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 de Master

Filière : Informatique

Option : Informatique Académique

Thème :

Identification de Services Web à partir d'une application orientée objet par Algorithme de Classification Hiérarchique

Encadré Par :

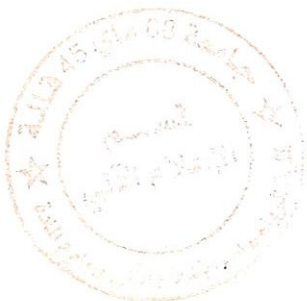
Mr. SERIDI ALI

Présenté par :

IRIVUZUMUREMYI DENIS

TAHIYA AMIR

Juin 2014



Remerciement

Nous rendons grâce à DIEU de nous avoir accordé la santé, la force et beaucoup de courage qu'il fallait pour commencer et terminer ce mémoire. Nous remercions nos parents pour leur soutien incessant sans condition, nos amis pour leurs encouragements et tous les enseignants qui ont contribué pour notre formation depuis l'école primaire. Mention spéciale à notre encadreur Mr. SERIDI Ali, pour ses conseils fructueux et sa disponibilité illimitée toute la durée de sa réalisation.

Nos remerciements les plus vifs s'adressent à monsieur le président ainsi qu'aux membres de jury pour avoir accepté d'examiner et d'évaluer ce modeste travail.

Et nous ne saurions pas terminer sans exprimer notre reconnaissance à tous nos camarades étudiants de la promotion du département d'informatique, en particulier notre promotion en INFORMATIQUE ACADEMIQUE.

En bref, nous remercions tous ceux qui nous ont aidé de près ou de loin que ce soit par leur soutien moral ou physique pour la réalisation de ce projet.

IRIVUZUMUREMYI DENIS

TAHIYA AMIR

Dédicaces

J'ai l'honneur d'exprimer ma gratitude à vous qui, de près ou de loin, ne cessez de satisfaire à mes besoins, de façon continu, volontiers. Vous m'avez témoigné de l'amour dans les moments d'activité scolaire et académique jusqu'à maintenant.

Je dédie ce travail

A Ma Mère NIRAGIRA par ses initiatives, m'a montrée et guidée sur le chemin de l'école ; à Mon père regretté NKOBONGO, malgré que Sa vie ne l'ait pas accordé assez de temps pour moi; à Ma sœur NDUWAMUNGU qui m'a épargné des reproches et enduré la fatigue pour ma cause; à Mon frère BIGIRINDAVYI en particulier pour ses conseils, son temps et ses appels inégalés, vous m'avez témoignés un amour plus grande que je puisse l'exprimer par écrit ; au reste de ma famille pour leurs conseils et recommandations et, leurs biens qu'ils m'ont partagé lors de ma formation : Vous avez forgé ma joie et mon bonheur ;

A toute la communauté qui m'a tenu compagnon par leurs prières.

A Mes enseignants qui m'ont partagé leur savoir, mes camarades et toute personne que Dieu a placée sur mon chemin ; à mes amis, mes voisins compatriotes et étrangers : vos services ont marqué ma vie à jamais ;

A vous TAHIYA AMIR, mon binôme, à notre Encadreur SERIDI ALI, c'est grâce à vos efforts, vos critiques et votre attention que ce document tient son existence;

Que le Dieu de ma foi récompense vos œuvres.

IRIVUZUMUREMYI DENIS

DEDICACE

Je ne peux pas commencer sans prononcer gloire à **Allah** le tout puissant et le tout miséricordieux de m'avoir donné **Santé, Force et Courage** de réaliser ce travail.

A celle qui m'a donnée la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur, ma réussite, qui a crue à moi et qui m'a encouragée jours et nuits, à ma ravissante mère **Sarifati Saindou**.

A mon père **Amir Houmdi** qui me disais toujours le savoir est primordial que toutes choses au monde.

A ma tente adorable **Zaouriya Saindou** qui est toujours présent pour moi, qu'elle s'occupa de moi depuis mon enfance en me donnant tout ce que je veux, sans oublier son mari et les autres tentes.

A mon oncle **Malide Saindou** et Soimdine Saindou.

A tout(e)s mes frères et mes sœurs, sans oublié aussi le nom de ma petite sœur que j'aime vraiment **Naziya Amir**.

A tout(e)s mes cousins et cousines, sans oublié le nom de mon bon cousin Assiandi Ibrahim.

A mes neveux et mes nièces.

A mes deux grandes mères Carima et Masaindou.

A mon binôme Denis, sans oublié Axel, Juste et Bengono qu' m'ont soutenues tout au long de mon parcours.

A mes ami(e)s proches Onzairou, Oumou K, Elia et Abasse, et mes ami(e)s d'enfance.

A tous mes enseignants, sans oublié mon encadreur Mr Seridi Ali.

A toute la promotion 2014, le département d'informatique et tout(e)s mes compatriotes qui sont en Algérie.

A tout(e)s mes ennemis qui m'ont poussé(e)s d'aller loin dans…

Je vous aime tout(e)s et merci beaucoup.

Tahiya Amir

Le 14-06-2014

Résumé

L'informatique en tant que science et outil a contribué de façon active dans le progrès scientifique et économique de l'humanité. L'arrivée de l'internet a encore vulgarisé l'utilisation de l'informatique en tant que moyen incontournable et a ouvert de nouveaux horizons aux entreprises pour exploiter ce réseau dans le partage et l'échange de l'information instantanée. L'architecture orientée service (SOA) est apparue comme une solution aux Systèmes d'information des entreprises qui permet, agilité, réutilisabilité et interopérabilité. Les études ont montré que le coût de passage vers la SOA en utilisant les techniques de réingénierie est beaucoup moins que celui émanant d'un redéveloppement nouveau du SI.

Notre étude s'inscrit dans le cadre de la réingénierie des systèmes orientés objet (OO) qui est prédisposés à évoluer vers des systèmes orientées service puisque les applications OO se basent sur le principe de modularité, d'encapsulation, d'abstraction et de réutilisabilité qui sont repris par la SOA.

Le travail que nous avons mené, consiste à réaliser un outil selon une approche basé sur l'algorithme de classification hiérarchique, qui prend en entrée une application orientée objet constituée d'un ensemble de classes Java et donne comme sortie une proposition de leur regroupement sous forme de services web. L'objectif principal de ce travail est l'identification des services potentiels à travers l'analyse du code orienté objet et le regroupement des classes fortement couplées et fortement cohésives par l'algorithme de regroupement hiérarchique.

Mots clés :

Architecture Orientée Service, Système patrimonial, Application orientée objet, Approche d'identification de services, Migration de systèmes patrimoniaux, Réingénieric, Classification ascendante hiérarchique, Métrique logicielle.

Sommaire

Listes des figures

Liste des tableaux

ACH: Algorithme de Classification Hiérarchique

SOA: Oriented Services Architecture

CBA: Component-Based Architecture

XML: eXtencible Marcup Langage

W3C: World Wide Web Consortium

WSDL: Web Service Description Language

SOAP: Simple Object Access Protocol

HTTP: HyperText Transfert Protocol

IDL: Interface Definition langage

UDDI: Universal Description Discovery and integration

WSOA: Web Service Oriented Architecture

SAAJ: SOAP with Attachement API for Java

API: Application Programming Interface

JAXM: Java API for XML Messaging

JAX-WS: JAX –WS (Java API for XML based web services)

RL: Software Reengineering

TI: Technology Information

UML: Unified Modeling langage

INTRODUCTION GENERALE

L'informatique reste une science pleine de défis et d'interrogations. Elle ne cesse de se développer et d'atteindre de nouveaux horizons. Un des pas géants qui a vulgarisé l'utilisation de l'informatique est l'apparition de l'internet. Ainsi de nouvelles applications utilisant le web sont de plus en plus croissantes. La disponibilité de ce réseau mondiale a suscité l'intérêt des entreprises qui y ont trouvé un moyen très abordable économiquement pour partager leurs ressources et échanger les informations de façon instantanée.

Parmi les nouveaux paradigmes de programmation apparues avec la vulgarisation de l'internet c'est l'architecture orientée service ou plus communément appelée Service Oriented Architecture (SOA). Cette architecture a connu ces dernières années un grand engouement des entreprises de tout secteur et de toute taille en raison de ses avantages économiques et technologiques. Pour exploiter ces avantages, plusieurs organisations ont décidé de faire évoluer leurs systèmes patrimoniaux (SP) existants vers une telle architecture. La Migration vers la SOA est devenue l'une des techniques importantes de modernisation des SP. Elle aide les organisations d'une part à réutiliser leurs anciens systèmes en leurs donnant une nouvelle vie, et d'autre part à profiter des avantages des systèmes à base de service [43].

Un des problèmes qui se posent dans ce domaine, est comment récupérer des applications existantes, fiables et toujours en cours d'utilisation, malheureusement développées avec d'anciennes technologies (appelées applications patrimoniales) et les faire évoluer vers une architecture SOA.

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

Nous avons choisi la réingénierie des applications orientées objet (OO) qui sont prédisposés à évoluer vers des systèmes orientées service puisque les applications OO se basent sur le principe de modularité, d'encapsulation, d'abstraction et de réutilisabilité qui sont repris par la SOA.

L'identification de service est une phase critique dans l'ingénierie des services. Notamment dans la conception à base d'une architecture SOA. La position, le sens et les activités d'identification de service diffèrent d'un contexte à un autre.

d'analyse. Dans cette approche top-down d'identification de service, les exigences opérationnelles sont analysées et modélisées de manière à identifier les services d'affaires.

Cependant dans notre cas de la réingénierie et la migration vers SOA, l'identification de service est effectuée dans la phase de la retro-ingénierie des systèmes existants. C'est une approche Bottom-up qui démarre de l'analyse du code source pour extraire les services candidats qui respectent les caractéristiques de qualité d'une architecture SOA.

Notre travail consiste à réaliser un outil selon une approche basé sur l'algorithme de regroupement hiérarchique et ainsi pour sa mise en œuvre qui prend en entrée une application orientée objet constituée d'un ensemble de classes Java et donne comme sortie une proposition de leur regroupement sous forme de services web. L'objectif principal de ce travail est l'identification des services potentiels à travers l'analyse du code orienté objet et le regroupement des classes fortement couplées et fortement cohésives par l'algorithme de regroupement hiérarchique.

Notre mémoire s'articule sur cinq chapitres

- ✓ **Chapitre 1:** Où nous allons introduire brièvement les concepts de base relatifs à l'architecture orientée service et les techniques d'implémentation utilisées pour mettre en œuvre une architecture orientée service.
- ✓ **Chapitre 2:** Il a pour objectif de présenter les concepts de la réingénierie logicielle, de migration d'une application orientée objet vers une architecture orientée services et enfin, nous mettons l'accent sur les approches d'identification des services.
- ✓ **Chapitre 3:** Nous exposons les principes de la classification hiérarchique, son utilisation pour regrouper un ensemble d'individus ainsi que son application.
- ✓ **Chapitre 4:** La conception qui est la feuille de route de notre travail. Nous allons expliquer la manière par laquelle nous avons conçu notre outil, comment nous avons utilisé et adapter l'algorithme de regroupement hiérarchique, tout cela après avoir spécifié les taches et les objectifs à atteindre pour réaliser cet outil.
- ✓ **Chapitre 5:** Ce chapitre comportera les détails d'implémentation, il présente les outils utilisés et les résultats obtenus.

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

CHAPITRE 1

SOA ET SERVICES WEB

1.1. Introduction et Historique

Aujourd'hui l'accès aux systèmes d'information s'appuie de plus en plus sur la technologie internet. Les efforts de standardisation dans ce contexte ont accentué l'engouement des organisations (aussi bien académiques, industrielles, commerciales, ou institutionnelles) pour l'utilisation de l'internet et ont permis l'émergence des services web comme support de développement des applications accessible par internet. Ainsi, les technologies associées aux services web sont devenues incontournables pour le développement d'applications interagissant les uns avec les autres par le biais de l'internet [1].

