (A, Ai)} = 5+1=6.$$

$$\text{Couplage (A, B)}=5/6.$$

$$\text{Couplage (A, D)}=1/6.$$

Comme la classe A est une sous classe de la classe C alors le couplage est le maximum en représentant par 1.

$$\text{Couplage (A, C)}=1.$$

Le résultat de la nouvelle formule donnée par le tableau suivant :

Les classes	A	B	C	D
A	0	5/6	1	1/6
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0

Tableau 4.4: *Tableau de pourcentage de couplage.*

3.4.2 Calcul de la fonction objectif:

Après cette étude nous avons réunie toutes ces caractéristiques dans une formule globale appelée la fonction objectif F qui permet de spécifier les services d'une application :

$$F\,(S) = NB\_FCT\,(S) + Cohésion(S) + Couplage(S).$$

NB_FCT: Le nombre des fonctionnalités public n'étant pas appelé à l'intérieur du service (représente l'interface de service)/ nombre total des méthodes.

Cohésion(S) : La cohésion intra service (entre les classes du service).

Couplage(S) : Le couplage intra service.

La formule finale de la fonction objectif est la suivante :

$$F(A,B) = \frac{NB\_FCT\,(A,B) + (1 - Cohésion\,(A,B)) + (1 - MoyenneCouplage(A,B))}{3}$$

A, B : deux classes.

Les valeurs de la fonction objectif entre 0 et 1.

Le résultat de la fonction objectif représente le degré de similarité entre deux classes.

3.4.3 Génération de la Matrice des similarités :

L'application de cette formule sur les classes deux a deux génère une matrice symétrique où la plus petite valeur signifie que le degré de similarité entre les classes est le plus élevé.

3.5 L'identification des services:

Cette phase c'est la phase la plus importante dans notre travail, elle porte sur l'adaptation des trois algorithmes hiérarchique, Soustractive Clustering et IsoData afin d'identifier les services candidats d'une application.

3.5.1 L'adaptation de l'algorithme hiérarchique:

On a choisi la méthode hiérarchique ascendante avec une agrégation selon le lien moyen (la façon par laquelle les clusters sont fusionnés), puisque l'agrégation moyenne donne une bonne représentation des classes similaires et facilite la distinction des classes homogènes dans le dendrogramme (voir les figures 4.8, 4.9) :

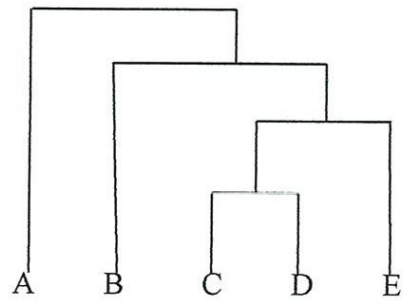


Figure 4.8 : *Agrégation selon le lien minimale.*

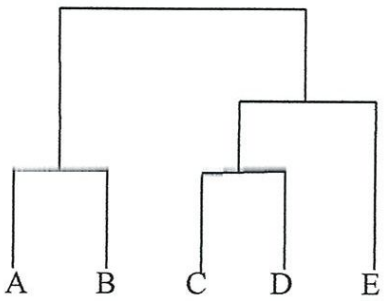


Figure 4.9 : *Agrégation selon le lien moyenne.*

nous avons adapté cet algorithme de la façon suivante

1. Démarrer en considérant chaque classe comme un cluster.
2. Calculer la matrice des distances :
   Dans notre cas la matrice des distances c'est une matrice symétrique résultant de l'application de la fonction objectif sur les paires des classes.
3. Chercher les deux clusters dont la valeur de la fonction objectif soit le minimum.
4. Fusionner les deux clusters les plus proches dans un seul nouveau cluster.
5. Mettre à jour la matrice de la fonction objectif en supprimant la ligne et la colonne des deux clusters les plus proches et les remplacer par leur moyenne.
6. Réitérer les étapes (3-5) jusqu'à ce que tous les clusters soient regroupés.

Figure 4.10 : *L'adaptation de l'algorithme hiérarchique.*

3.5.1.1 Formation des services dans l'algorithme hiérarchique:

L'algorithme hiérarchique délivre comme résultat un dendrogramme qui représente les dégrées de proximité et de similarité entre les classes sans proposer des groupe distincts, nous avons dû proposer notre propre algorithme de sélection de groupe constituant des services candidats.

L'algorithme de sélection consiste à un parcours infixé en profondeur du dendrogramme qui descend jusqu'à les feuilles puis extrait le premier fils gauche non visité puis il compare la valeur de leur père avec la valeur de R*P (R : La valeur de racine du dendrogramme, P un pourcentage). Si cette valeur est inférieur à R*P alors on découpe le nœud père en deux (fils gauche, fils droit) chacun est représenté comme un service et il marque ces deux fils comme des nœud visité sinon il monte vers le nœud père du nœud courant mais cette fois il compare la différence D entre l'ancien nœud père et le nouveau nœud père, si elle est inférieur à R*P il monte vers le haut sinon le nœud en question est identifié comme un service, il sera marqué comme un nœud visité et l'algorithme traite ensuite le nœud suivant.

L'algorithme reçoit comme paramètre un dendrogramme et donne comme résultat la liste des services (voir la figure suivante) :

1. Parcours infixé en profondeur du dendrogramme.
2. Comparer la valeur du père de la première feuille gauche non visité S(P) avec R*P tel que :
   - P : C'est le pourcentage par rapport à la valeur de la racine du dendrogramme (seuil pour couper le dendrogramme), ce pourcentage est soit choisi par l'utilisateur soit choisi comme valeur par défaut (un pourcentage qui représente la meilleure façon de couper n'importe quel dendrogramme).
   - R : c'est la valeur du nœud racine du dendrogramme.

   2.1 Si (S(P) < R*P) on découpe P en deux (fils gauche, fils droit) et on détermine chacun comme un service.

   2.2 Sinon on détermine le père de P.
      - Calculer D= S(Père(P))-S(P).
      - Si (D> R*P) alors le service est déterminé à partir de ce nœud.
      - Sinon aller à l'étape 2.1.
3. Mettre à jour les valeurs du dendrogramme et marquer les nœuds services comme des nœuds visités.
4. Itérer les étapes (1-3) jusqu'à ce que tous les nœuds du dendrogramme soient visités.

Figure 4.11 : *La sélection des services dans l'algorithme hiérarchique.*

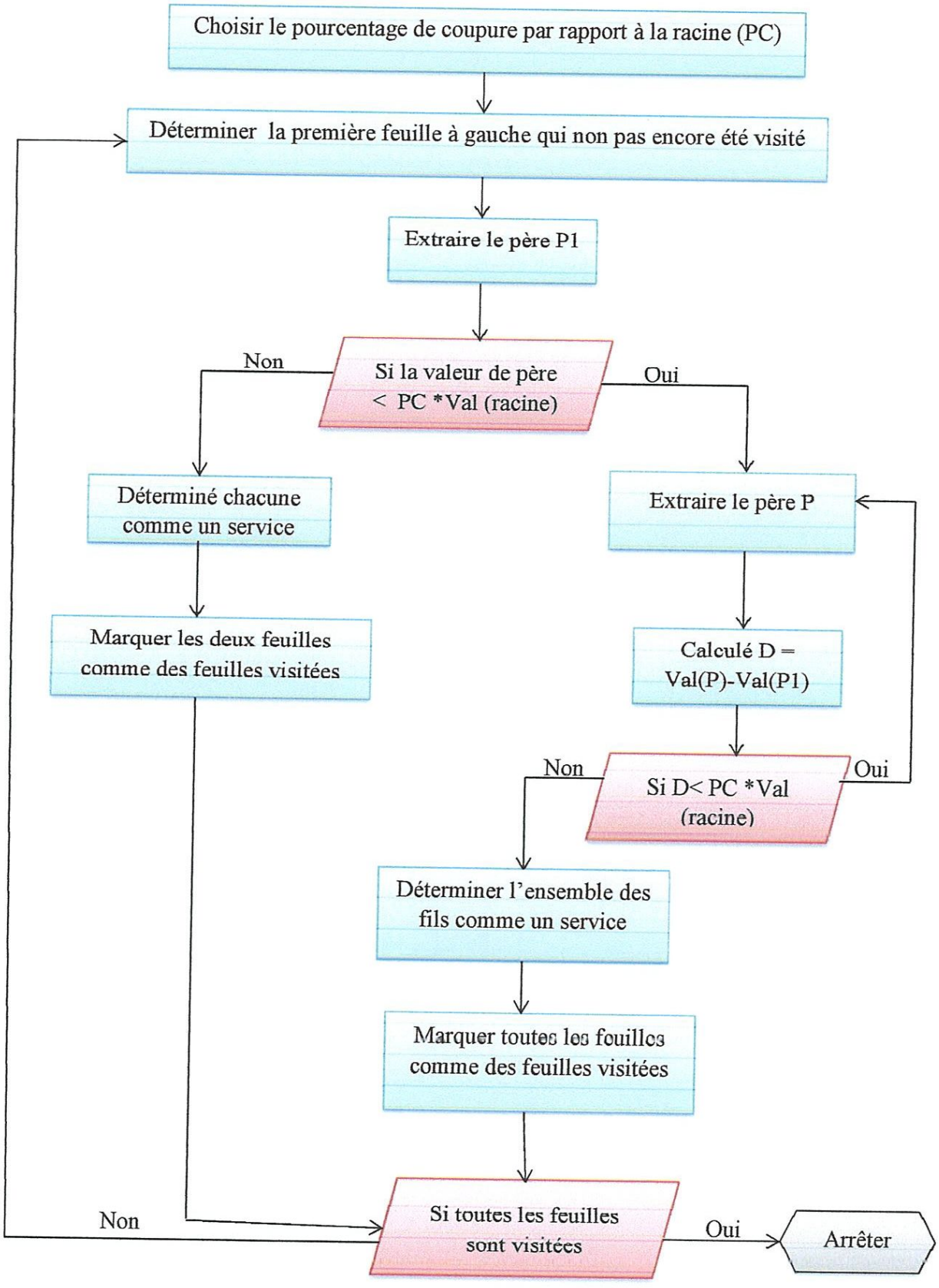


Figure 4.12 : *Diagramme du processus d'identification de service de l'algorithme hiérarchique*

3.5.2 L'adaptation de l'algorithme Soustractive Clustering:

Nous l'avons adapté de la manière suivante :

Etape 1 : choisir la classe avec le plus grand potentiel comme centre du premier groupe.

La classe qui représente le centre potentielle peut être une des options suivante:

- La classe dont la valeur de la fonction objectif est la plus petite.
- La classe dont le nombre des fonctionnalités est le plus grand.
- La classe dont la moyenne des distances (les valeurs de la fonction objectif) avec les autre classe est la plus petite.
- Ou bien choisir une classe aléatoirement selon l'ordre des classes dans le projet.

Etape 2 : évincer toutes les classes dans le voisinage du premier centre de l'amas (déterminé par le rayon).

Le rayon est choisi suivant l'une des méthodes suivante:

- Manuellement : une valeur entre [0-1].
- Ou choisir d'une manière automatique :

$$R = \frac{\text{la valeur min de la fonction objectif} + \text{la valeur max de la fonction objectif}}{2}$$

Avec un pas d'arrondissement de 0,1.

Etape 3 : Itérer le processus jusqu'à ce que toutes les classes soient dans un rayon du centre d'un amas.

Figure 4.13 : *L'adaptation de l'algorithme Soustractive Clustering.*

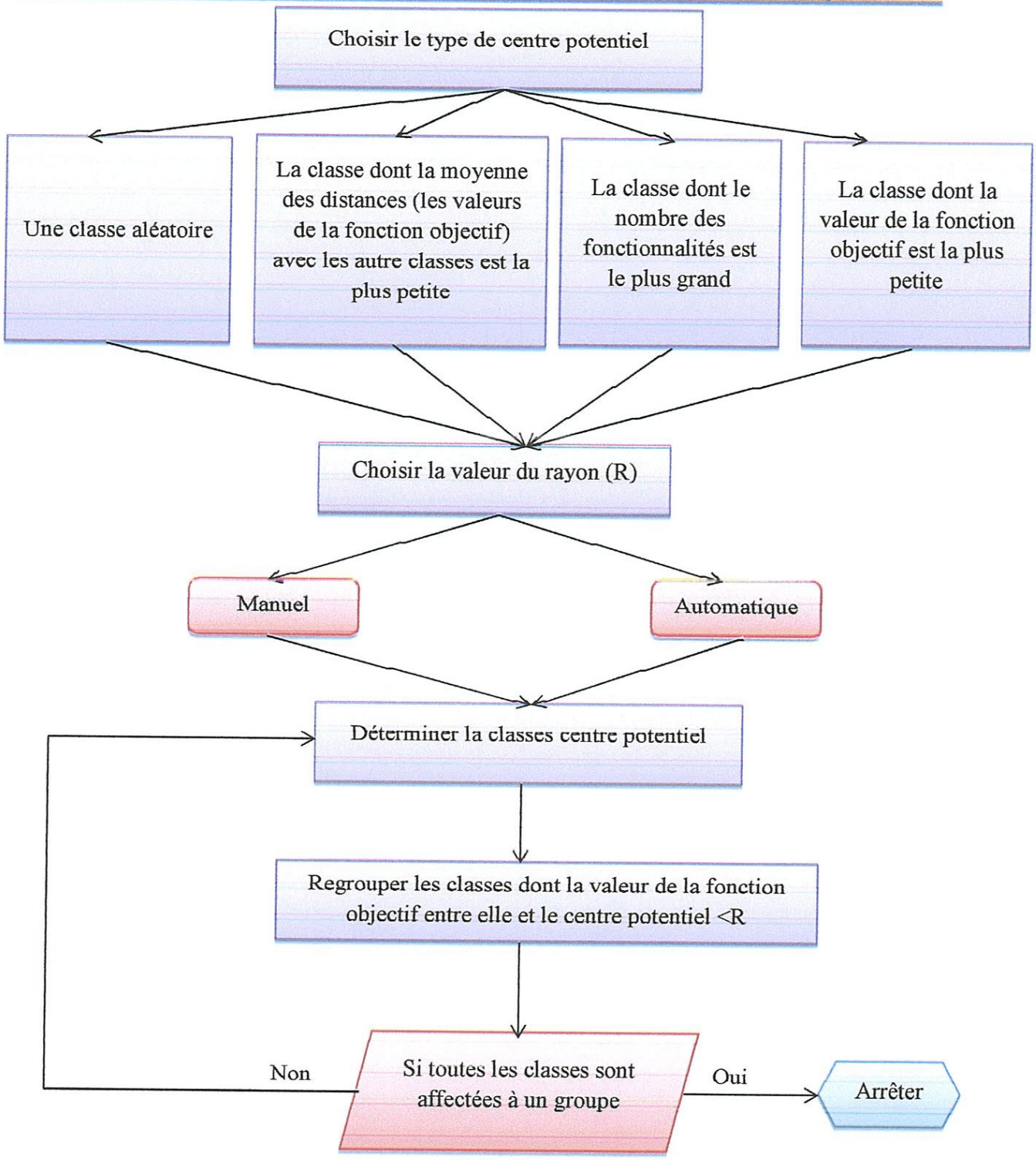


Figure 4.14 : *Diagramme du déroulement de l'algorithme Soustractive Clustering adapté.*

3.5.2.1 Discussion des choix d'adaptation:

Dans cet algorithme nous avons effectué les adaptations suivantes:

Comme notre matrice représente la similarité entre deux classes, le centre potentiel doit être une classe et non pas une valeur (distance) de la matrice de similarité. La méthode du choix du centre potentiel n'étant pas imposée, nous avons préféré de donner le choix à l'utilisateur d'utiliser une des options suivantes:

- La classe qui contient la plus petite valeur de similarité avec une autre classe. Le choix de cette option permet de regrouper au moins deux classes où la distance <= rayon.
- La classe dont le nombre des fonctionnalités est le plus grand signifie que cette classe a une grand espace d'interfaçage (possède plusieurs relations avec les autres classes) donc elle est plus disposé à jouer le rôle de centre potentiel.
- La classe dont la moyenne de similarité avec les autres classes soit la plus petite (plus proche des autres classes), donc il y a une grande probabilité que cette classe regroupe plus d'éléments.

En ce qui concerne le choix du rayon, et puisque la matrice de similarité change avec le changement des applications, nous avons choisi de représenter le rayon automatique par la moyenne approximé entre le minimum et le maximum de ce dernier et afin d'arriver à un regroupement un peux plus judicieux.

3.5.3 L'adaptation de l'algorithme IsoData:

Afin d'utiliser IsoData, nous avons dû fixer les paramètres de la façon suivante :

I_{max} : c'est le nombre total des classes.

Le seuil S est fixé :

- Entre [0-1] (manuellement).
- Soit automatique :

$$S = \frac{\text{la valeur min de la fonction objectif} + \text{la valeur max de la fonction objectif}}{2}$$

1. Appliquer l'algorithme de k-means :

    1.1 Les k centre initiaux sont fixés suivant une des options suivantes:

    - ✓ La première classe de l'application (min).
    - ✓ Toutes les classes d'une application (max).
    - ✓ Ou bien choisir une valeur entre 1 et le max des classes d'une application (manuel).

    1.2 Attribuer chaque classe au centre le plus proche (la valeur de la fonction objectif entre cette classe et le centre soit la plus petite).

    1.3 Calculer les nouveaux centres :

    La classe centre= Min (moyenne distance (Aj, Ai)).

    1.4 Itérer le processus jusqu'à ce que les classes centre soient les mêmes.

2. Si la distance entre les membres du cluster et le centre est supérieure au seuil S, alors éclater le cluster en deux, puis calculer le nouveau centre du nouveau cluster (1.3).
3. Si la distance entre deux clusters (entre les centres des clusters) est inférieur au seuil S fusionner les deux clusters.
4. Aller à l'étape 1 jusqu'à atteindre I_{max} ou les membres des groupes ne change plus.

Figure 4.15: *L'adaptation de l'algorithme IsoData.*

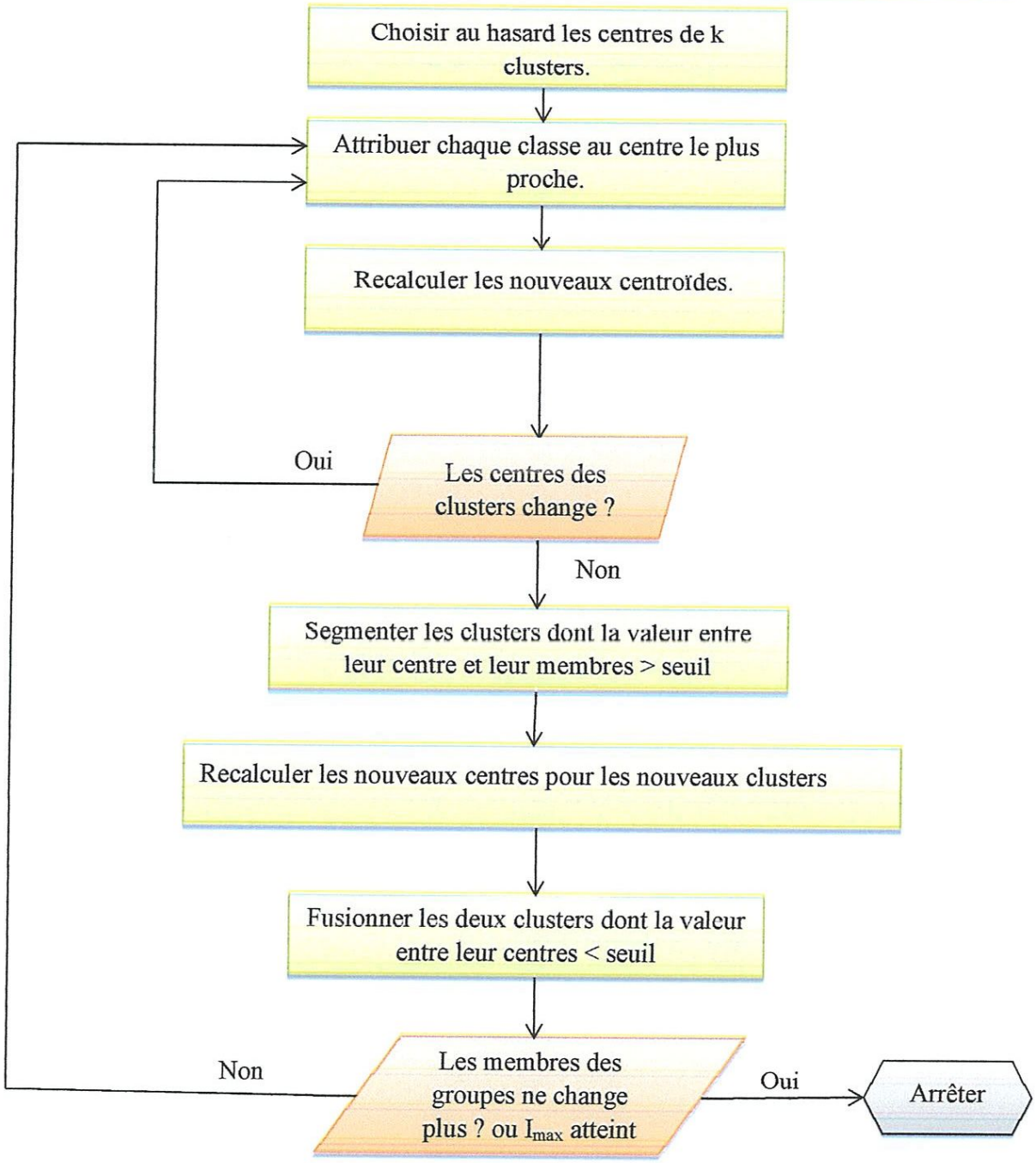


Figure 4.16 : *Diagramme du déroulement de l'algorithme IsoData adapté.*

3.5.3.1 Explication des choix d'adaptation:

Puisque le nombre des services résultant ne dépasse pas le nombre totale des classes du projet nous avons limité le choix du nombre des centres initiaux ente 1 (min) et le nombre total des classes du projet (max), et comme les valeurs dans la matrice de similarité reflètent une

relation entre deux classes, nous devons choisir un centroïde parmi les classes existantes et en aucun cas calculer une nouvelle valeur pour centroïde.

Afin de choisir le nouveau centroïde pour chaque nouveau cluster nous avons proposé un algorithme décrit dans figure 4.17.