Certes, un service Web est une application appelable via Internet par une autre application d'un autre site Internet permettant l'échange de données (de manière textuelle) afin que l'application appelante puisse intégrer le résultat de l'échange à ses propres analyses. Les requêtes venant de l'application appelante et les réponses sont soumises à des standards et sont normalisés à chacun de leurs échanges [2].

L'architecture orientée services est une approche architecturale de dernière génération qui permet de passer d'une vision «application» à une vision «services». La SOA réunit un ensemble de concepts tels que la composition, l'autonomie, le couplage faible, l'abstraction, la granularité, etc. [3].

Les concepts de la programmation et des architectures logicielles n'ont cessés d'évoluer ces dernières années. Après que les techniques conventionnelles de programmations ont été remplacées par le développement de modèles à objets, la programmation par composants est apparue et a permis un développement plus modulaire des applications. Basé sur cette technique de programmation par composant, un nouveau concept architectural est né: l'architecture orientée services ou SOA (Service Oriented Architecture). Voici un schéma illustratif de l'évolution de la programmation jusqu'à SOA :

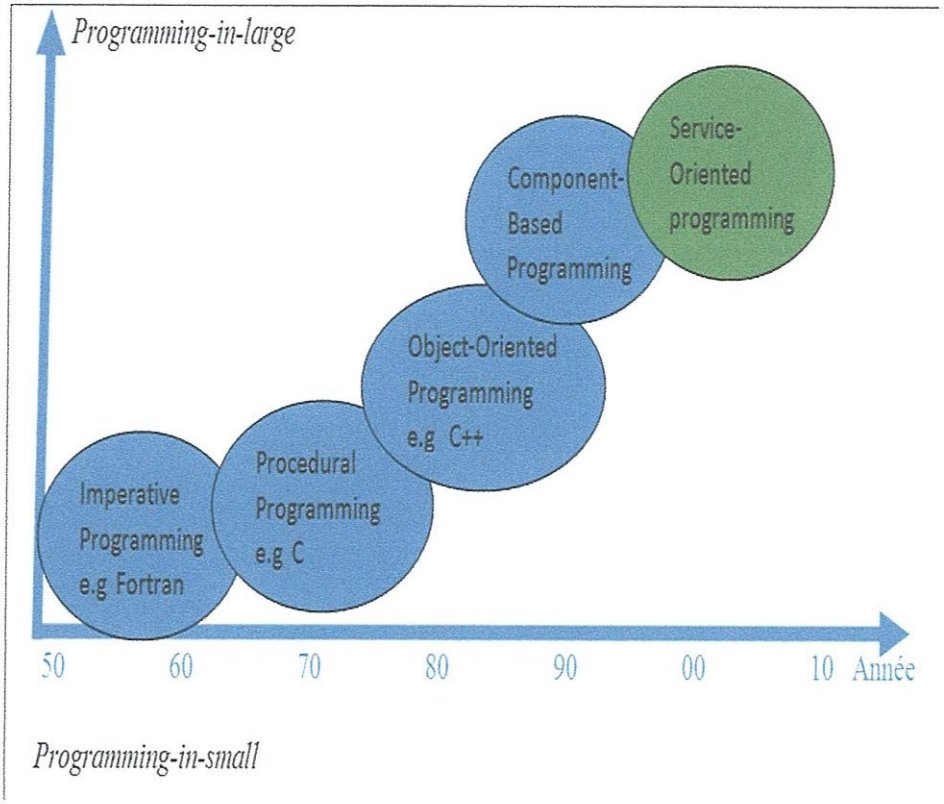


Figure 1. 1 : *Historique de développement de la programmation jusqu'à SOA [41]*

Dans ce chapitre on va parler brièvement les concepts de base de SOA, ses principes généraux, la technologie utilisée pour créer de services web en présentant les différents protocoles qui interviennent lors de son implémentation, la méthodologie de développement de services web et en fin on terminera par les services web Java.

1.2. Les concepts de base de SOA

Les concepts d'une architecture orientée services sont nécessaires à étudiés. Nous permettrons de comprendre les constituants principaux de cette architecture. Comme concepts Nous présentons en bref les trois notions suivantes : services, service métier et service web.

1.2.1. Services

Un service est un composant logiciel distribué, exposant les fonctionnalités à forte valeur ajouté d'un domaine métier. Un service est un composant autonome qui ne dépend d'aucun contexte ou service externe, il est aussi l'unité atomique d'une architecture SOA [4].

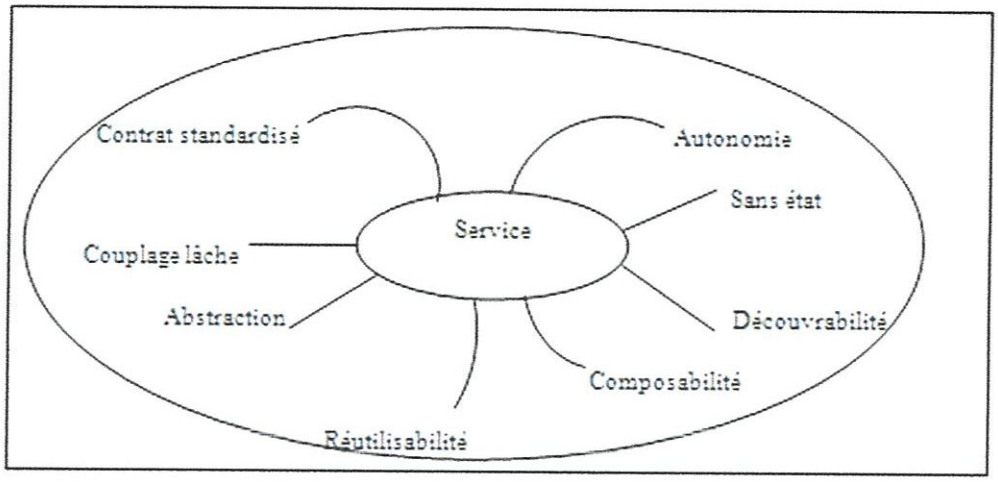


Figure 1. 2 : *Les propriétés de service*

Les caractéristiques associées aux services permettent de construire le fondement d'une architecture orientée service. En effet, plusieurs notions associées aux services constituent les piliers d'une architecture orientée service [5]

- ✓ Autonomie : Un service doit exercer un contrôle fort sur son environnement d'exécution sous-jacent. Plus ce contrôle est fort, plus l'exécution d'un service est prédictible.
- ✓ Sans état (stateless) : Un service doit minimiser la consommation de ressources en déléguant la gestion des informations d'état quand cela est nécessaire.
- ✓ Découvrabilité : Un service est complété par un ensemble de métadonnées de communication au travers desquels il peut être découvert et interprété de façon effective.
- ✓ Composabilité : Un service doit être conçu de façon à participer à des compositions de services.
- ✓ Réutilisabilité : Un service exprime une logique agnostique et peut ainsi être positionné comme une ressource réutilisable.

- ✓ Abstraction : Le contrat d'un service ne doit pas contenir que les informations essentielles à son invocation en omettant les informations non essentielles. Certes, seules les informations importantes doivent être publiées.
- ✓ Couplages lâches : Le couplage entre services est un couplage lâche et les communications peuvent être synchrones ou asynchrones.
- ✓ Contrat standardisé : L'ensemble des services d'un même système technique sont exposés à travers d'un contrat respectant les mêmes règles de standardisation.
- ✓ Communication par message: Ils s'échangent de messages respectant le contrat avec un format commun par exemple le format XML. Dans le cas de transactions complexes, un message peut transiter par de multiples services avant de revenir à l'initiateur. Chaque service intervient alors sur le message pour le compléter ou le modifier.
- ✓ Politique de fonctionnement: La relation entre les services est gouvernée par des politiques : chaque service doit adhérer à la politique du tiers avec lequel il désire interagir pour coopérer. Ces politiques permettent de préciser le cadre d'exécution du contrat : par exemple les règles de sécurités, le niveau transactionnel, la fiabilité de messages, la synchronisation, la qualité de service,....
- ✓ Frontière explicites : En effet, les frontières entre services sont explicites. Par ailleurs, le prérequis et les restrictions pour traverser la frontière et accéder aux services sont clairement définis et connus : identité, protocoles....

1.2.2. Les services métier

Un service métier est la représentation d'une activité métier élémentaire ou complexe. Par exemple l'annulation d'une commande est un ordre simple de suppression. Mais le processus de recrutement d'un nouvel employé dans une entreprise est représenté par un service plus complexe. Un service métier vu par un processus peut combiner plusieurs services de granularité plus fine [6].

1.2.3. Les services Web

Un service web est un programme informatique permettant la communication et l'échange des données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit d'un ensemble de fonctionnalités exposées sur internet ou sur intranet, par des applications ou machines, sans intervention humaine, et en temps réel. Le service Web se

caractérise par une standardisation des implémentations, par une localisation à distance des services et par une récupération de l'interface d'accès permettant l'exécution du traitement correspondant. Selon Gartner Group, le service web est à ce jour et il sera probablement demain au cœur de la SOA même si toute technologie objet avancée peut servir de base à la mise en place d'une architecture orientée services [7].

1.3. La différence entre composant et service

Il s'avère important de citer la différence entre composant et service puisque l'origine de la programmation orientée service est l'orienté par composant. Il n'y a pas vraiment des limites très claires qui séparent SOA (Service-Oriented Architecture) et CBA (Component-Based Architecture), en principe la SOA est l'amélioration de CBA dans le cas où un composant peut être utilisé comme un service web [41].

La différence entre SOA et CBA peut être résumée dans les points suivants :

- ✓ Niveau de l'encapsulation : L'une à gros grain (qui est le service) et l'autre à grain fine (qui est le composant).
- ✓ Travailler avec les composants c'est manipuler du code tandis que travailler avec les services, c'est utiliser certains fonctions distantes à travers le réseau sous un certains contrats.
- ✓ La composition de services dans un processus de haut niveau est totalement différente de lier quelques composants ensemble dans une application.
- ✓ Un contrat de service est totalement différent aux interfaces de composants.
- ✓ CBA pourrait être très pertinente à des fournisseurs de services dans le cas où c'est la façon d'implémentation d'un service particulier. Par contre un service peut être implémenté par composant, objet, ou un script, donc d'un point de vue SOA la CBA n'est pas vraiment pertinente.

1.4. Principes généraux d'une architecture orientée services

Il n'existe pas à proprement parlé de spécifications officielles d'une architecture SOA. Cependant les principales notions fédératrices que l'on retrouve dans une telle architecture sont les suivantes : la notion de service, la description du service, la publication et

la découverte des services. Dans ce qui suit on va présenter un aperçu général sur cette architecture SOA [42].

1.4.1. Définition SOA

La SOA est un ensemble des 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 [8].Cette idée d'un «contrat» bien défini, rempli par un prestataire et utilisé par des consommateurs, est au cœur des principes de la SOA. Les prestataires et consommateurs peuvent résider dans la même organisation ou dans des organisations séparées (voire des entreprises différentes).

1.4.2. L'objectif principal de SOA

Une architecture orientée service notée SOA est une architecture logicielle s'appuyant sur un ensemble des services simples. L'objectif d'une architecture SOA est donc de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services, fournies par des composants et faire une description d'un schéma d'interaction entre ces services [12].

L'idée principale est de cesser de construire la vie de l'entreprise autour des applications dépendantes des entités (une application par agence ou département) pour faire en sorte de construire une architecture logicielle globale décomposées en services correspondant aux processus métier de l'entreprise [13].

1.4.3. Les caractéristiques de SOA

Les caractéristiques essentielles de SOA peuvent être résumées dans les points suivants [11]:

- ✓ Réutilisation et composition: sont particulièrement puissantes pour créer des nouveaux processus d'affaires rapide et fiable.
- ✓ Recomposition : il s'agit de 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: ceci permet la communication des prestataires de services, l'extension des services, en modifiant les fournisseurs de services et les consommateurs. Tous ces éléments peuvent être en toute sécurité, grâce au couplement bien contrôlé.
- ✓ La capacité à construire progressivement le système: cela est particulièrement vrai pour toute intégration basée sur la SOA.

1.4.4. Structure de l'architecture SOA

L'architecture orientée services est une méthode de construction d'application qui utilise des services communs aux fonctions de support des entreprises.
En effet, la SOA est un concept architectural caractérisé par un ensemble de composants à la fois obligatoires et optionnels [11].

- ✓ Les producteurs (Producers en anglais) :C'est une entité qui offre un service spécifique ou fonctionnalité.
- ✓ Les consommateurs(Consumers) : Une entité qui utilise le service offert par le producteur.
- ✓ La SOA Infrastructure qui est composée en trois sous couches à savoir :

- Applications : Fournir une interface graphique et des degrés divers du logique métier étroitement couplé pour que les consommateurs exécutent leurs tâches.
- Service : Une entité qui effectue une tâche spécifique lorsqu'il est invoqué.
- Support de service : Une entité qui fournit les fonctions de support de fond pour SOA.

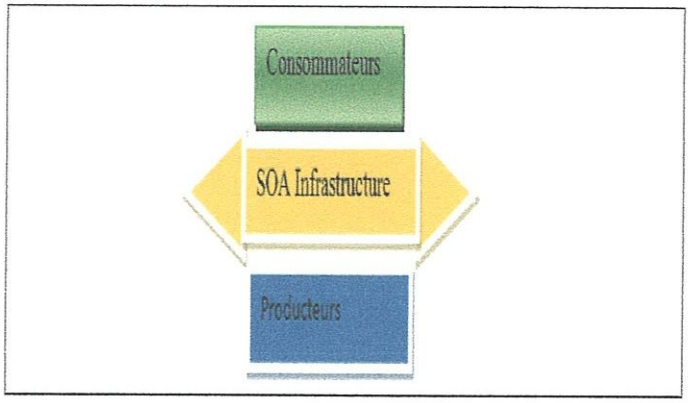


Figure 1. 3: Les trois composants de l'architecture SOA

1.4.5. Les acteurs de l' architecture SOA

Le service est le composant clé de l'architecture orientée service, il dispose de fonctionnalité bien déterminé selon les différents participants en définissant chacun leurs rôles. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe [42].

- ✓ Le fournisseur de service crée le service, puis publie son interface ainsi que les informations d'accès au service, dans un annuaire de service Web.
- ✓ L'annuaire de service rend disponible l'interface du service ainsi que ses informations d'accès, pour n'importe quel demandeur potentiel de service.
- ✓ Le client de service accède à l'annuaire de service pour effectuer une recherche afin de trouver les services désirés, ensuite, il se lie au fournisseur pour invoquer le service.

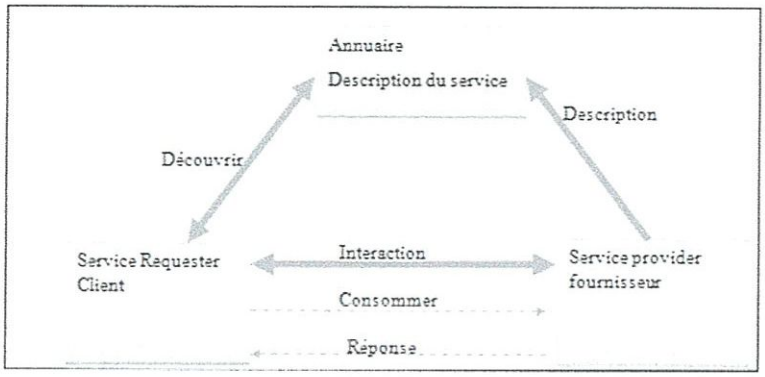


Figure 1. 4: Les acteurs de l'architecture SOA

1.4.6. Les bénéfices de l'architecture orientée service

L'architecture orientée service avec sa nature faiblement couplée permet aux entreprises de [11]:

- ✓ 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.

1.4.7. Les avantages et les inconvénients de SOA

On présente certains avantages et inconvénients de l'architecture SOA

✓ **Les avantages d'une architecture orienté services**

Les avantages d'une architecture orientée service SOA peuvent être résumés dans les points suivants [9]:

- 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).

✓ **Les inconvénients de l'architecture orientée service**

L'architecture orientée service présente certains inconvénients qui sont [11] :

- Coût de conception et de développement initiaux plus importants.
- Nécessité d'appréhender de nouvelles technologies.
- Existant non SOA dans les entreprises.
- Performances réduites pour les traitements simples (couche supplémentaire).

1.5. Technologie d'implémentation des services web

Dans ce qui suit, on va présenter un aperçu sur le principe de base d'implémentation de l'architecture SOA qui est basée sur la technologie des services web. Etant l'une des approches plus récentes dans le domaine d'informatique, le concept de services web est aussi l'une des technologies la plus populaire pour mettre en œuvre une architecture orientée service. Il est en grande partie basé sur la technologie XML (eXtencible Marcup Langage) ; ce dernier permet aux services web de gérer l'interopérabilité entre les applications hétérogènes.

1.5.1. Définition des services web

Selon la société W3C (World Wide Web Consortium) : Un service web est une application conçue pour permettre l'interopérabilité entre les machines d'interagir sur réseau. Il dispose d'une interface décrite dans un format machine (spécifiquement WSDL). Les autres systèmes d'interagissent avec le service web de la manière prescrite par sa description en utilisant des messages SOAP, typiquement transmise via HTTP avec une sérialisation XML en conjonction avec d'autres technologies liées à des normes [10].

1.5.2. XML (eXtensible Markup Language)

C'est un langage de balisage, un langage de programmation qui sert à enregistrer des données textuelles. Ce langage a été standardisé par W3C en février 1998 et actuellement très populaire. Il est aussi un langage de base pour les services web car grâce à ce langage les services web sont indépendants de plateformes et langages de développements, ce qui permet leur interopérabilité (voir figure 1.5) [14].

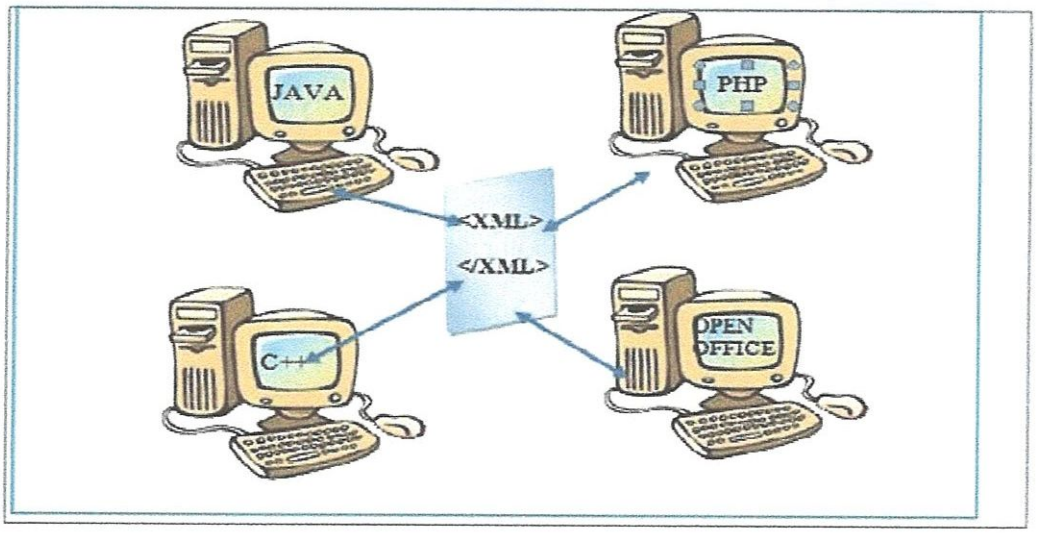


Figure 1. 5: Interaction entre les applications hétérogènes à travers XML

1.5.3. SOAP (Simple Object Access Protocol)

C'est un protocole qui fournit un moyen de communication entre applications. Il assure l'échange de messages structurés entre ces applications dans un environnement centralisé. Il utilise un format XML pour la spécification des messages, pour cette raison il est indépendant de tout langage de programmation ou système d'exploitation. Il utilise aussi le protocole HTTP qui fournit la meilleure façon de communication entre les applications car il ne pose pas de problème de sécurité grâce à les firewalls et le serveur proxy. Ce protocole est supporté par tous les serveurs et les navigateurs Internet. Le protocole SOAP à une structure à respecter [15], voir figure 1.6.

La structure d'un message SOAP est composée des éléments suivants [16] :

- ✓ Une enveloppe : qui définit le document XML comme un message SOAP.
- ✓ Un entête optionnel pour stocker les informations métier spécifiques à la transaction (authentification, jeton d'autorisation, état de la transaction,…).
- ✓ Un corps contenant les données à transporter.
- ✓ Une gestionnaire d'erreur qui identifie la condition d'erreur, émis par le fournisseur de service, il est utilisé en cas d'échec d'exécution du service dans le serveur.

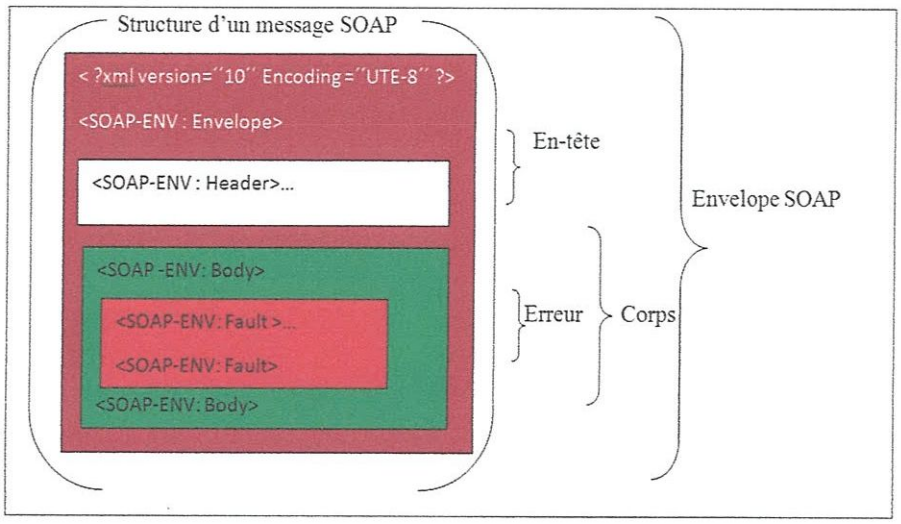


Figure 1. 6: Structure générale d'un message SOAP

1.5.4. WSDL (Web Service Description Language)

C'est un langage de représentation des interfaces de services web en XML. Ce langage est indépendant par rapport au langage de programmation et à la plateforme utilisée. En effet WSDL est la représentation XML du langage IDL (Interface Definition langage) [3]. Il joue un double rôle :

- ✓ Il sert de référence de description de l'implémentation de service.
- ✓ Il assure le couplage entre le client et le serveur par le biais des interfaces.

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

- ✓ Le groupe de définitions abstraites qui est composé de quatre éléments suivants :
  - Types : c'est l'élément qui définit les types de données utilisées dans les messages échangés par le service web.
  - Message : contient des paramètres de fonctions (les entrés séparés les sorties) ou les descriptions de document.
  - Operations : c'est l'élément qui décrit d'une manière abstraite les actions supportées par le service.
  - Types de ports : c'est l'élément qui représente un ensemble d'opérations correspondant chacune à un message entrant ou sortant

- ✓ Le groupe de descriptions concrètes qui est composé de trois éléments à savoir :
  - Liaison : définit les protocoles de communication utilisés lors des invocations du service web et aussi le format de message.
  - Port : une adresse d'accès au service.
  - Service : l'élément qui regroupe une collection de ports.

Le WSDL est un langage qui permet de décrire un service. Il est indispensable à leur déploiement, WSDL est utilisé via le standard UDDI pour faire la publication de services.