Il consiste à calculer la moyenne des similarités de chaque membre avec le reste des membres du même cluster, puis nous choisissons la classe qui aura la plus petite moyenne.

```
Fonction Nouveau_Centre( M : tableau (1..n) (1..n) de réel) : chaine de caractère;
Var S: réel; Nb : entier; T : tableau (1..n) de réel;
Début
S←0;
Nb←0;
        Pour i allant de 1 a n faire
        Début
                Pour j allant de 1 a n faire
                Début
                        Si (i ≠ j) alors
                                Début
                                S←S+M[i][j];
                                Nb←Nb+1,
                                Fin;
                Fin;
                T[i]←S/Nb;
                S←0;
                Nb←0;
        Fin;
Nouveau_Centre←Min(T); // renvoie le nom de la classe dont la moyenne est la plus petite.
Fin;
```

Figure 4.17 : *Algorithme de sélection des nouveaux centroides.*

En ce qui concerne le seuil nous l'avons calculé automatiquement de la même façon que le rayon de l'algorithme Soustractive Clustering en prenant la moyenne entre la valeur min et la valeur max de la matrice de similarité. Ainsi le seuil va changer suivant les valeurs de la matrice de similarité.

4 Conclusion:

La phase de conception représente la troisième phase de développement du logiciel, Cette phase consiste à présenter la démarche à suivre qui dépend du problème à traiter.

Dans ce chapitre nous avons détaillé les le contexte, les objectifs et les différentes étapes de notre projet à partir de l'analyse du code source, puis la décomposition de l'application jusqu'à l'identification des services candidats.

Nous avons dû faire une correspondance entre les deux paradigmes objet et service puis extraire les caractéristiques d'un service de qualité, que nous avons utilisé pour former des groupes de classe qui peuvent être considérés comme des services candidats.

L'identification des services étant modélisé sous forme de problème de regroupement de classe, nous avons choisi d'adapter des algorithmes de classification non supervisée (hiérarchique, Soustractive Clustering et IsoData) puisque le nombre de services résultant ne peut être connu à l'avance.

Le choix de plusieurs algorithmes de classification est justifié par la volonté de consolider les résultats d'identification et de faire une étude comparative afin d'aider l'utilisateur final à avoir une idée sur le comportement de chaque algorithme.

Les résultats de l'étude comparative seront détaillés dans le chapitre suivant.

Chapitre 5 :

Implémentation

1 Introduction :

Dans ce dernier chapitre et après l'aperçu théorique des chapitres précédents, nous allons présenter le côté pratique de notre application. Notre but est d'automatiser l'identification des services à partir d'une application orientée objet et plus particulièrement Java.

Ce chapitre présente tout d'abord les différents outils de développement que nous avons utilisé soit pour implémenter notre application comme le langage de programmation JAVA soit pour confirmer quelques résultat exemple STAN qui vérifie le résultat de notre couplage, ensuite nous allons présenter les interfaces et les fonctionnalités de notre logiciel afin de faciliter son utilisation.

Enfin, nous présenterons un exemple à travers lequel nous comparerons les résultats des différents algorithmes.

2 Les outils de développement :

2.1 Java :

Nous avons choisi le langage Java qui est un bon représentant des langages orientées objet assez puissant et fiable avec l'éditeur eclipse, qui offre des interfaces faciles à manipuler

2.2 STAN :

Stan est un outil d'analyse des structures java, il rassemble le développement est l'assurancc de qualité des logiciels. Nous l'avons utilisé au début du projet pour comparer les résultats des calculs du couplage. Et il s'est avéré que nos résultats étaient assez proches.

Stan permet au développeur et les chefs des projets de visualiser leur conception, comprendre le code et mesuré la qualité des logiciel ainsi de rapporter les défauts de conception.

Stan permet de calculer un ensemble des métriques en présentant leur résultat sous forme graphique, il suffit de télécharger gratuitement le logiciel à partir de site (http://stan4j.com) soit en télécharger une application standard pour Windows ou bien une extension pour eclipse [w35].

L'interface de Stan est similaire à celle d'Eclipse (voir figure 5.1) :

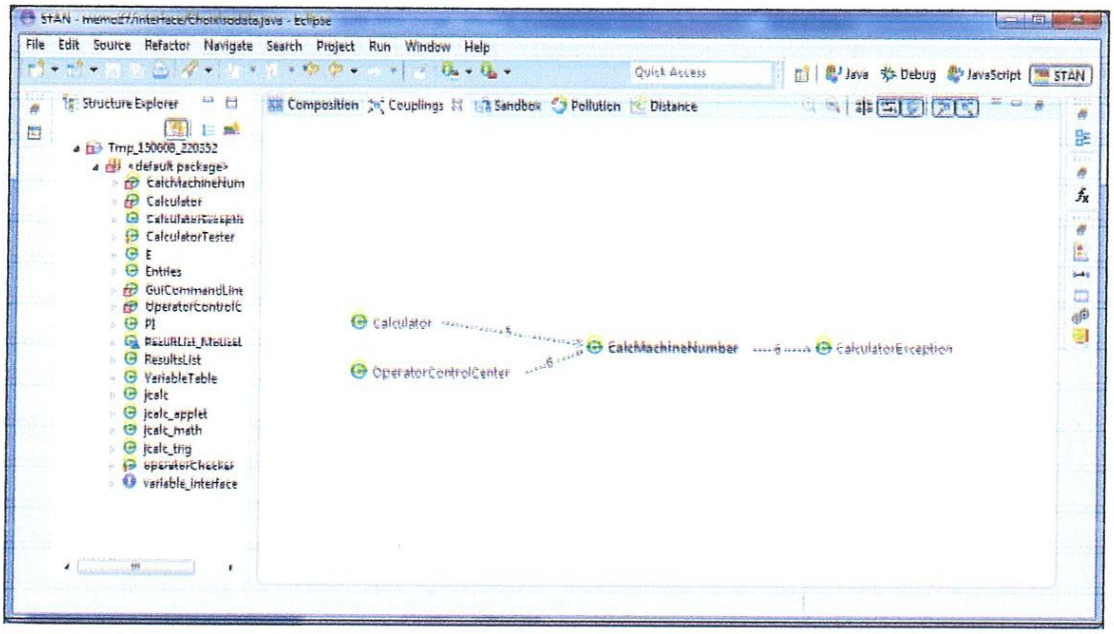


Figure 5.1 : *L'interface de STAN.*

2.3 ASTParser :

ASTParser est un analyseur syntaxique qui convertit un fichier source java à un arbre syntaxique abstrait AST.

L'analyseur syntaxique (ASTParser) permet d'analyser et de découper les différents éléments du langage Java. Il permet ainsi d'extraire les différentes composantes d'une classe donnée (le nom des classes, les importations, les méthodes, les appels, les variables, etc).

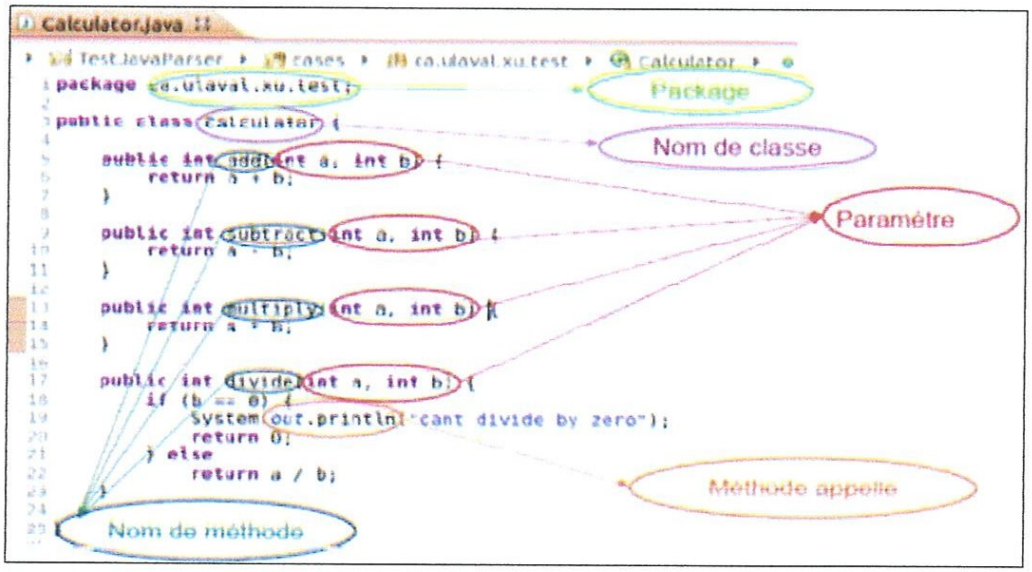


Figure 5.2 : *Les composantes d'une classe java [w36].*

ASTParser est disponible sous forme d'une bibliothèque (JAR) (sur le site https://code.google.com/p/javaparser/) utilisable dans beaucoup d'environnements. Cette bibliothèque est un projet écrite entièrement en Java et ne requiert aucun environnement particulier. Il est indépendant de l'IDE ou de toute autre technologie, il est simple à utiliser et de petite taille [w36].

Voici un exemple de quelques éléments en java et les éléments qui correspondant en ASTParser:

L'élément en java.	L'élément correspond en ASTPareser.
Package	PackageDeclaration
Import	ImportDeclaration
JavaClass	ClassOrInterfaceDeclaration
Method	MethodDeclaration,ConstructorDeclaration
Field	FieldDeclaration
MethodCall	MethodCallExpr
Parameter	Parameter
Variable	VariableDeclaratorId

***Tableau 5.1** : Quelques éléments en java et leur correspondance en ASTParser [w36].*

3 L'architecture globale de notre application:

Au départ l'utilisateur charge une application orientée objet à partir de l'interface, cette dernière est analysée par l'analyseur ASTParser. Les informations des différentes parties de l'application seront extraites puis envoyées au package de couplage et de cohésion pour calculer ces deux métriques. Pendant le calcul de la cohésion, un calcul parallèle est effectué pour récupérer le nombre des fonctionnalités, ensuite nous appliquons la fonction objectif afin de générer la matrice de similarité qui constitue l'entrée du package d'identification de service. Le package d'identification permet d'appliquer les algorithmes de classification pour proposer à la fin à l'utilisateur des services candidats.