1.5.5. UDDI (Universal Description Discovery and integration)

L'annuaire UDDI permet de publier, de découvrir et de faire une recherche des informations sur une entreprise et ses services web.
La publication des informations concerne les fournisseurs de services et ces derniers doivent être spécifiés en XML afin que les recherches et l'utilisation soient faites de manière dynamique et automatique. L'annuaire UDDI facilite la localisation d'un service web.

L'annuaire UDDI possède une structure de données bien spécifique qui contient l'ensemble des informations divisée en trois parties (les pages blanches, les pages jaunes et les pages vertes chacune d'elles peut être utilisée pour effectuer une recherche via UDDI) qui doivent être décrites en XML, et par la suite définit le rôle de chaque page (voir figure 1.7).

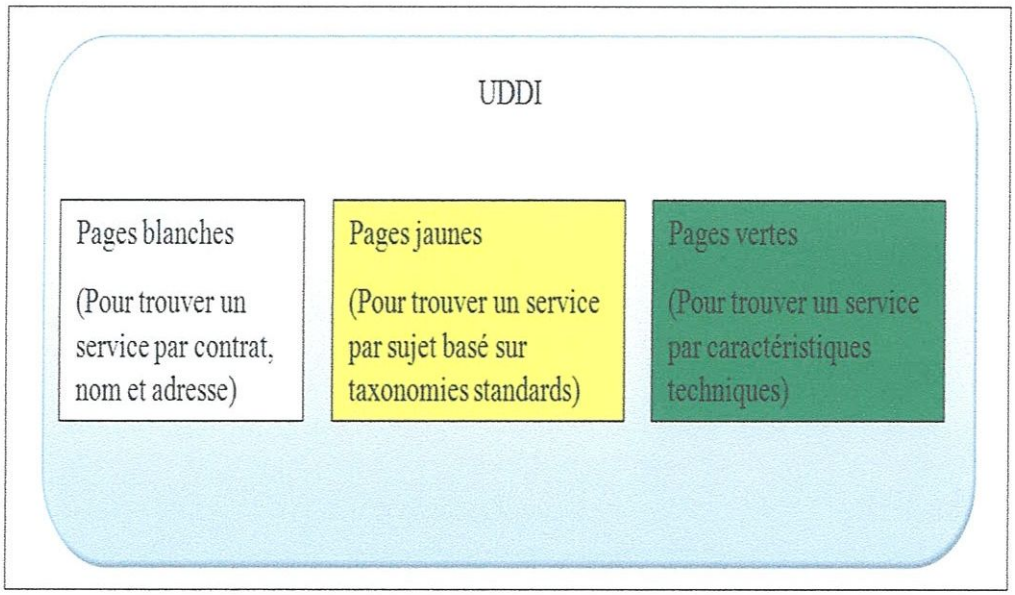


Figure 1. 7: Structure de données de l'annuaire

- ✓ Pages blanches : détiennent les informations générales sur l'entreprise.
- ✓ Pages jaunes : description détaillée du format WSDL de services web déployés.
- ✓ Pages vertes : fournissent les informations techniques détaillées sur les services web proposés.

1.5.6. Architecture des services web

L'architecture des services web fournit un prototype nécessaire à la compréhension des services web et des relations entre les standards utilisés, comme décrit dans la figure 1.8.

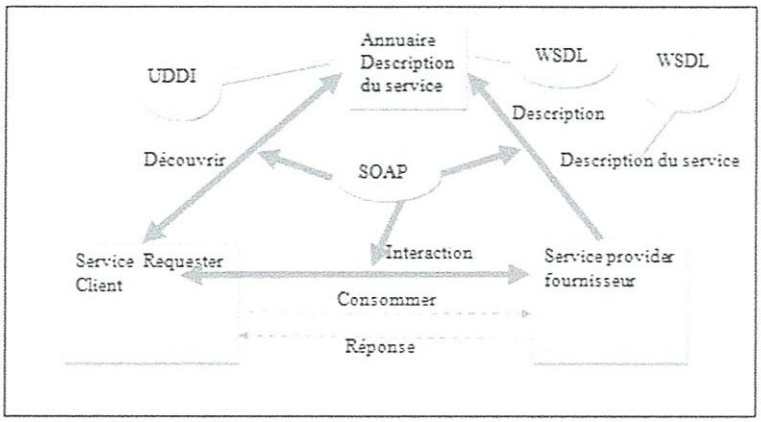


Figure 1.8: Architecture WSOA (Web Service Oriented Architecture)

Pour explique le fonctionnement de services web, on passe par les étapes inscrites ci-dessous :

- ✓ Le fournisseur de service crée le service web, il définit sa description puis le publie dans un annuaire de service web sous la forme de fichier WSDL.
- ✓ Le client accède à l'annuaire de service pour effectuer une recherche afin de trouver les services désirés. Lorsque le client détient la description de service web désirés UDDI, ce dernier envoie la référence de ce service au client pour sélectionner.
  Le client télécharge le fichier WSDL de service sélectionné depuis le fournisseur. Ensuite, le client utilise la description du service sélectionné pour récupérer les informations nécessaires qui lui permettra de se connecter au fournisseur du service et d'interagir avec lui.
- ✓ Enfin, le fournisseur envoie le résultat de l'exécution des fonctionnements du service web, ce qui termine le cycle de vie de service web.

1.6. Méthodologie de développement de services web

Il existe deux approches pour la création des services web, on cite l'approche code first et l'approche WSDL first.

1.6.1. L'approche code first

C'est la technique de création de services web la plus courante, elle consiste à inférer les interfaces de services web à partir du code source orienté service. Elle est souvent appelée « code first » ou « implémentation first » c'est-à-dire la priorité au code du service web ; car l'interface du service web décrite d'une façon formelle dans un document WSDL qui est dérivée le code de service web.

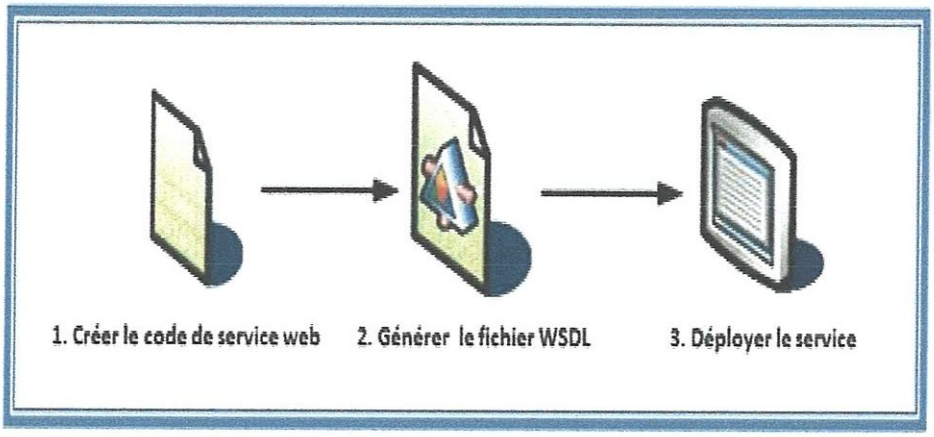


Figure 1. 9: Développement d'un service par la méthode "code first"

Cette technique permet d'abord à écrire le code du service web (étape n°1). Après la compilation, l'infrastructure des services web utilise ce code pour générer de façon dynamique un fichier WSDL (étape n°2). Puis on lance le déploiement de service (étape n°3). Lorsque les clients demandent la définition du service web, ils récupèrent le fichier WSDL généré et créent le client à partir de cette définition.

1.6.2. L'approche WSDL first

Cette approche consiste à créer d'abord le fichier WSDL est aussi appelée parfois « schéma first » (priorité au schéma) ou «contract first ».

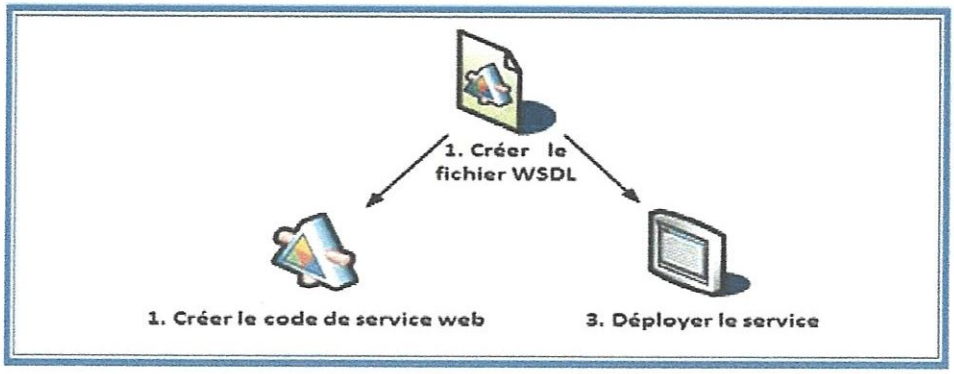


Figure 1. 10: Développement d'un service web par la méthode " WSDL first"

Cette technique permet à travailler en suivant les trois étapes décrites dans la figure ci-dessus :

- Créer le fichier WSDL.
- Créer le code du service web.
- Déployer le service web.

1.7. Les services web java

Plusieurs APIs standards existent pour la mise en œuvre et l'utilisation de services web en Java. On distingue les deux principales APIs Java en XML :

- ✓ **SAAJ (SOAP with Attachement API for Java):** Elle permet l'envoi et la réception de messages respectant les normes SOAP 1.1. Elle propose un niveau d'abstraction assez élevé permettant de simplifier l'usage de SOAP. Les classes de cette API sont regroupées dans le package javax.xml.soap. Initialement, cette API était incluse dans JAXM. Depuis la version 1.1, elles ont été séparées. SAAJ propose des classes qui encapsulent les différents éléments d'un message SOAP : (SOAPMessage, SOAPPart, SOAPEnvelope, SOAPHeader et SOAPBody). Tous les échanges de messages avec SOAP utilisent une connexion encapsulée dans la classe SOAPConection. Cette classe permet la connexion directe entre l'émetteur et le receveur des messages.
- ✓ **JAX-WS : JAX–WS (Java API for XML based web services):** Une nouvelle API qui est fortement recommandé pour les nouveaux développements. Elle propose un modèle de programmation pour produire (coté serveur) ou consommateur (coté client) des services web qui communiquent via des messages XML de type SOAP. Le but principal de cette API est de faciliter et de simplifier le développement des services web notamment grâce à l'utilisation des annotations. JAX WS fournit les spécifications pour le cœur du support des services web pour la plateforme Java SE et Java EE. Cette API repose sur plusieurs autres JSR :

- JSR 181 (Web Service MetaData for the Java Platform): propose un ensemble d'annotations qui permettent de définir les services web.
- JSR 109 et JSR 921 (Implementing Entreprise Web Services) : décrit comment déployer, gérer et accéder aux services web via un serveur d'application.

- JSR 183 (Web Services Message Security APIs) : décrit la sécurisation des messages SOAP.

1.8. Conclusion

Dans ce premier chapitre on a clarifié les concepts fondamentaux de l'architecture SOA et les techniques d'implémentation de cette architecture qui est basé sur la technologie des services web, force est de constater que ces derniers sont particulièrement adaptés à ce type d'architecture et qu'ils ont fortement contribués à son apparition. Le second chapitre présentera les notions de la réingénierie logicielle et de la migration vers SOA.

CHAPITRE 2

REINGENIERIE ET MIGRATION VERS SOA

2.1. Introduction

La réingénierie logicielle (RL) est un processus dont l'objectif principal consiste à réaliser la transformation et l'amélioration du logiciel patrimonial (cf. 2.2.1).Il permet de prolonger le cycle de vie d'un logiciel par un recyclage de ses composants. Un logiciel fiable a besoin d'une bonne compatibilité avec les technologies disponibles ainsi que les compétences des équipes de maintenance. Le logiciel résultat de la réingénierie logicielle ne doit pas subir de régressions au niveau fonctionnel mais doit gagner en performance et en évolutivité. L'obsolescence rapide des logiciels se manifeste dans un secteur en évolution permanente et augmente le coût de la maintenance. Il y a aussi une pénurie des compétences pour ce qui concerne les plus vieux logiciels. Ces anciennes applications ne sont pas adaptées au changement de technologie et perdent de ce fait leur performance.

De cet évidence, l'architecture orientée services (SOA ou Service Oriented Architecture) que nous avons introduit dans le chapitre précédent semble compatible avec la technologie actuelle et promet de le rester pour une période encore. SOA est une architecture logicielle s'appuyant sur un ensemble de services simples. Elle permet de décomposer une fonctionnalité en un ensemble de fonctions basiques appelées services fournies par des composants et décrit finement le schéma d'interaction entre ces services [17]. La migration vers SOA est devenue l'une des techniques importantes de modernisation, elle aide les organisations d'une part à réutiliser leurs anciens systèmes en leurs donnant une nouvelle vie, et d'autres part à profiter des avantages des systèmes à base de service [43].

Dans ce chapitre, on passera en revues les points essentiels de la réingénierie logicielle. Nous présenterons aussi le processus de migration vers SOA en présentant tout d'abord les bénéfices attendus d'une architecture SOA et ensuite les méthodologies de migration vers une architecture SOA et on terminera cette partie en exposant les bénéfices de migration et le problème lié à la migration. En fin nous mettons l'accent sur l'étape d'identification de services qui est une étape incontournable dans le processus de conception et de migration vers l'architecture SOA.

2.2. Réingénierie logicielle

La réingénierie de logiciels permet l'amélioration et la transformation d'un logiciel existant. Elle permet de renouveler une application de tel façon que la compréhension, le contrôle et l'utilisation de cette application puisse de nouveau être performant et évolutif. L'intérêt pour la réingénierie de logiciels a connu une croissance spectaculaire. Elle est liée au vieillissement des systèmes patrimoniaux.

Ce vieillissement concerne leur architecture, les plateformes les supportant ainsi que sur leur adaptabilité et leur stabilité face aux développements techniques requis par l'évolution des besoins.

La réingénierie s'avère nécessaire afin de rétablir et de réutiliser les ressources logiciels existants, de contrôler davantage les coûts de maintenance de logiciels souvent élevés et de créer une base solide qui permet de supporter les évolutions en besoins fonctionnels à venir [18].

2.2.1. Historique sur la réingénierie logicielle

Le concept de la réingénierie logicielle est apparu au début des années 90. L'explosion de l'information contenue dans les systèmes informatiques ayant atteint un niveau critique en termes de quantité et leur vie tirait à leur fin. Par ailleurs, les entreprises ont dû remettre en question l'exploitation de ces systèmes sur des machines (ordinateurs) centraux ou les convertir sur des architectures plus légères. Ces systèmes informatiques évoluent grâce à la maintenance ; or la mise à jour de la documentation n'ayant pas suivie, ce qui entraîne une perte de connaissance de leur fonctionnement dans l'entreprise. Avant l'apparition de la réingénierie, il existait d'abord l'ingénierie. Ce dernier concept est apparu en 1968. Dans les années 70, le nombre croissant de logiciels en exploitation nous oblige à reconnaître le fait qu'un logiciel ne meurt pas facilement. Cette époque, il y avait l'apparition de la maintenance logicielle qui acquiert sa reconnaissance. En outre, cette activité de plus en plus demandée en raison du volume de logiciels occasionne de nombreux retards.

Dans les années 80, de nouvelles architectures émergèrent menaçant de plus en plus ces « vieux » logiciels et rendant leur maintenance contraignante. A partir des années 90, on a

assisté à des changements et des améliorations remarquables tant dans le domaine du logiciel que celui de l'infrastructure.

À la même époque, grâce à l'Internet et à la fibre optique est né le concept de la réingénierie du logiciel qui devient un sujet d'étude répandu. Le bogue de l'an 2000 a lancé véritablement l'amélioration des méthodes de réingénierie du logiciel, d'où la problématique de réconception des systèmes patrimoniaux demeure entière.

2.2.2. Les systèmes patrimoniaux

Les logiciels patrimoniaux (Legacy Softwares), sont des programmes qui ont été développés avec des technologies qui sont devenues avec le temps périmées.

Brodie définit les systèmes d'information patrimoniaux comme «n'importe quel système qui résiste significativement à la modification et à l'évolution ». Les systèmes patrimoniaux incluent désormais non seulement les langages des années 1970 «Fortran, COBOL, PLI, C», mais aussi les langages de quatrième génération des années 1980 «CSP, les frames ORACLE, etc.» et même les premiers langages orientés objets des années 1990 tels que «C++, Smalltalk». Il considère que n'importe quel organisme qui a utilisé une technologie pour plus de cinq ans est concerné par un problème de logiciel patrimonial. Cela est dû au cycle de rénovation de technologie qui est passé à moins de cinq ans [35].

2.2.3. Définitions et principes de la réingénierie

Plusieurs définitions ont été proposées selon différents organismes :

- ✓ D'après Chikofsky (1990), la réingénierie logicielle est l'étude et la modification d'un système en vue de le recréer sous une nouvelle forme. Elle n'altère pas obligatoirement les fonctionnalités du logiciel mais elle peut éventuellement en créer de nouvelles [20].

- ✓ Le SWEBOK (le document de base à la normalisation en ingénierie du logiciel) la définit comme suit : C'est l'analyse d'un logiciel en vue de le recréer sous une nouvelle forme en incluant l'implémentation de cette nouvelle forme [21].

La réingénierie est souvent appliquée aux systèmes hérités ou systèmes patrimoniaux, mais ses concepts et outils sont de plus en plus populaires pour le développement de système moderne.

Certainement, la réingénierie logicielle s'appuie sur une multitude de processus, on va présenter les quatre principaux processus à savoir la recomposition, la rétro-ingénierie, la redocumentation ainsi que la restructuration dont par la suite on définira ces processus (voir figure 2.1) présenté dans [20].

- ✓ Rétro-ingénierie (Reverse Engineering en anglais)**:** C'est l'activité qui consiste à étudier un objet pour en déterminer le fonctionnement interne ou la méthode de fabrication. 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).
- ✓ Recomposition (Forward Engineering en anglais): 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.
- ✓ Redocumentation (en anglais Redocumentation)**:** C'est la modification des informations contenues dans le code et des documents spécifiques au logiciel.
- ✓ Restructuration (en anglais Restructuring): C'est la transformation d'un logiciel en un autre logiciel avec pour objectif de préserver son comportement externe sur les fonctionnalités et la sémantique.

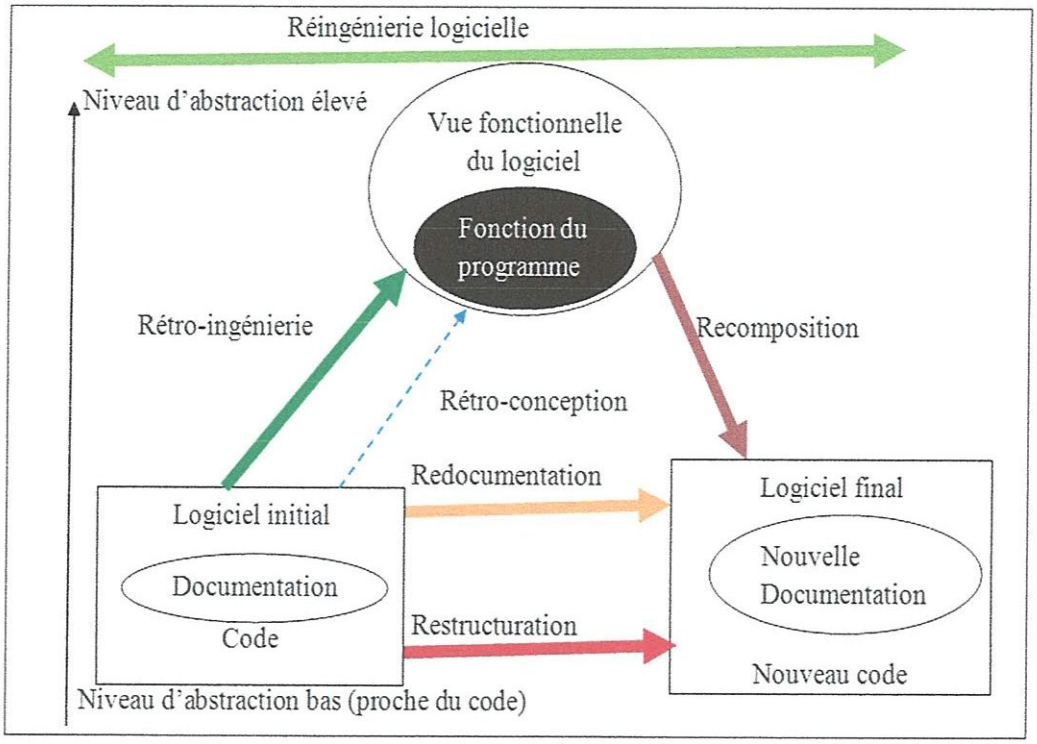


Figure 2. 1: Les principaux processus utilisés en réingénierie logicielle

2.2.4. Les avantages de la réingénierie logicielle

Les principaux avantages de la réingénierie logicielle sont les suivants :

- ✓ Réduit les risques : redévelopper un système critique introduit beaucoup des nouveaux risques.
- ✓ Réduit les coûts : redévelopper un nouveau système coûte plus cher que la modification d'un système existant.
- ✓ La baisse des coûts opérationnels et de maintenance.
- ✓ Structures de développement propriétaires pour raccourcir les cycles de développement, accroitre la productivité des développeurs et limiter les défectuosités.

2.2.5. Les caractéristiques de la réingénierie logicielle

Les principaux caractéristiques de la réingénierie logicielle comprennent :

- ✓ Des stratégies, des méthodologies, des techniques et des outils de modélisation éprouvés.
- ✓ Un modèle complet de prestation de services conçu pour réduire les délais d'exécution et les coûts.

- ✓ Des experts très expérimentés possédant une vaste expérience en gestion de projets d'envergure de la réingénierie des applications.
- ✓ Un vaste éventail d'options et d'approches de réingénierie.
- ✓ L'accès à un vaste réseau de partenaires et une objectivité complète en matière de recommandation de partenaires et de technologies.
- ✓ L'accès à un réseau d'experts spécialisés dans les domaines d'affaires, technique et fonctionnel.
- ✓ L'assurance qualité du logicielle.

2.3. Comparaison entre l'approche OO et SOA

Une architecture orientée objet est une architecture basée sur un ensemble d'entités (appelés communément objets) qui communiquent entre eux par envoie des messages. Chaque objet est caractérisé par :

- ✓ Son identité, unique et invariante, pour servir de référence à l'objet d'une manière non ambigüe.
- ✓ Son état qui est l'ensemble des valeurs, des attributs,
- ✓ Ainsi que son comportement qui définit l'ensemble des opérations (méthodes) applicables à l'objet.

Contrairement, à une architecture orientée services (SOA) est une architecture basée sur un ensemble d'entités logicielles communément appelés services. Les services sont dépourvus de la notion d'état et communiquent entre eux par envoie des messages ou requêtes. Chaque service est localisable par une adresse unique. Cette dernière est nécessaire lors de l'envoi d'une requête par le consommateur de service. A noter que tout service est une application logicielle autonome qui accepte des requêtes et qui envoie des réponses à travers une interface standard bien définie.

2.4. Le processus de migration de systèmes orienté objet

Dans la première partie de ce chapitre on a passée en bref les avantages de la réingénierie et la définition des systèmes patrimoniaux. Par la suite, nous allons exposer les atouts et limites dans le cas de migration d'une application basé sur une architecture orientée objet vers application basée sur une architecture orientée services. On va présenter les

motivations (bénéfices) qui intéressent les entreprises souhaitant la réingénierie et l'adoption de l'architecture orientée services. En fin de chapitre, on introduit les méthodologies de la migration vers une architecture orientée services.