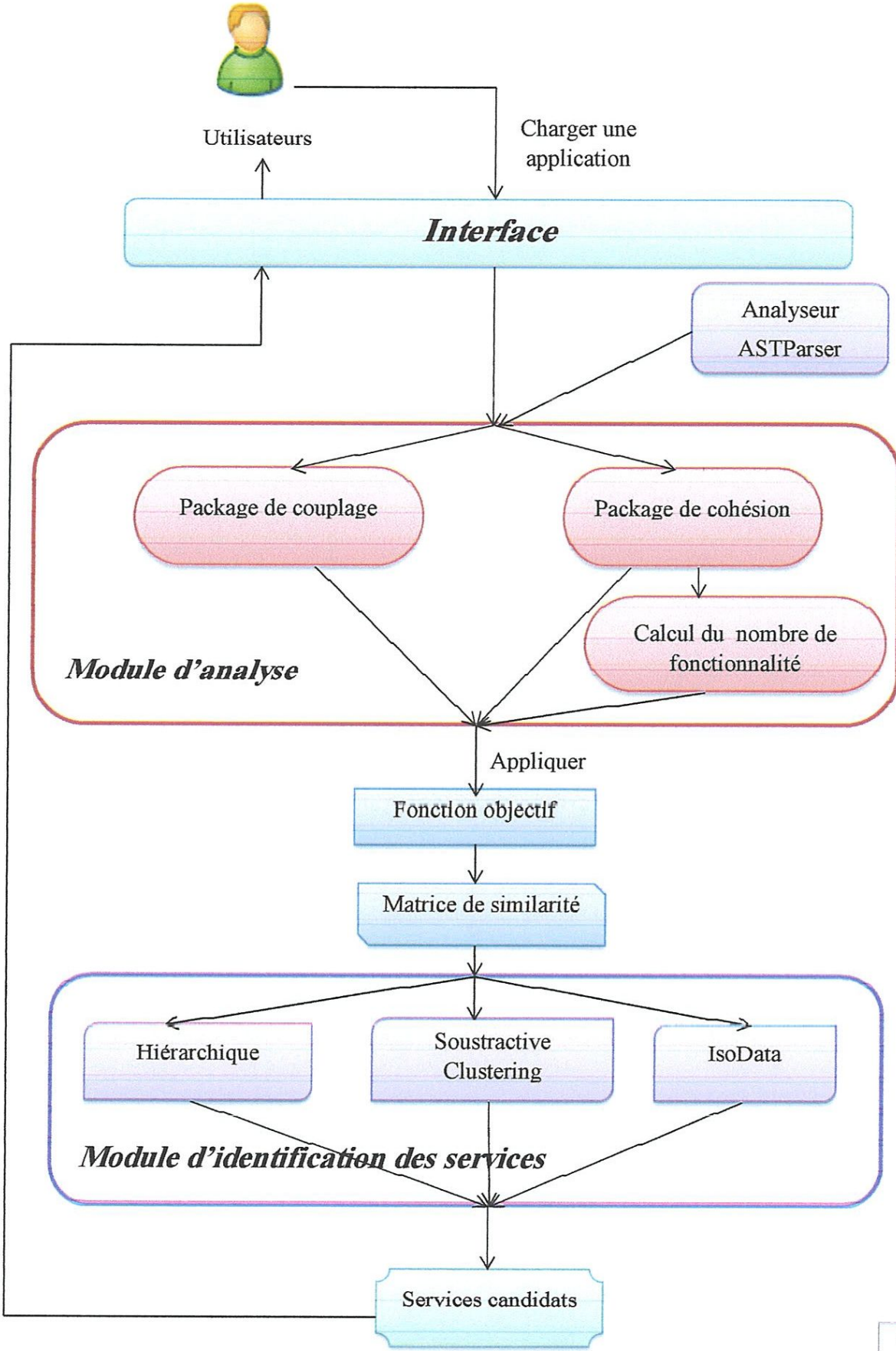


Figure 5.3 : *Architecture global de notre application.*

4 L'interface de l'application:

Nous avons divisé notre interface en deux parties: une contenant quelques options d'accueil et l'autre réservée à l'identification des services (voir figure 5.4) :

Figure 5.4 : *L'interface de notre application.*

4.1 Les options d'accueil :

Contient six fonctionnalités:

- Charger un projet java :
- Récupérer une matrice de similarité à partir d'un fichier texte :
- Remplir une matrice de similarité :
- Le guide d'utilisation:
- Calculer le temps d'exécution :
- Sauvegarder une matrice :
- Restaurer les fenêtres précédentes
- Quitter l'application :

La partie suivante détaille chaque fonctionnalité.

4.1.1 Charger un projet java :

Une fois que nous cliquons sur ce bouton une fenêtre s'affiche permettant aux utilisateurs de choisir et d'importer un projet java, après le choix du projet une fenêtre qui indique le nombre total des classes apparait puis l'analyseur commencera à filtrer tous les fichiers source et de calculer le couplage, la cohésion, le nombre des fonctionnalités ainsi que le calcul de la matrice de similarité (figure 5.5). Puis toutes les informations seront affichées dans des fenêtres séparées.

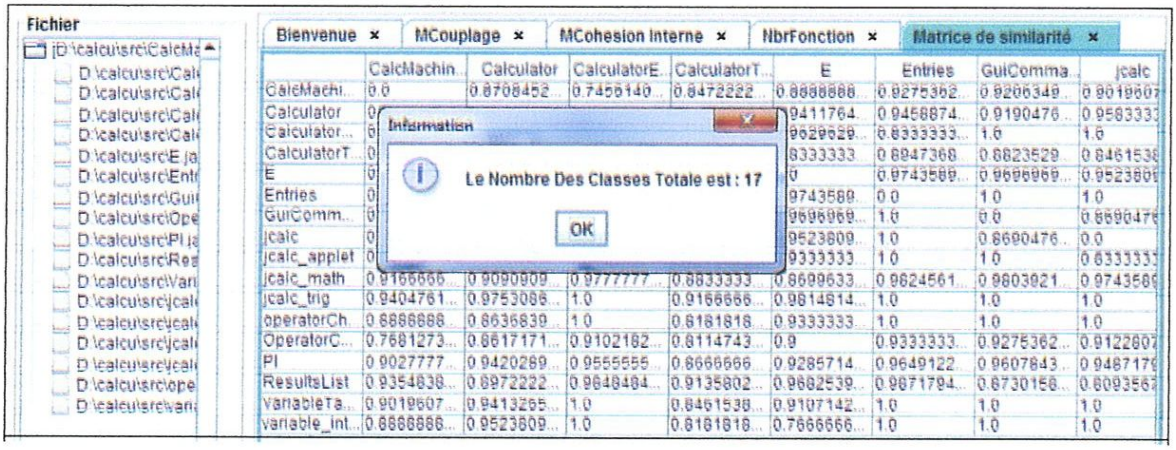
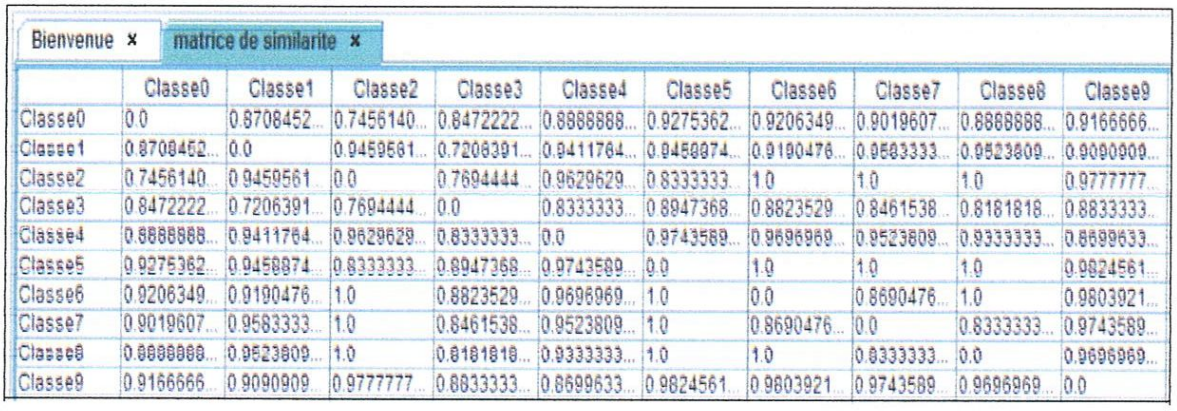


Figure 5.5 : *Fenêtre des résultats d'analyse d'un projet java.*

4.1.2 Récupérer une matrice de similarité à partir d'un fichier texte :

Cette option donne à l'utilisateur la possibilité de récupérer une matrice carré symétrique de similarité écrite dans un fichier texte (figure 5.6).



	Classe0	Classe1	Classe2	Classe3	Classe4	Classe5	Classe6	Classe7	Classe8	Classe9
Classe0	0.0	0.8708452...	0.7456140...	0.8472222...	0.8888888...	0.9275362...	0.9206349...	0.9019607...	0.8888888...	0.9166666...
Classe1	0.8708452...	0.0	0.9459561...	0.7206391...	0.9411764...	0.9458874...	0.9190476...	0.9583333...	0.9523809...	0.9090909...
Classe2	0.7456140...	0.9459561...	0.0	0.7694444...	0.9629629...	0.8333333...	1.0	1.0	1.0	0.9777777...
Classe3	0.8472222...	0.7206391...	0.7694444...	0.0	0.8333333...	0.8947368...	0.8823529...	0.8461538...	0.8181818...	0.8833333...
Classe4	0.8888888...	0.9411764...	0.9629629...	0.8333333...	0.0	0.9743589...	0.9696969...	0.9523809...	0.9333333...	0.8699633...
Classe5	0.9275362...	0.9458874...	0.8333333...	0.8947368...	0.9743589...	0.0	1.0	1.0	1.0	0.9824561...
Classe6	0.9206349...	0.9190476...	1.0	0.8823529...	0.9696969...	1.0	0.0	0.8690476...	1.0	0.9803921...
Classe7	0.9019607...	0.9583333...	1.0	0.8461538...	0.9523809...	1.0	0.8690476...	0.0	0.8333333...	0.9743589...
Classe8	0.8888888...	0.9523809...	1.0	0.8181818...	0.9333333...	1.0	1.0	0.8333333...	0.0	0.9696969...
Classe9	0.9166666...	0.9090909...	0.9777777...	0.8833333...	0.8699633...	0.9824561...	0.9803921...	0.9743589...	0.9696969...	0.0

Figure 5.6 : *Récupération d'une matrice de similarité à partir d'un fichier texte.*

4.1.3 Remplir une matrice de similarité:

L'utilisateur fixera la dimension de la matrice, une fois que la dimension est fixée une matrice carrée s'affiche (figure 5.7).

La matrice remplie doit être une matrice symétrique.

Figure 5.7 : *Matrice carrée de taille 4*4.*

4.1.4 Le guide d'utilisation:

Permet d'identifier dans quelques lignes notre projet globalement, ainsi il aide l'utilisateur à comprendre comment fonctionne notre application (figure 5.8).

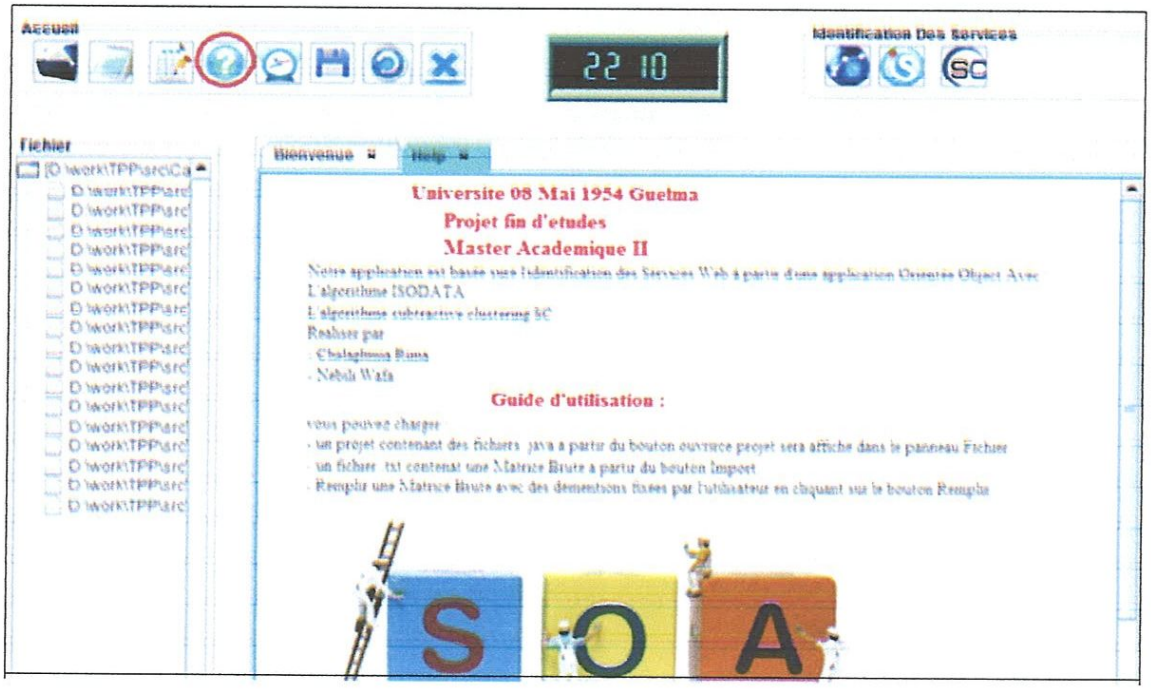


Figure 5.8 : *Le guide d'utilisation.*

4.1.5 Le calcul du temps d'exécution:

Après l'identification des services nous pouvons cliquer sur cette option pour voir le temps écoulé pendant l'exécution en nanoseconde (figure 5.9).