2.4.1. Les bénéfices attendus d'une architecture orientée services

Les architecture orientées services offrent plusieurs bénéfices dans le monde TI (Information Technology), dont nous citons les plus essentiels à savoir :

- ✓ **Bénéfice de la réutilisation**

L'élimination des redondances par le partage des services procure à long terme une réduction des coûts de développement et d'assurance quand vient le temps de procéder à des changements ainsi que des économies importantes au niveau des frais d'entretien pour un service partagé. Pour effectuer des modifications ou introduire des nouvelles fonctions sur un seul service implique aussi un cycle de vie de développement, dont une capacité à répondre plus efficacement aux besoins d'affaires. Être plus compétitif en réagissant plus rapidement aux impératifs du marché est parfois difficile à calculer en terme financiers, c'est pourquoi il faut insister sur la réduction des frais de développement, de test et d'entretien qui sont plus facile à estimer [22].

- ✓ **Bénéfice du développement d'applications composites (multiservices)**

L'assemblage de services multiples pour l'introduction de nouvelles applications permet des économies substantielles en termes d'intégration. Ici l'impact financier provient autant d'une réduction des frais de développement informatique que d'une augmentation des revenus de l'entreprise entraînée par une meilleure réponse aux besoins d'affaire [22].

- ✓ **Bénéfice du couplage lâche (loosely coupled)**

Le couplage lâche donne une flexibilité qui rend l'entreprise plus agile en isolant les interventions informatiques nécessaires et en permettant un déploiement itératif. De plus, ce couplage lâche permet des économics dc budget car les ressources qui interviennent n'ont pas besoin de connaître les technologies de chaque service impliqué. Finalement, le couplage lâche facilite la connectivité interne et externe, donc un plus grand potentiel d'automatisation et d'alliances commerciales [22].

En résumé, un modèle financier de justification d'un projet SOA devrait aborder les trois éléments suivants :

- ✓ Efficacité d'affaires:
  - Plus grande agilité et meilleure réponse à la dynamique de marché.
  - Plus grande efficacité des processus.
  - Meilleur déploiement des ressources [22].
- ✓ Réduction des coûts:
  - Réduction des frais d'entretien.
  - réduction des efforts nécessaires pour supporter les changements organisationnels.
  - Choix technologiques plus flexibles étant donnés le couplage lâche des applications [22].
- ✓ Réduction des risques:
  - Niveau plus élevé de qualité de service des TI.
  - Déploiement itératif.
  - Développement plus rapide qui assure un retour sur investissement plus rapide [22].

2.4.2. Les méthodologies de migration vers SOA

Dans ce qui suit, on présente les différentes méthodologies de migration vers une architecture orientée services. Ces différentes méthodologies permettent d'évoluer les systèmes patrimoniaux vers une architecture moderne qui est l'architecture SOA, tout cela en se basant sur la réingénierie. Ainsi, une étude des différentes stratégies d'évaluations a été présentée [23].

Selon Bisbal [24] distingue trois catégories des approches d'évolutions de systèmes patrimoniaux qui sont :

- ✓ Redéveloppement: qui réécrit les applications existantes ;
- ✓ Wrapping: qui enveloppe (wraps) un composant existant dans un nouveau plus accessible ;
- ✓ Migration: qui transforme un système patrimonial en un système plus flexible tout en gardant les données et les fonctionnalités originales.

Umar [25] sépare les deux principales stratégies d'évolution des systèmes patrimoniaux vers une architecture orientée services :

- ✓ Stratégie de migration: qui consiste à modifier et restructurer un système patrimonial à fin de trouver un nouveau système dans notre cas l'architecture SOA, cela on distingue deux façons :
  - Graduelle : des parties du système sont converties l'une âpres l'autre
  - Complète (soudaine) : stratégie big bang, tout est remplacé en une fois.
- ✓ Stratégie d'intégration: il consiste à garder la structure interne d'un système patrimonial et de l'interconnecter avec l'extérieur en utilisant les services web et ESB. La technique la plus utilisée est le wrapping essaye plutôt de proposer un modèle décisionnelle, puis il offre un support pour les entreprises pour décider et opter pour une des stratégies selon leur cas précis.

Marchetto [26] met l'accent sur la migration et classifie les travaux effectués sur la migration en quatre catégories :

- ✓ Approches Orientées Métier (Business-Oriented): utilisent les informations du processus métier réalisé par l'application cible comme point de départ pour réaliser la migration ;
- ✓ Approches basées fonctionnalités (Functionality-Based): qui se concentrent principalement sur l'identification des fonctionnalités implémentés par le système cible, puis les localiser dans le code en utilisant une techniques tel que : le slicing ou feature location.
- ✓ Approches basées modèles (Model-Based) : approches qui construisent un nouveau système orientée services en partant du modèle du système cible, ce modèle est généralement extrait par l'application des techniques de retro-ingénierie en commençant par l'analyse des artefacts du système cible tels que le code, les traces d'exécution et la documentation.
- ✓ Approches basées interactions (Interaction-Based): Approches qui utilisent comme point de départ pour la migration, les interactions entre l'utilisateur et le système cible ou entre les composants systèmes tel que : les bases de données et le logique métier.

Almonaies [27] maintient la même classification (redéveloppement, wrapping, et migration) en ajoutant une quatre qui est le remplacement. Par conséquent, on discrimine quatre catégories de classification à savoir:

- ✓ Le remplacement: Il consiste à retirer complètement l'application et la remplacer carrément par une nouvelle récupérée sur étagère. Cela est faisable lorsque les fonctionnalités et les règles métiers sont bien compréhensibles et le système patrimonial est obsolète ou difficilement maintenable.
- ✓ Le redéveloppement: Réfère à l'application de l'approche de la réingénierie et de retro-ingénierie afin d'ajouter des fonctionnalités orientées services aux systèmes patrimoniaux. Selon Chikofsky la retro-ingénierie est le processus d'analyse d'un système sujet pour créer une représentation du système dans un niveau plus haut d'abstraction, « la réingénierie est-elle définie comme l'examen et la modification d'un système pour reconstituer sous une nouvelle forme ». Certes, la réingénierie peut inclure des activités telles que la retro-ingénierie, la restructuration, la réconception et la ré-implémentation. Par cet effet, on introduit les trois principaux enjeux de la réingénierie orientée services qui sont : l'identification des services, qu'on présentera à la fin de ce chapitre, le packaging des services et le déploiement des services.
- ✓ Le wrapping: fournit une nouvelle interface orientée service aux composants du système patrimonial existant, en les rendant facilement accessible par les autres composants logiciels. Il s'agit d'une technique de modernisation à boîte noire puisqu'elle se concentre sur l'interface de l'ancien système tout en cachant la complexité de son fonctionnement interne. Cette stratégie est utilisée lorsque la réécriture du code existant est trop couteuse, et lorsqu'une solution rapide et économique est nécessaire. Elle peut constituer une bonne option si l'ancien système à une valeur commerciale élevée et le code est de bonne qualité. Le principal problème est que cette stratégie ne change pas les caractéristiques fondamentales des applications existantes et ne va pas résoudre les problèmes déjà présents, comme les problèmes de maintenance et de mise à niveau.
- ✓ La migration: elle comporte le redéveloppement et le wrapping. La distinction de migration de redéveloppement et wrapping n'est toujours pas évident car le terme de migration est utilisé en référence de toute approche qui déplace le système patrimonial entier dans un nouvel environnement.

En plus des quatre catégories citées par Almonaies, il adressé un ensemble de critères qui permettent de comparer les différentes approches d'évolution vers une architecture orientée services:

- ✓ La stratégie de modernisation adoptée: (migration, remplacement, redéveloppement ou wrapping) ;
- ✓ Le type du système patrimonial à faire évoluer: procédural, orientée objet, code binaire exécutable, etc.
- ✓ Le degré de complexité de l'approche d'évolution: temps, coût, complexité de la méthode ;
- ✓ La profondeur de l'analyse du système patrimonial: superficielle, profonde ;
- ✓ Adaptabilité du processus d'évolution: le processus s'adapte t'il bien au système patrimonial pour minimiser l'étendue des modifications nécessaires ;
- ✓ Support d'outils: A quelle dégrée le processus est automatisé, et est-ce que l'outil est développé ou proposé ;
- ✓ Le degré de convergence: est-ce que l'approche présente une stratégie complète pour évoluer vers une SOA, ou juste des spécifications de modernisation.
- ✓ Validation et maturité: est-ce que l'approche proposée a été appliquée et validée avec succès sur un nombre suffisant de cas ou est-ce qu'elle reste dans le cadre d'idées techniques ? est ce que une technique commerciale prouvée, etc.

2.4.3. Bénéfices de la migration

La migration offre plusieurs bénéfices aux entreprises et à leurs services, qui sont les suivants [28] :

- ✓ L'adaptation aux nouveaux besoins,
- ✓ L'amélioration des services clients,
- ✓ Une intégration plus étroite avec les partenaires et les fournisseurs,
- ✓ La réduction des coûts 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,
- ✓ L'élimination de la dépendance forte à l'ensemble des anciennes compétences.

2.4.4. Problèmes liés à la migration

On présente les différentes difficultés de migration de systèmes d'information d'entreprise, cela est dû à plusieurs facteurs qui sont :

- ✓ La complexité des systèmes patrimoniaux: Elle est considérée comme le plus grand limiteur du processus de migration, elle est produite à cause de la taille énorme du système, de l'incompréhensibilité des systèmes à migrer et des phases successives de maintenance déjà réalisé.
- ✓ Les risques de migration: elles ne sont pas majeures, 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. En outre, prévenir de l'insuffisance de compréhension, de gestion des risques et des techniques de réductions du risque deviennent difficile [28].

2.5. Approches d'identification de services

Comme les définitions (cf. Chapitre 1) donnent des services comme éléments moteur de l'architecture SOA, il faut donc les identifier afin de s'assurer une bonne migration. Les applications orientées objets semble être mieux, la meilleure source de migration parce qu'ils possèdent quelques propriétés communes avec les applications SOA : comme la modularité (ensemble des fonctionnalités qui font sens), l'abstraction et l'encapsulation.

La suite du chapitre consiste à présenter les approches et démarches théoriques applicables à l'identification.

2.5.1. Définition d'identification de services

L'identification de services est une phase cruciale pour la migration vers une architecture SOA. Cette phase consiste à détecter les services.

Elle décompose les processus d'affaire qui ont été préalablement modélisés afin d'identifier les services. Dans ce cas, on peut citer deux types de services; les services d'affaires qui sont associés à des tâches d'affaires et les services applicatifs qui communiquent avec les applications qui supportent les processus [18].

2.5.2. Les différentes approches d'identification de services

Dans ce paragraphe on va présenter en bref les approches d'identification des services. Nous citons l'approche Top down, Bottom up et outside in représenté dans la figure 2.2, presenté dans [29].

✓ Approche top down

Cette approche suit une logique descendante. Elle consiste à commencer l'intégration par la définition des processus métiers et passer graduellement à la définition des services métiers nécessaire au fonctionnement des processus. Le but principal de cette logique est de poursuivre l'alignement du système d'information sur le métier de l'entreprise. Elle permet de piloter l'architecture par les besoins métier en minimisant les redondances des services. Toutefois les coûts engendrés par cette approche sont importants dans la mesure où elle nécessite une refonte du système d'information.

✓ Approche bottom up

Elle opère dans le sens inverse de la précédente approche c'est-à-dire elle suit une logique ascendante. Elle est réalisée dans les entreprises possédant déjà un système d'information. La conception se déroule de manière ascendante en commençant par identifier des fonctions existantes du SI qu'on peut l'utiliser en tant que service, puisque, on se base sur un système déjà conçu alors l'entreprise peut gagner un gain sur les coûts d'implémentation. La seconde approche est meilleur marché que le top down. En effet, elle consiste à une réutilisation différente d'une fonction déjà existante d'un système, ce qui ne permet pas une autonomie considérable des services qui restent encore très dépendants de leur système de base.

✓ Approche middle in ou outside in

Elle permet de combiner les deux premières approches. Elle permet de développer les approches « top down » et « bottom up » de manière parallèles pour obtenir des résultats judicieux, qui permettront de déterminer la manière optimale pour définir les processus métiers. Bien qu'il y a de risque au moment de la réunion des deux approches. En fin, cette approche offre un pilotage efficace de la SOA et favorise la réutilisation des services.