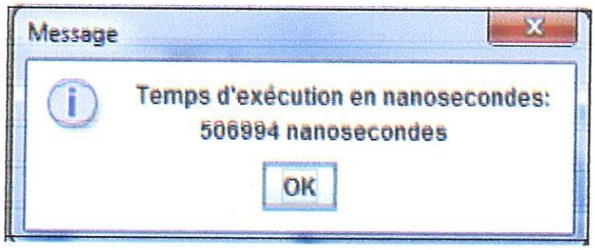


Figure 5.9 : *Le temps d'exécution.*

4.1.6 Sauvegarder une matrice:

C'est une opération qui aide les utilisateurs à sauvegarder une matrice de similarité sous forme de texte dans n'importe quel endroit choisi par ce dernier, lorsque la matrice est sauvegardée une boite de dialogue sera affichée (figure 5.10).

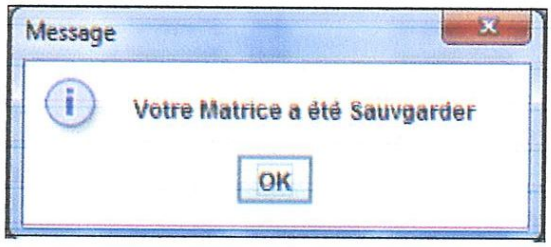


Figure 5.10 : *Boite de confirmation de sauvegarde d'une matrice.*

4.1.7 Restaurer les fenêtres précédentes:

Ce bouton restaure les fenêtres quittées par l'utilisateur.

4.1.8 Quitter l'application:

Cette option permet d'abandonner l'application en cliquant sur ok pour confirmer la sortie.

4.2 Les algorithmes d'identification des services:

Après l'extraction de la matrice de similarité nous pouvons choisir l'un des algorithmes suivant pour identifier les services :

- L'algorithme hiérarchique 

- IsoData : 
- Soustractive Clustering : 

Chaque algorithme fonctionne sur la matrice de similarité de la manière suivante :

- Dans l'algorithme hiérarchique nous devons choisir un pourcentage de coupure par rapport à la valeur de la racine du dendrogramme (manuellement ou une valeur par défaut) figure 5.11.

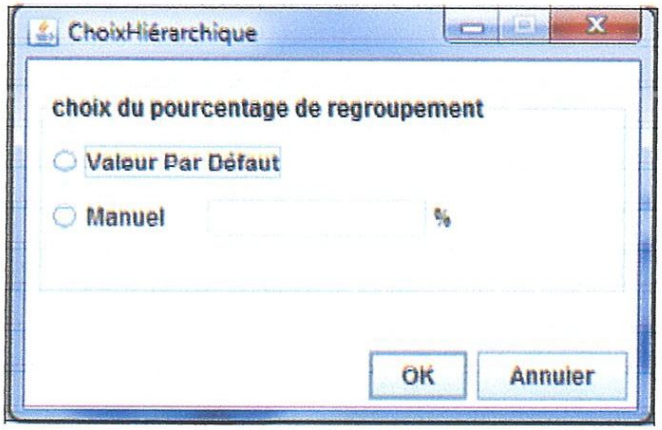


Figure 5.11 : *Boite du choix du pourcentage de regroupement.*

- La méthode Soustractive Clustering demande le choix du centre potentiel (figure 5.12), ainsi que la valeur du rayon :

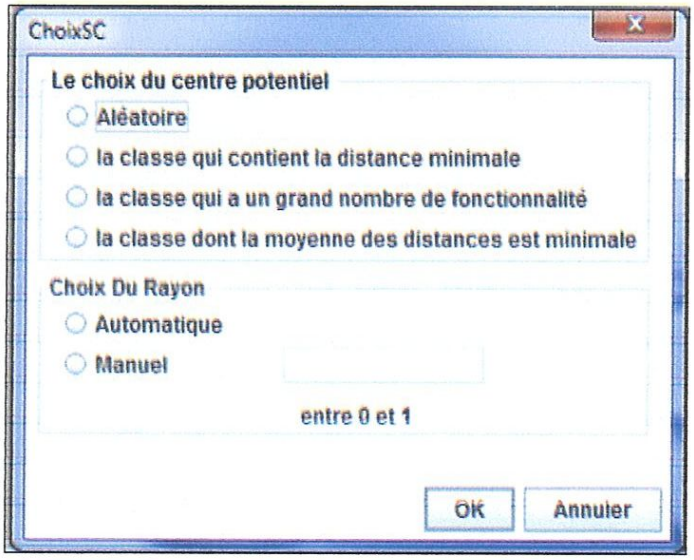


Figure 5.12 : *Le choix du centre potentiel et du rayon pour Soustractive Clustering.*

- l'algorithme IsoData nous impose de fixer le nombre des centres initiaux, Et le seuil (figure 5.13):

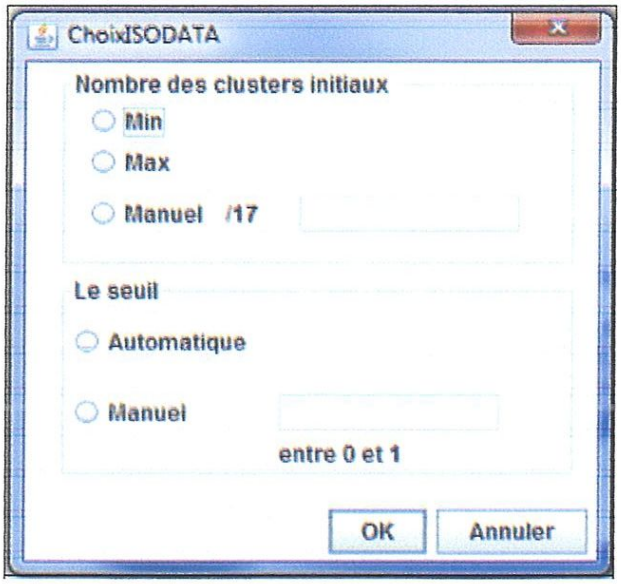


Figure 5.13 : *Choix du nombre des centres initiaux et le seuil pour IsoData.*

A la fin de chaque algorithme le nombre des services s'affiche dans une boite de dialogue (figure 5.14).

Une autre boite s'affiche pour choisir le type d'affichage du résultat (figure 5.15) soit sous forme d'un tableaux (figure 5.16) ou bien sous forme graphique (figure 5.17), quel que soit le type choisi le résultat apparait dans une fenêtre intitulée par le nom de l'algorithme choisi.

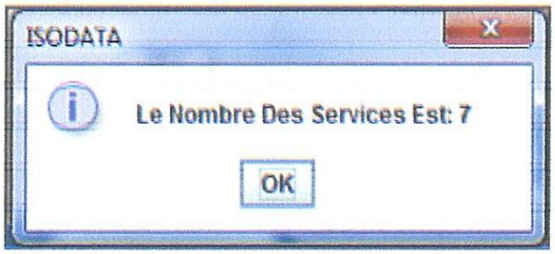


Figure 5.14: *Le nombre des services.*

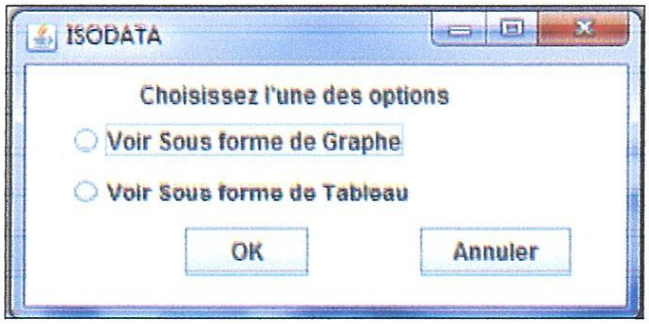


Figure 5.15 : *Le choix du type d'affichage pour les résultats obtenus.*

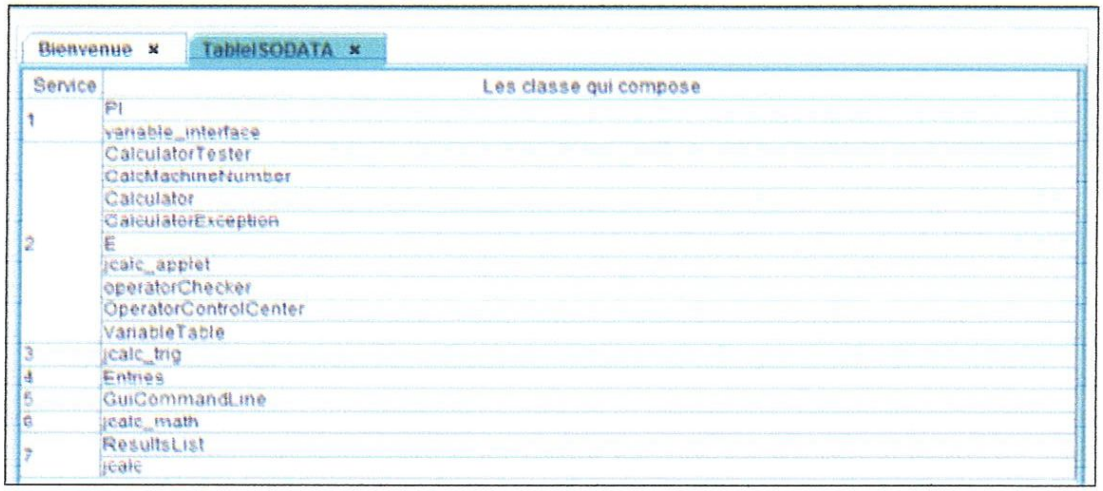


Figure 5.16 : *Le résultat de L'algorithme IsoData sous forme d'un tableau.*

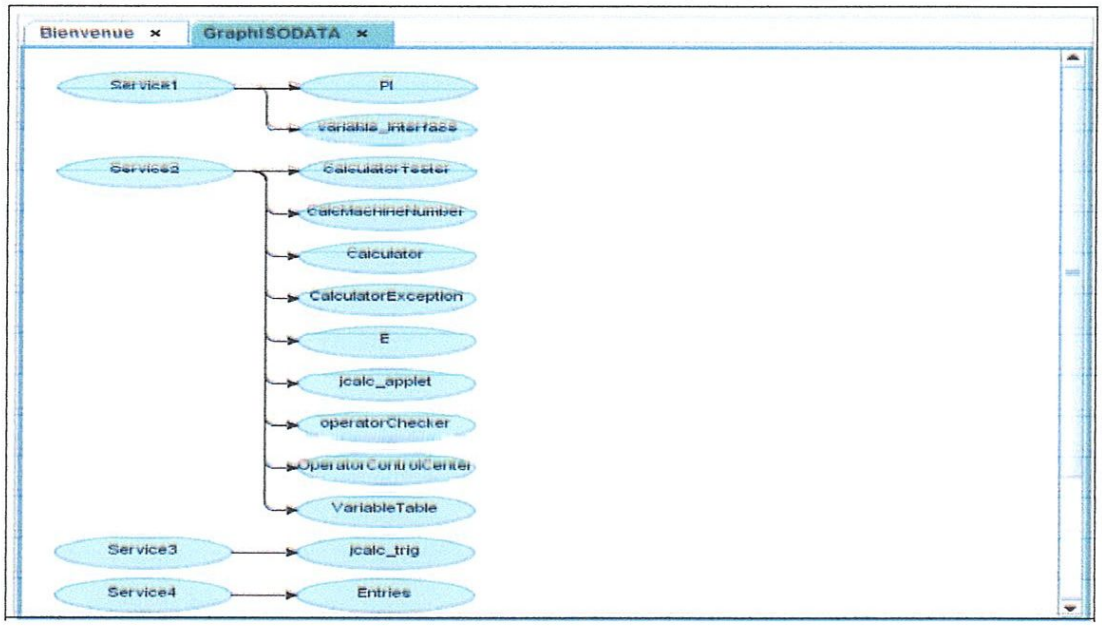


Figure 5.17 : *Le résultat de L'algorithme IsoData sous forme graphique.*

5 Expérimentation:

Afin de valider nos résultats nous avons choisi la démarche suivante :

1- chercher dans les travaux similaires qui ont essayé de faire migrer des applications orientées objet vers la SOA. Nous avons prix 2 exemples de tailles différentes afin de voir le comportement des algorithmes pour les différents cas. Puis nous avons comparé nos résultats avec les résultats des travaux similaires

2- Un deuxième type de d'expérimentation a été effectué, avec l'objectif d'améliorer le paramétrage de chaque algorithme, donc nous avons testé les différents algorithmes avec différentes combinaisons de paramètre afin de trouver le meilleur paramétrage.

3- Enfin nous avons essayé de comparer les différents algorithmes à travers deux critères, la qualité des résultats et le temps d'exécution.

5.1 Présentation des applications de test:

Nous avons choisi deux applications orientées objet comme exemple de test que nous connaissons à l'avance leur services résultant:

Exemple 1: ***Java Calculator Suite*** c'est une collection des classes java (17 classes) décrit une calculatrice. Il effectue des opérations mathématiques de base, dispose d'une interface graphique et offrir 25 opération différentes, cet application est disponible sur le net sur [w37].

Exemple 2: ***Calcul_entier*** est une application simple qui contient 12 classes java, elle est composée de trois modules spécialisés dans le calcul sur deux entier (la somme, le produit et la soustraction). Chaque module est accessible à partir de son propre interface et utilise quelque fonctionnalités déclarées dans ces classes (fonction de lecture des entiers, fonction de calcul, et fonction d'affichage des résultats).

Le résultat attendu de chaque exemple dans le tableau suivant :

L'exemple	Les résultats attendus	
	Le numéro de service	Les classes qui composent
Java Calculator Suite	Service 1	CalcMachineNumber
	Service 2	OperatorControlCenter
	Service 3	Calculator
		CalculatorException
		CalculatorTester
		Jcalc
		Jcalc_applet

	Service 4	E
		Jjcalc_math
		Jcalc_trig
		Variable_interface
	Service 5	VariableTable
		OperatorChecker
	Service 6	PI
	Service 7	Entries
		GuiCommandLine
		ResultsList
Calcul_entier	Service 1	Add
		Add_affiche
		Add_fonct
		Add_param
	Service 2	Mul
		Mul_affiche
		Mul_fnct
		Mul_param
	Service 3	Sous
		Sous_affiche

		Sous_fnct
		Sous_param

Tableau 5.2 : *Les résultats attendus des deux exemples d'expérimentation.*

Remarque: pour l'exemple 1 les résultats obtenus sont récupérés de [31] afin de les comparer. Et pour l'exemple 2 chaque module compose d'un service, et chaque service présente une des fonctionnalités suivantes :

- La somme de deux entiers.
- Le produit de deux entiers.
- La soustraction de deux entiers.

Chaque service permet à l'utilisateur de saisir deux nombres entiers, puis le calcul enfin affiche le résultat à travers une interface.

Nous avons effectué une étude comparative entre notre algorithmes d'identification des services sur ces deux exemples afin d'extraire les avantages et les inconvénients de chacun.

5.2 Le choix des paramètres :

5.2.1 L'algorithme hiérarchique:

5.2.1.1 Le choix du pourcentage de coupure pour l'algorithme hiérarchique:

Afin de choisir le meilleur pourcentage de coupure nous avons testé cet algorithme pour quelques pourcentages afin de choisir le meilleur (voir tableau 5.3).

Le pourcentage par rapport au nœud racine	Le nombre des services obtenu	
	Exemple 1	Exemple 2
10 %	17 services	12 services
20%	17 services	12 services
30%	17 services	12 services
40%	17 services	12 services
50%	17 services	12 services

60%	17 services	12 services
70%	17 services	12 services
75%	5 services	2 services

Tableau 5.3: *Le résultat de coupure de l'algorithme hiérarchique pour des pourcentages différents.*

D'après les résultats nous avons constaté que le pourcentage 75 % donne un nombre de services le plus proche à celui des résultats attendus donc nous avons défini que 75% comme une valeur par défaut.

5.2.2 L'algorithme Soustractive Clustering :

5.2.2.1 Le choix du centre potentiel et le rayon de la méthode Soustractive Clustering:

L'application de cet algorithme pour ces deux applications donne le tableau suivant :

Le rayon / Les services résultants	Le centre potentiel							
	Aléatoire		La classe avec la similarité minimale.		La classe avec le plus grand nombre de fonctionnalité.		La classe avec moyenne de ces similarités est minimale.	
	Ex 1	Ex 2	Ex 1	Ex 2	Ex 1	Ex 2	Ex 1	Ex 2
0.05	17	12	17	12	17	12	17	12
0.1	17	12	17	12	17	12	17	12
0.2	17	12	17	12	17	12	17	12
0.3	17	12	17	12	17	12	17	12
0.4	17	12	17	12	17	12	17	12
0.5	17	12	17	12	17	12	17	12

Rayon automatique	6	3	6	3	8	5	7	3

Tableau 5.4 : *Le résultat de l'algorithme SC pour des différents rayons et différents centres potentiels.*

Rayon automatique = moyen entre similarité minimal similarité maximal limité dans un intervalle.

Nous remarquons que le choix du centre potentiel et le rayon ont une grande influence sur les résultats obtenues. La combinaison du rayon automatique avec le centre potentiel prix comme la classe avec la moyenne des similarités la plus petite donne un bon résultat et qui est le plus proche aux résultats attendus.

5.2.3 L'algorithme IsoData:

5.2.3.1 Le choix du nombre des clusters initiaux (K) et le seuil:

Le tableau suivant résume les résultats collectés suivant les différentes valeurs du nombre des clusters initiaux et du seuil pour les mêmes exemples précédents:

services résultant. / Seuil	Le nombre des clusters initiaux					
	Min =1.		Max = nombre totale des classes initiaux.		La moitié du nombre des classes totales initiaux.	
	Ex 1	Ex 2	Ex 1	Ex 2	Ex 1	Ex 2
0.05	17	12	17	12	17	12
0.1	17	12	17	12	17	12
0.2	17	12	17	12	17	12
0.3	17	12	17	12	17	12
0.4	17	12	17	12	17	12
0.5	17	12	17	12	17	12
Seuil automatique	7	3	7	9	7	5

Tableaux 5.5: *Le choix du seuil et de k cluster pour IsoData.*

Seuil automatique = la moyen entre la similarité minimal et la similarité maximal.

Nous constatons que le choix des paramètres, seuil et nombre des clusters initiaux influe directement sur les résultats et que la combinaison du seuil automatique en démarrant un seul

groupe de cluster qui contient toute l'application nous donne le résultat le plus proche aux résultats attendus.

5.2.3.2 Le choix de nombre des itérations :

Puisque IsoDATA se termine soit en atteignant les itérations fixées soit avec la stabilisation des groupe, nous avons voulu vérifier est ce que le nombre des itérations fixées influe sur la convergence de l'algorithme, donc nous avons laissé ouvert le nombre d'itération et finalement nous avons constaté que l'algorithme converge avant d'atteindre le nombre que nous avons fixé qui est égale au nombre des classes initiales.

Les résultats résumés dans le tableau suivant (le seuil choisi automatiquement) :

Le nombre des centres initiaux.	Les exemples	
	Ex 1 / Le nombre des itérations jusqu'à la stabilisation.	Ex 2 / Le nombre des itérations jusqu'à la stabilisation.
Min	5 itération/7 services	3 itération/ 3 services
Max	5 itération/ 7 services	2 itération/ 9 services
La moitié du nombre des classes initiaux.	3 itération /7 services	2 itération/ 5 services

Tableaux 5.6: *Tableau comparatif des nombres des itérations pour la méthode IsoData.*

Pour aboutir à un nombre des services similaire à celui des résultats attendus, il faut choisir :

- Le nombre des clusters initiaux – le min(1).
- Le seuil = la moyenne entre la distance minimale et la distance maximal (reflète le seuil automatique).
- Le nombre des itérations I_{max}= le nombre des classes totales de l'application.

5.3 La comparaison entre les trois algorithmes:

Un bon algorithme donne des bons résultats dans un petit temps, nous avons étudié les choix des paramètres de chaque algorithme dans les parties précédentes et dans cette partie nous essayerons de comparer les résultats (tableau 5.7) en calculant le temps d'exécution (tableau 5.8) des algorithmes pour les exemples (1 et 2), afin d'extraire les avantages et les inconvénients.

Nous nous sommes fixés les choix qui donnent les meilleurs résultats comme suit :

- Pour l'algorithme hiérarchique nous avons choisi 75% comme une valeur par défaut de coupure du dendrogramme.
- Pour l'algorithme Soustractive Clustering :
  - ✓ Le seuil automatique.
  - ✓ Nous essayerons toutes les options pour les centres potentiels.
- Pour l'algorithme IsoData :
  - ✓ Le seuil automatique.
  - ✓ Le nombre initial des clusters min(1).
  - ✓ Le nombre des itérations I_{max}= nombre des classes totales de l'application.