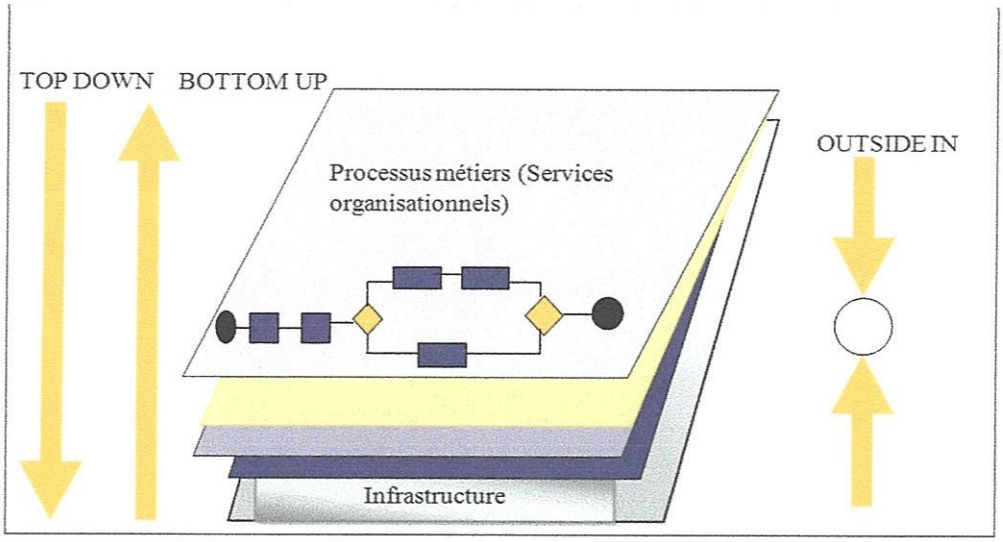


Figure 2. 2: Les approches d'identification de services web

2.5.3. Les critères de choix des approches

Le choix entre ces approches dépend de certains critères décisifs [29], tel que :

- ✓ L'état des systèmes informatiques existants (nature, degré de complexité, capacités fonctionnelles et techniques).
- ✓ Les objectifs attendus de la migration.
- ✓ Les capacités des groupes de travail.

2.5.4. Problèmes liés à l'identification des services

Un des problèmes centraux pour mettre en œuvre une architecture orientée services est lié à l'identification des services à savoir:

- ✓ La granularité des services est fondamentale: détermine en grande partie la réutilisabilité des services.
- ✓ Éviter une granularité trop fine qui entraine: beaucoup d'interaction et des problèmes de performance.
- ✓ Attention à une granularité trop épaisse.
- ✓ Un service qui fait trop de chose, risque de ne pas être réutilisable.

2.6. Démarche d'identification

Il n'existe probablement pas de démarche universelle pour l'identification des services. Le contexte de l'entreprise, les méthodes ou les modèles d'urbanisations utilisés sont autant de facteurs qui devront être pris en compte. Naturellement, ces démarches types n'excluent pas (et ne sont pas exhaustives). Dans la réalité, le système se constitue par une articulation de ces différentes démarches, par consolidations successives, en parallèle avec la vision globale de l'architecture, presenté dans [19].

2.6.1. Démarche orientée processus

La démarche orientée processus s'appuie sur une analyse des processus métiers de l'entreprise (ou d'un domaine particulier), dans l'objectif de déployer des composants de service de type processus. Les principales activités sont les suivants:

- ✓ Cartographie des processus métiers: Inventaire des processus métiers, avec leurs propriétés et leurs liens. Cette cartographie reste au niveau macro, sans détailler le déroulement de chacun des processus.
- ✓ Identification des composants processus: Détermination des processus métiers à automatisés. On privilège les processus à forte valeur ajoutée pour l'entreprise (on évite les processus internes comme la gestion du personnel) et les plus évolutifs, de façon à aboutir à un réel apport métier et un retour sur investissement notable.
- ✓ Elaboration des modèles de processus métier: Ces modèles représentent l'enchaînement des processus métiers (activités, flux, acteurs) avec un point de vue maîtrise d'ouvrage. Les flux échangés par les activités établissent une première structure des types de données d'échange utilisés.
- ✓ Modèles de processus exécutable: Le passage dc modèles métiers aux modèles exécutables nécessite la description détaillé de tous les éléments (l'ensemble de chemins, les erreurs, les compensations éventuelles), et de préciser les services consommés par de processus.

2.6.2. Démarche orientée données

Cette démarche aboutit à la mise en place de composants de type entité, avec la définition des types de données d'échange associés. Elle assure un accès banalisé aux informations gérées par les bases de données.

- ✓ Identification des objets métiers clé: Cette identification s'appuie fortement sur la connaissance fonctionnelle. Chaque objet métier clé donne lieu à un composant de service de type entité, qui fournit les services d'accès (création, modification, suppression et requête).
- ✓ Définition des types de données d'échanges: Pour chaque objet métier clé (et donc pour chaque composant de type entité), un ou plusieurs formats d'échange est défini. Ce format est naturellement basé sur les modèles des objets métiers, et consiste à découper celui-ci sous la forme de format de type document XML, qui seront échangés lors de l'invocation des opérations de services.
- ✓ Modèle des objets métiers: Ce modèle représente la structure et les propriétés des objets métiers. Typiquement, on utilise le formalisme de classe UML, en s'appuyant sur les structures propres de bases de données existantes c'est-à-dire le diagramme de classe.

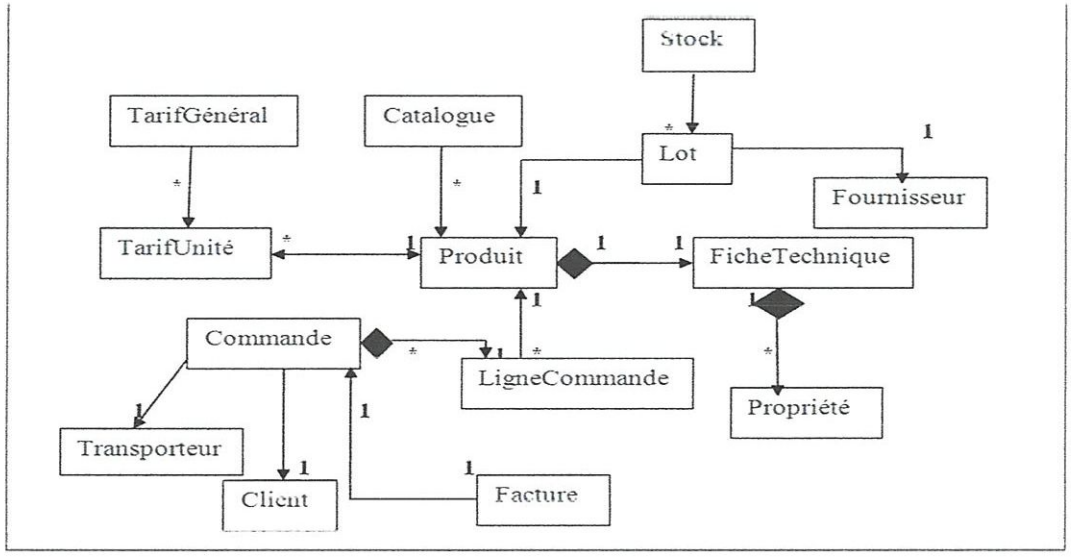


Figure 2. 3: Modèle des objets métier (notion UML)

- ✓ Branchement de composant: Par leur nature particulière, les composants de types entité obligent à une modification dans les accès à l'information. En effet, ces

composants sont par définition d'unique canal vers les données persistantes : les autres formes d'accès doivent donc être remaniées (par exemple les lectures directes vers les bases de données).

2.6.3. Démarche orientée application

L'objectif de cette démarche est la restructuration des certains publications par mutualisation des services.

- ✓ Cartographie des applications: C'est un modèle des applications existantes intégrant notamment les échanges de flux inter applicatifs.
- ✓ Identification des services mutualisés: L'analyse de flux et leurs caractéristiques (volume, fréquence) signalent les applications fortement dépendantes. De plus, de connaissance plus détaillée des applications peut faire apparaître des redondances fonctionnelles ou des duplications historiques. Ces éléments se conjuguent pour isoler les services potentiellement mutualisables, et d'obtenir une découpe plus simple et plus cohérente, voir figure 2.4.
  - ✓ Restructuration: Les composants de services doivent apparaître comme des éléments stables, avec un consensus d'interprétation sans entraîner une refonte complète du domaine. Dans le cas contraire, une restructuration globale est souvent préférable, avec une mise à plat de l'analyse métier. Les composants de services résultants sont généralement de type fonction, proches de l'utilisation métier.

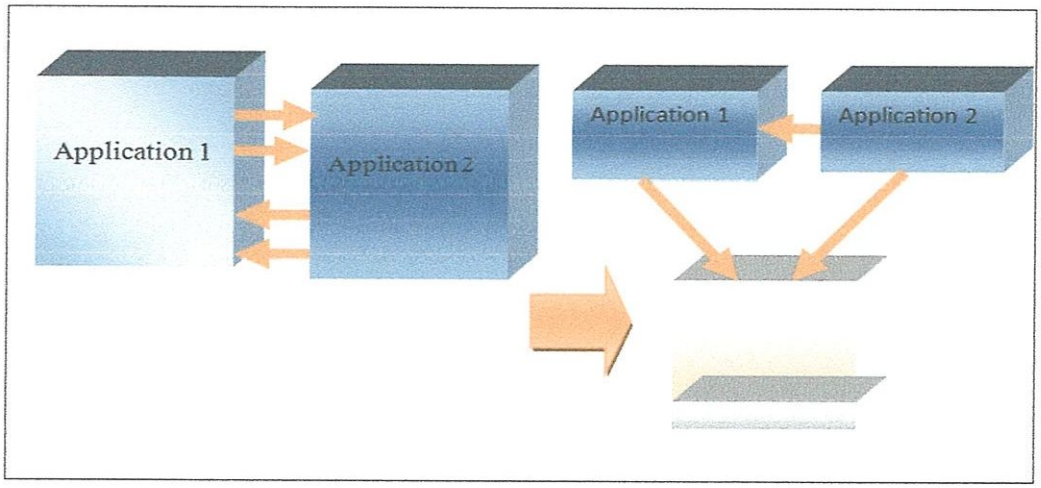


Figure 2. 4: Restructuration par mutualisation des services

2.7. Conclusion

La réingénierie logicielle s'applique aux systèmes patrimoniaux afin de leur donner une nouvelle vie. En effet, la migration de ces systèmes est vue comme une étape cruciale qui favorise les entreprises du point de vue économique et développement. Les approches d'identification de services que nous avons vu ne correspondent pas toutes aux besoins de la migration. Nous garderons l'approche bottom up parce qu'elle peut s'appliquer dans le cas où il y a un système d'information d'où son rôle dans la réingénierie des systèmes patrimoniaux L'identification de services Web avec la méthode bottom up permet d'aider aux concepteurs à déterminer les fonctionnalités existantes dans le système patrimonial et d'extraire les parties réutilisables. On peut appliquer aussi la méthode mixte pour ajouter de nouvelles fonctionnalités.

La migration vers une architecture SOA implique des modifications fondamentales dans l'organisation de l'entreprise en général.

A la suite d'identification de services, nous nous focaliserons dans le chapitre suivant sur les concepts de la classification ascendante hiérarchique, qui va nous montrer la manière dont on peut regrouper ce dernier.

CHAPITRE 3

CLASSIFICATION HIERARCHIQUE

3.1. Introduction

La classification est une méthode qui a pour but de regrouper les individus d'une population en des classes disjointes telles que les membres d'une même classe soient similaires et les individus dans des classes différentes soient dissimilaires.

Cette méthode est composée de deux types qui sont la classification non supervisée et la classification supervisée [30] [31].