L'algorithme hiérarchique			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	CalcMachineNumber OperatorControlCenter Calculator CalculatorException, CalculatorTester E Jcalc_math Jcalc_trig Variable_interface PI VariableTable OperatorChecker Entries	Service 1	CalcMachineNumber
Service 2	Jcalc	Service 2	OperatorControlCenter
Service 3	ResultsList	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet

Service 4	GuiCommandLine	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	Jcalc_applet	Service 5	VariableTable OperatorChecker
		Service 6	PI
		Service 7	Entries GuiCommandLine ResultsList

Tableau 5.7: *Comparaison des résultats de l'algorithme hiérarchique pour l'exemple 1.*

L'algorithme hiérarchique			
Les services résultants pour l'exemple 2			
Le résultat obtenu		Le résultat attendu	
Service 1	Mul Mul_param Mul_affiche Mul_fnct Sous_fnct Sous_affiche Sous Sous_param	Service 1	Add Add_affiche Add_fonct Add_param
Service 2	Add Add_fonct Add_param Add_affiche	Service 2	Mul Mul_affiche Mul_fnct Mul_param
		Service 3	Sous Sous_affiche Sous_fnct

			Sous_param

Tableau 5.8 : *Comparaison des résultats de l'algorithme hiérarchique pour l'exemple 2.*

L'algorithme SC : le centre aléatoire			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	CalcMachineNumber OperatorControlCenter Calculator CalculatorException CalculatorTester E Variable_interface PI OperatorChecker	Service 1	CalcMachineNumber
Service 2	Entries	Service 2	OperatorControlCenter
Service 3	GuiCommandLine ResultsList Jcalc	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet
Service 4	Jcalc_math Jcalc_trig	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	PI	Service 5	VariableTable OperatorChecker
Service 6	VariableTable	Service 6	PI
		Service 7	Entries

			GuiCommandLine ResultsList

Tableau 5.9: *Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre aléatoire).*

L'algorithme SC le centre : la classe dont leur similarité est le min			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	OperatorControlCenter CalculatorTester Variable_interface VariableTable	Service 1	CalcMachineNumber
Service 2	CalcMachineNumber Calculator CalculatorException, E Jcalc_applet OperatorChecker	Service 2	OperatorControlCenter
Service 3	GuiCommandLine ResultsList Jcalc	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet
Service 4	Jcalc_math Jcalc_trig	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	Entries	Service 5	VariableTable OperatorChecker
Service 6	PI		

		Service 6	PI
		Service 7	Entries GuiCommandLine ResultsList

Tableau 5.10: *Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre est la classe dont leur similarité est le min).*

L'algorithme SC le centre : la classe dont leur nombre des fonctionnalités est le max.			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	CalcMachineNumber CalculatorTester Entries CalculatorException,	Service 1	CalcMachineNumber
Service 2	GuiCommandLine ResultsList Jcalc	Service 2	OperatorControlCenter
Service 3	Jcalc_applet	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet
Service 4	Jcalc_math Jcalc_trig OperatorControlCenter	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	OperatorChecker Calculator	Service 5	VariableTable OperatorChecker
Service 6	E Variablc_intcrfacc		

Service 7	PI	Service 6	PI
Service 8	VariableTable		
		Service 7	Entries GuiCommandLine ResultsList

Tableau 5.11: *Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre est la classe dont leur nombre des fonctionnalités est le max).*

L'algorithme SC le centre : la classe dont leur nombre des fonctionnalités est le max.			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	CalcMachineNumber CalculatorTester Entries CalculatorException,	Service 1	CalcMachineNumber
Service 2	GuiCommandLine ResultsList Jcalc	Service 2	OperatorControlCenter
Service 3	Jcalc_applet	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet
Service 4	Jcalc_math Jcalc_trig OperatorControlCenter	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	OperatorChecker Calculator	Service 5	VariableTable

Service 6	E Variable_interface		OperatorChecker
Service 7	PI	Service 6	PI
Service 8	VariableTable		
		Service 7	Entries GuiCommandLine ResultsList

Tableau 5.12 : *Comparaison des résultats de l'algorithme SC pour l'exemple 1 (le centre est la classe dont leur nombre des fonctionnalités est le max).*

L'algorithme SC le centre : la classe dont la moyenne des similarités est le min			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	OperatorControlCenter CalculatorTester Variable_interface VariableTable CalcMachineNumber CalculatorException, Jcalc_applet E Calculator Jcalc OperatorChecker	Service 1	CalcMachineNumber
Service 2	PI	Service 2	OperatorControlCenter
Service 3	Jcalc_math	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet

Service 4	ResultsList	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	GuiCommandLine	Service 5	VariableTable OperatorChecker
Service 6	Entries		
Service 7	Jcalc_trig	Service 6	PI
		Service 7	Entries GuiCommandLine ResultsList

Tableau 5.13: *Comparaison des résultats de l'algorithme Soustractive Clustering pour l'exemple 1 (le centre est la classe dont la moyenne des similarités est le min).*

L'algorithme Soustractive Clustering le centre aléatoire			
Les services résultants pour l'exemple 2			
Le résultat obtenu		Le résultat attendu	
Service 1	Add Add_affiche Add_fonct Add_param	Service 1	Add Add_affiche Add_fonct Add_param
Service 2	Mul Mul_affichc Mul_fnct Mul_param	Service 2	Mul Mul_affiche Mul_fnct Mul_param
Service 3	Sous Sous_affiche Sous_fnct Sous_param	Service 3	Sous Sous_affiche Sous_fnct Sous_param

Tableau 5.14: *Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre soit le min.*

L'algorithme Soustractive Clustering le centre : La classe dont leur similarité est le min			
Les services résultants pour l'exemple 2			
Le résultat obtenu		Le résultat attendu	
Service 1	Add Add_affiche Add_fonct Add_param	Service 1	Add Add_affiche Add_fonct Add_param
Service 2	Mul Mul_affiche Mul_fnct Mul_param	Service 2	Mul Mul_affiche Mul_fnct Mul_param
Service 3	Sous Sous_affiche Sous_fnct Sous_param	Service 3	Sous Sous_affiche Sous_fnct Sous_param

Tableau 5.15: *Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre est la classe dont leur similarité est le min.*

L'algorithme Soustractive Clustering le centre : la classe dont leur nombre des fonctionnalités est le max.			
Les services résultants pour l'exemple 2			
Le résultat obtenu		Le résultat attendu	
Service 1	Add Add_affiche Add_fonct Add_param	Service 1	Add Add_affiche Add_fonct Add_param
Service 2	Mul	Service 2	Mul

	Mul_affiche Mul_fnct Mul_param		Mul_affiche Mul_fnct Mul_param
Service 3	Sous Sous_affiche	Service 3	Sous Sous_affiche Sous_fnct Sous_param
Service 4	Sous_fnct		
Service 5	Sous_param		

Tableau 5.16: *Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre est la classe dont leur nombre des fonctionnalités est le max.*

L'algorithmc Soustractive Clustering le centre : la classe dont la moyenne des similarités est le min.			
Les services résultants pour l'exemple 2			
Le résultat obtenu		Le résultat attendu	
Service 1	Add Add_affiche Add_fonct Add_param	Service 1	Add Add_affiche Add_fonct Add_param
Service 2	Mul Mul_affiche Mul_fnct Mul_param	Service 2	Mul Mul_affiche Mul_fnct Mul_param
Service 3	Sous Sous_affiche Sous_fnct	Service 3	Sous Sous_affiche Sous_fnct

	Sous_param		Sous_param

Tableau 5.17: *Comparaison des résultats de l'algorithme SC pour l'exemple 2 dont le centre c'est la classe dont leur moyenne de similarité est le min.*

L'algorithme IsoData			
Les services résultants pour l'exemple 1			
Le résultat obtenu		Le résultat attendu	
Service 1	Variable_interface PI	Service 1	CalcMachineNumber
Service 2	CalcMachineNumber OperatorControlCenter Calculator CalculatorException, CalculatorTester E VariableTable OperatorChecker Jcalc_applet	Service 2	OperatorControlCenter
Service 3	Jcalc_trig	Service 3	Calculator CalculatorException, CalculatorTester Jcalc Jcalc_applet
Service 4	Entries	Service 4	E Jcalc_math Jcalc_trig Variable_interface
Service 5	GuiCommandLine	Service 5	VariableTable OperatorChecker
Service 6	Jcalc_math	Service 6	PI

Service 7	ResultsList Jcalc	Service 7	Entries GuiCommandLine ResultsList

Tableau 5.18: *Comparaison des résultats de l'algorithme IsoData pour l'exemple 1.*

L'algorithme IsoData			
Les services résultants pour l'exemple 2			
Le résultat obtenu		Le résultat attendu	
Service 1	Add Add_affiche Add_fonct Add_param	Service 1	Add Add_affiche Add_fonct Add_param
Service 2	Mul Mul_affiche Mul_fnct Mul_param	Service 2	Mul Mul_affiche Mul_fnct Mul_param
Service 3	Sous Sous_affiche Sous_fnct Sous_param	Service 3	Sous Sous_affiche Sous_fnct Sous_param

Tableau 5.19: *Comparaison des résultats de l'algorithme IsoData pour l'exemple 2.*

Le nombre des classes d'une application Java.	Le temps d'exécution de l'algorithme hiérarchique : le pourcentage valeur par défaut.	Le temps d'exécution de l'algorithme Soustractive Clustering.	Le temps d'exécution de l'algorithme IsoData.
Exemple 1	2940223 ns	Le centre aléatoire 172669 ns	3329636 ns
		Le centre c'est la classe de min similarité 333263 ns	
		Le centre c'est la classe qui a grandes nombre de fonctionnalité Le centre aléatoire 324811 ns	
		Le centre la classe dont la moyenne des similarités est le min 316963 ns	
Exemple 2	1380749 ns	Le centre aléatoire 31999 ns	1642168 ns
		Le centre c'est la classe de min similarité 149123 ns	
		Le centre c'est la classe qui a grandes nombre de fonctionnalité 463067 ns	
		Le centre la classe dont la moyenne des similarités est le min 87543 ns	

Tableau 5.20: *Tableau comparatif de temps d'exécution écoulé dans les trois algorithmes.*

Et on a résumé les limites et les avantages de chaque algorithme dans le tableau suivant :

Le nom de l'algorithme.	L'algorithme hiérarchique.	L'algorithme Soustractive Clustering.	L'algorithme IsoData.
Le temps d'exécution.	Le temps acceptable par rapport IsoData mais moins rapide que Soustractive Clustering.	C'est l'algorithme le plus rapide.	C'est l'algorithme le plus lent.
Le résultat obtenu.	Malheureusement cette méthode ne donne pas un bon résultat le résultat reste à tester par les expertes.	Fournit des résultats acceptable et plus proche au deux exemple surtout quand le centre c'est la classe dont la valeur de similarité est le min.	Fournis des résultats acceptable proche au premier exemple mais n'est pas plus proche au deuxième exemple.

Tableau 5.21: *Tableaux comparatif pour les trois algorithmes.*

6 Conclusion:

Dans ce dernier chapitre nous avons évalué nos algorithmes (hiérarchique, Soustractive Clustering, IsoData) suivant plusieurs critères afin de comparer leurs résultats. Notre évaluation montre que chaque algorithme a ses limites et ses avantages, et que le choix du meilleur algorithme dépend des besoins des utilisateurs.

Conclusion générale

Conclusion général et perspectives

Aujourd'hui, plusieurs entreprises se basent sur des systèmes d'information complexes, monolithiques, et inflexibles, parfois non conformes aux changements rapides du marché.

L'Architecture Orientée Services (AOS) apparaît comme la meilleure approche permettant l'intégration flexible des applications autonomes, distribuées et hétérogènes au sein et au-delà de l'entreprise. Plus qu'une nouvelle technologie ou méthode, c'est la convergence de plusieurs approches existantes, et l'émergence d'un style d'architecture et de gouvernance de SI.

Nous avons eu l'occasion à travers ce projet de manipuler cette technologie, en abordant le problème de la réingénierie des systèmes patrimoniaux (orienté objet) vers des nouveaux systèmes (orienté service). L'étape la plus importante dans le processus de migration vers les nouveaux systèmes c'est l'identification des services.

Pour automatiser cette phase d'identification nous avons eu recours à quelque algorithmes de classification non supervisé (hiérarchique, Soustractive Clustering, IsoData) .Nous les avons adapté de façon à les rendre plus flexible à nos besoin.

L'idée de base était de partir d'une application orientée objet, puis de regrouper les parties de cette application OO sous forme de groupes susceptibles de constituer des services. Nous avons dû choisir des critères qui nous ont permis d'orienter ce regroupement vers un respect total des conditions de qualité imposées par une architecture orientée service.

A la fin de cette expérience nous avons menu une phase de validation des résultats où nous avons étudié le comportement de nos algorithmes à travers plusieurs exemples, et à travers la comparaison avec des études similaires précédentes. Cette phase était énormément bénéfique puisqu'elle nous a permis d'améliorer nettement l'adaptation et le paramétrage des différents algorithmes utilisé.

Comme perspective nous proposons de tester notre projet d'identification automatique de service sur des SI et des projets réels en cours d'exploitation afin de valider les résultats par des experts du terrain et soumettre notre travail à des possibles améliorations.

Liste des webographies

[w1][http://msdn.microsoft.com/fr-fr/library/bb469924.aspx#wsdlexplained_topic03, 29/12/2014].

[w2][http://blog.xebia.fr/2009/03/04/soa-du-composant-au-service-le-contrat-standardise/, 29/12/2014].

[w3][https://tel.archives-ouvertes.fr/tel-00828598/document, 17/05/2015].

[w4][http://www.dotnet-france.com/Documents/WCF/Introduction%20a%20WCF.pdf, 17/05/2015].

[w5][http://www.redsen-consulting.com/2011/07/concepts-fondamentaux-soa/, 17/05/2015].

[w6][http://www.memoireonline.com/12/08/1644/SOA--Definition-Utilisation-dans-le-monde-de-la-banque-et-methodologie-de-test.html, 31/03/2015].

[w7][http://genexo-a-m.googlecode.com/files/SOA%20final.doc, 31/12/2014].

[w8][http://www.journaldunet.com/developpeur/tutoriel/theo/051013-explication-soa.shtml, 31/12/2014].

[w9][https://www.ibisc.univ-evry.fr/~tmelliti/cours/CPAR/cours6.pdf, 31/12/2014].

[w10][http://www.cirano.qc.ca/pdf/publication/2003RB-07.pdf, 31/12/2014].

[w11][http://www.zdnet.fr/actualites/soa-comprendre-l-approche-orientee-service-39206712.htm, 05/06/2015].

[w12][http://download.docslide.fr/uploads/check_up03/212015/5560b541d8b42afe3b8b497a.pdf, 05/06/2015].

[w13][http://www.lirmm.fr/~seriai/encadrement/m2recherche/2012/sujet3.html, 20 /03/2015].

[w14][http://www.ledicodumarketing.fr/definitions/Reingenierie-des-Processus-de-Gestion.html, 20 /03/2015].

[w15][http://tvquality.verollet.fr/files/Reingenierie_Rapport_Tornier_Verollet.pdf, 21/03/2015].

[w16][http://www.journaldunet.com/solutions/0403/040323_reengineering.shtml, 21/03/2015].

[w17][http://www.ricoh.fr/common-business-terms/business-process-re-engineering/, 21/03/2015].

[w18][http://hexawaretechnologies.fr/lmm-eng-3.htm, 21/03/2015].

[w19][http://www.iro.umontreal.ca/~dufour/cours/ift3912-h10/notes/19-Evolution.pdf, 21/03/2015].

[w20][http://publicationslist.org/data/a.april/ref295/M%C3%A9moire%20Serge%20Bri%C3%A8re.pdf, 21/03/2015].

[w21][http://www.journaldunet.com/solutions/0311/031110_soa.shtml, 24/03/15].

[w22][http://blog.sodifrance.fr/le-soa-la-mort-de-lobjet/, 24/03/15].

[w23][https://liris.cnrs.fr/inforsid/sites/default/files/2011_11.pdf, 24/03/15].

[w24][http://www.memoireonline.com/12/08/1644/m_SOA--Definition-Utilisation-dans-le-monde-de-la-banque-et-methodologie-de-test1.html, 24/03/15].

[w25][http://www.thibaut-delcroix.fr/blog/article-34-introduction-%C3%A0-la-soa-ou-architecture-orient%C3%A9e-services.html, 25/03/15].

[w26][http://laurent-audibert.developpez.com/Cours-UML/?page=introduction-modelisation-objet, 24/03/15].

[w27][http://scd-theses.u-strasbg.fr/1325/01/blansche06classification.pdf,18/12/2014].

[w28][http://apiacoa.org/publications/teaching/data-mining/clustering.pdf, 26/03/2015].

[w29][http://dspace.univ-tlemcen.dz/bitstream/112/1045/4/Memoire.pdf, 26/03/2015].

[w30][http://lipn.univ-paris13.fr/~cabanes/Publi/Nat/Cabanes_EGC08.pdf, 27/03/15].

[w31] [https://samos.univ-paris1.fr/archives/ftp/preprints/samos148.pdf, 09/04/2015].

[w32][http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20070038185.pdf, 10/04/2015].

[w33][http://research.cs.tamu.edu/prism/lectures/pr/pr_l15.pdf, 10/04/2015].

[w34][http://fourier.eng.hmc.edu/e161/lectures/classification/node12.html,10/04/2015].

[w35][http://www.eclipsezone.com/eclipse/forums/t106423.rhtml, 17/04/2015].

[w36][Xu Zhang : Analyse de la similarité du code source pour la réutilisation automatique de tests unitaires à l'aide du CBR, université LAVAL, Québec, Canada, mémoire, 2013].

[w37][http://sourceforge.net/projects/bfegler/, 13/06/2015].

[w38][http://www.thibaut-delcroix.fr/blog/article-34-introduction-%C3%A0-la-soa-ou-architecture-orient%C3%A9e-services.html, 21/04/2015].

Liste des bibliographies

[1] Hüsemann Stefan, « SOA : L'utilité organisationnelle, technique et financière de l'architecture orientée service », Mémoire de magister, Université de Fribourg, Suisse, Août 2013.

[2] Brown A, Johnston S, Kelly K, « Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications », CA: Rational Software Corporation, Santa Clara, 2002.

[3]M.Nakamura, H.Igaki, T.Kimura, K.Matsumoto, « Extracting service candidates from procedural programs based on process dependency analysis », IEEE Asia-Pacific Services Computing Conference, p.484, 491, APSCC 2009, 7-11 Dec 2009.

[4]Livre blanc BEA, « SOA et virtualisation : quelle complémentarité », 2008.

[5] Cyrielle Lablanche, Florens Seine, Sébastien Gastaud, « Les Web Services », Mémoire de licence, Université de Nice-Sophia, Antipolis, 2004–2005.

[6] Michel Leblanc, « Les web services: Définition, technologies, acteurs, impacts sur les entreprises et problèmes », Montréal, Novembre 2002.

[7] Dan Vodislav, « Services web », Cours IED Master informatique M1, Université de Cergy-Pontoise, France.

[8] EJ.Chikofsky, JH.Cross II, « Reverse Engineering and Design Recovery: Taxonomy », IEEE Software, 7(1), p.13–17, Janvier 1990.

[9] Oracle, « Oracle IT Modernization Series: The Types of Modernization», Oracle White Paper, September 2008.

[10] G.Lewis, E.Morris, D.Smith, L.Wrage, L.O'Brien, « Service-Oriented Migration and Reuse Technique (SMART) », Proceedings of 13th IEEE International Workshopon Software Technology and Engineering Practice (WSTEP'05), September 2005.

[11] Sodki Chaari, « Interconnexion des processus Interentreprises: une approche orientée services », Thèse de doctorat, Lyon, 18 Décembre 2008.

[12] Séridi Ali, Seriai djamel-Abdelhak, Bourbia Riad, « Approches Pour L'Evolution Des Systèmes Patrimoniaux Vers Une Architecture Orientée Service », WOTIC 2011 : The Fourth Workshop on Information Technologies and Communication Casablanca, Morocco, Oct 13-15 2011.

[13] Lotfi Khodja, « Contribution à la Classification Floue non Supervisée », thèse de doctorat, Université de Savoie, France.

[14] Dawid Weiss, « Descriptive Clustering as a Method for Exploring Text Collections », PhD thesis, Institute of Computing Science Poznàn, Poland, 2006.

[15] Pavel Berkhin, « Survey of Clustering Data Mining Techniques», Accrue Software CA, USA, 2002.

[16] S.C.Johnson, «Hierarchical Clustering Schemes », Psychometrika, Vol.32, p.241-254, 1967.

[17] Ronald R.Yager, « S-mountain method for obtaining focus points from data ». International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 16(6), p.815–828, 2008.

[18] Gérard Govaert, « Analyse des données », IC2 (série Traitement du signal et de l'image), Lavoisier, 2003.

[19] L.Candillier, « Contextualisation, visualisation et évaluation en apprentissage non supervisée », Thèse de doctorat, Université Charles De Gaulle Lille 3, France, 2006.

[20] A.K.Jain, M.N.Murty, P.J.Flynn, « Data clustering: a review », ACM Computer Survey, 31(3), p.264–323, 1999.

[21] Kelaiaia Abdessalem, « Classification non supervisée de textes arabes appliquée à la recherche documentaire », Mémoire de magister, Université 08 mai 45 de Guelma, Algérie, 2007.

[22] Nesma Settouti , « Renforcement de l'Apprentissage Structurel pour la reconnaissance du Diabète », Rapport de recherche, Université Abou Bekr Belkaid de Tlemcen, Algérie, 2011.

[23] Lazher Sadaoui , « Evaluation de la cohésion des classes : une nouvelle approche basé sur la classification »,Mémoire, Université du Québec, juin 2010.

[24] Sylvain Chardign, « Extraction d'une architecture logicielle à base de composants depuis un système orienté objet », Thèse de doctorat, Université de Nantes, 23 octobre 2009.

[25] Lionel Briand, Prem Devanbu, Walcelio Melo, «An investigation into coupling measures for C++, In Proc of the Int », Conference on Software Engineering, ACM, p.412–421, 1997.

[26] Shyam Chidamber, Chris Kemerer, « A metrics suite for object-oriented design », IEEE Trans on Software Engineering, 20(6), p.476–493, juin 1994.

[27] Shyam Chidamber, Chris Kemerer, «Towards a metrics suite for object oriented design», Conference proceedings on Object-oriented programming systems, languages, and applications, p.197–211, ACM Press, New York, NY, USA, 1991.

[28] Martin Hitz, Behzad Montazeri, «Measuring coupling and cohesion in object-oriented systems. In Proc. Intl. Sym. on Applied Corporate Computing », Nov 1996.

[29] Y.Lee, B.Liang, S.Wu, F.Wan, «Measuring the coupling and cohesion of an object-oriented program based on information flow», ICSQ'95, p.81–90, 1995.

[30] W. Li, S.Henry, « Object oriented metrics that predict maintainability », Journal of Systems and Software, 223, p.111–122, 1993.

[31] Seza Adjoyan, Abdelhak-Djamel Seriai, Anas Shatnawi, «Service Identification Based on Quality Metrics - Object-Oriented Legacy System Migration Towards SOA», Conference on Software Engineering and Knowledge Engineering (SEKE), Vancouver-Canada, 2014.