- ✓ La classification supervisée: Etant donné un certain nombre de classes prédéfinies et un ensemble d'individus, il s'agit de trouver la meilleure classe à laquelle chaque individu doit appartenir.
- ✓ La classification non supervisée: Etant donné un ensemble d'individus, il s'agit de structurer des classes pas encore identifiées qui regroupent ces individus.

L'algorithme de classification hiérarchique, fait partie de la catégorie des classifications non supervisées. Nous avons choisi cette méthode puisqu'elle nous permet de créer des ensembles de classes d'une application orientée objet qui peut être proposées comme des services respectant les principes de l'architecture orientée services. La création de ces ensembles ainsi que l'identification des services est guidée par les propriétés de qualité logicielle des services, ainsi la fonction de similarité/dissimilarité est proposée de façon à ce que les classes qui peuvent former le meilleur ensemble respectant les critères d'un bon service soient regroupées.

A travers ce chapitre nous présenterons tout d'abord les concepts fondamentaux de la classification hiérarchique à savoir l'algorithme utilisé ainsi que toutes les notions associées à son implémentation. Ces dernières correspondent aux variables ou propriétés des individus à regrouper, l'indice de similarité ou dissimilarité, les liens d'agrégation et la représentation des individus par un dendrogramme. L'adaptation de ce dernier à notre problématique fera l'objet du chapitre suivant.

3.2. Fonctionnement de l'algorithme

L'algorithme suit un ensemble d'étapes que nous pouvons classer comme suit :

- ✓ Choix de variables : chaque individu est caractérisé par un ou plusieurs propriétés pouvant varier et avoir une unité de mesure.
- ✓ Choix d'indice de dissimilarité (similarité) entre les individus à classer, distance ou lien entre individus.
- ✓ Choix de lien d'agrégation entre les classes d'individus déjà identifiées.
- ✓ Dendrogramme à retenir qui respecte au mieux l'objectif.

3.3. Présentation de l'algorithme

L'algorithme hiérarchique génère une répartition de l'espace des données, mais aussi une succession de partition de l'espace des données Celle-ci est souvent représentée sous la forme d'un dendrogramme. Un dendrogramme est un arbre binaire de partitions successives de l'espace des données. La méthode sera dite respectivement descendante (division) ou ascendante (agglomération) selon que l'on parcourt le dendrogramme de haut en bas ou du bas en haut.

- ✓ La méthode hiérarchique descendante commence avec un groupe contenant tous les individus. Puis de procéder à une division du groupe en plus petit groupe jusqu'à ce que chaque groupe ne contienne plus qu'un exemple.

- ✓ La méthode hiérarchique ascendante, on commence avec autant de classe qu'il y a d'individu. Ensuite, à chaque étape, on regroupe les deux classes qui sont jugées les plus similaires pour terminer avec un seul grand cluster englobant tous les individus.

  Dans notre cas, nous allons utiliser la méthode ascendante. Cet algorithme de classification ascendante hiérarchique permet de regrouper incrémentalement les classes qui sont les plus proches jusqu'à obtenir un groupe les contenants toutes. Un algorithme qui résume les différentes étapes de classification hiérarchique est présenté ci-dessous [32][34][45].

1. Placer chaque individu dans un cluster
2. Calculer la matrice triangulaire des distances entre toutes les classes
3. Tant que il reste au moins deux clusters à regrouper faire
4. Rechercher les clusters avec la distance la plus petite
5. Fusionner les deux clusters
6. Recalculer la nouvelle matrice des distances en remplacement les éléments agrégés par le nouveau cluster
7. Fin tant que
8. Fin.

Figure 3. 1: Algorithme ascendante hiérarchique

3.4. Indice de dissimilarité

Dans ce paragraphe nous exposons quelques distances de similarité ou de dissimilarité. Pour que deux éléments d'un groupe soient similaires il faut utiliser une mesure qui s'applique sur ces derniers. De nombreuses mesures de la distance entre individus ont été proposées. Le choix d'une (ou plusieurs) d'entre elles dépend des données qu'on veut regrouper et certaines distances sont rappelées dans le cas d'analyse de données (qualitatives et/ou quantitatives) [33].

- ✓ Distance euclidienne : C'est probablement le type de distance le plus couramment utilisé. Il s'agit simplement d'une distance géométrique dans un espace multidimensionnel.

$$d(I_i, I_j) = \sqrt{\sum_k (X_{ik} - X_{jk})^2}$$

- ✓ Distance euclidienne carrée: On élève la distance euclidienne standard au carré afin de surpondérer les objets atypiques (éloignés).

$$d(I_i, I_j) = \sum_k (X_{ik} - X_{jk})^2$$

- ✓ Distance du City-block (Manhattan): Cette distance est simplement la somme des différences entre les dimensions. Cette mesure produit des résultats proches de ceux obtenus par la distance euclidienne simple.

$$d(I_i, I_j) = \sum_k |X_{ik} - X_{jk}|$$

- ✓ Distance de Tchebychev: Cette mesure de distance est adoptée lorsque nous considérons deux individus comme étant différents à partir du moment où ils sont différents sur l'une des dimensions.

$$d(I_i, I_j) = Max|X_{ik} - X_{jk}|$$

- ✓ Distance en puissance: Nous pouvons parfois souhaiter augmenter ou diminuer la pondération progressive associée à des dimensions pour lesquelles les individus respectifs sont très différents.

$$d(I_i, I_j) = (\sum_k |X_{ik} - X_{jk}|^p)^{1/r}$$

- ✓ Distance Percent disagreement: Cette mesure est particulièrement utile si les données des dimensions utilisées dans l'analyse sont de nature catégorielle (abstraite).

$$d(I_i, I_j) = \frac{Nombre\ de\ X_{ik} \neq X_{jk}}{K}$$

- ✓ Distance 1-r de Pearson: Cette mesure est calculée à partir du coefficient de corrélation, à l'aide de la formule.

$$d(I_i, I_j) = 1 - r_{ij}$$

En fait, un indice de dissimilarité doit aussi satisfaire les conditions suivantes :

- ✓ Non négativité: $d(I_i, I_j) \geq 0$
- ✓ Symétrie: $d(I_i, I_j) = d(I_j, I_i)$
- ✓ Normalisation: $d(I_i, I_i) = 0$

Un indice de dissimilarité est une vraie distance, s'il vérifie également l'inégalité triangulaire : $d(I_i, I_j) \leq d(I_i, I_k) + d(I_k, I_j)$

3.5. Liens ou indices d'agrégation

L'application de l'algorithme de classification hiérarchique suppose également que nous fassions le choix d'une distance entre les ensembles d'individus (clusters). Le concept de lien d'agrégation propose des solutions qui permettent de calculer la distance entre classes quelconques sans refaire les calculs de distances qui existent entre les individus composant chaque classe [33]. Ces solutions s'appellent règles d'agrégation. Il existe plusieurs critères pour calculer les nouvelles distances entre le nouveau cluster et les anciens. Nous pouvons citer :

- ✓ Lien simple (Single link): il définit la distance entre deux classes comme étant la plus petite distance entre toutes les paires de concepts des deux classes.

$$D(A,B) = min_{I\in A} min_{J\in B} d(I,J)$$

- ✓ Lien complet (Complete link): il définit la distance entre deux classes comme étant la plus grande distance entre toutes les paires de concepts de deux classes.

$$D(A,B) = max_{I\in A} max_{J\in B} d(I,J)$$

- ✓ Lien moyen (Average link): il définit la distance entre deux classes en faisant intervenir tous les objets présents dans ces classes.

$$D(A,B) = \frac{1}{n_A n_B} \sum_{I\in A, J\in B} d(I,J)$$

- ✓ Lien moyen de groupe (Group average link): il définit la distance entre deux classes comme étant la distance entre les centres des classes.

$$D(A,B) - \frac{1}{(n_A + n_B)(n_A + n_B - 1)} \sum_{I,J\in A\cup B} d(I,J)$$

3.6. Dendrogramme

La classification hiérarchique ascendante fournit une hiérarchique binaire de clusters ce qu'on appelle dendrogramme. La figure 3.2 montre un dendrogramme obtenu après classification de six individus. Ce dendrogramme dépend des choix de variables utilisées pour classer les individus, le type de distance appliqué aux variables et aussi le type de lien d'agrégation, plusieurs dendrogrammes sont possibles pour les mêmes individus.

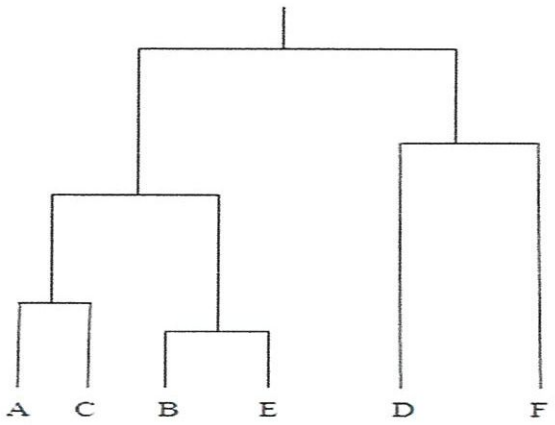


Figure 3. 2: Exemple d'un dendrogramme

3.7. Conclusion

Nous avons exposé dans ce chapitre les techniques fondamentales de l'algorithme de classification hiérarchique. Ces techniques sont radicalement exploratoires [34]. Les résultats varient en fonction des trois paramètres qui sont les variables participant à la classification, le type de distance de similarité ainsi que le type d'agrégation appliqués aux classes.

Notre intérêt pour cet algorithme est motivé par notre problématique qui consiste à former des groupes de classes à partir d'une application orientée objet qui peut être identifiées comme des services qui respectent les principes d'une architecture orientée service. Le chapitre suivant détaillera comment nous avons utilisé et adapté l'algorithme de regroupement hiérarchique, pour l'identification de services.

CHAPITRE 4

CONCEPTION

4.1. Introduction

L'intérêt au développement des applications à base de services ne cesse de croitre ces dernières années. Nous avons déjà présenté dans les chapitres précédents les motivations et les avantages du passage vers la SOA. Elle aide les entreprises d'une part à réutiliser leurs anciens systèmes existants en leurs donnant une nouvelle vie et d'autre part à profiter des avantages des systèmes à base de services [43].

Un des problèmes qui se posent dans ce domaine, est comment récupérer des applications existantes, fiables et toujours en cours d'utilisation, malheureusement développées avec d'anciennes technologies (appelées applications patrimoniales) et les faire évoluer vers une architecture SOA.

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

Nous avons choisi la réingénierie des applications orientées objet (OO) qui sont prédisposés à évoluer vers des systèmes orientées service puisque les applications OO se basent sur le principe de modularité, d'encapsulation, d'abstraction et de réutilisabilité qui sont repris par la SOA.

L'identification de service est une phase critique dans l'ingénierie des services. Notamment dans la conception à base d'une architecture SOA. La position, le sens et les activités d'identification de service diffèrent d'un contexte à un autre.

Dans le processus d'ingénierie logicielle orientée service, pour réaliser les exigences et besoins de l'entreprise, l'identification de service prend place dans la phase de spécification et d'analyse. Dans cette approche top-down d'identification de service, les exigences opérationnelles sont analysées et modélisées de manière à identifier les services d'affaires.

Cependant dans notre cas de la réingénierie et la migration vers SOA, l'identification de service est effectuée dans la phase de la retro-ingénierie des systèmes existants. C'est une approche Bottom-up qui démarre de l'analyse du code source pour extraire les services candidats qui respectent les caractéristiques de qualité d'une architecture SOA.

Afin d'automatiser l'identification de services à partir du code source, nous avons choisi d'utiliser et d'adapter l'algorithme de classification hiérarchique afin de regrouper les classes de l'application OO dans des ensembles qui formerons ultérieurement des services tout en respectant les caractéristiques principales d'une architecture orientée service qui sont l'autonomie, la composition, le couplage lâche, l'abstraction et la réutilisabilité.

4.2. Objectif

L'objectif principal de notre étude est d'automatiser au maximum la phase d'identification des services dans le processus de réingénierie d'une application OO vers une architecture SOA.

Afin de réaliser ce but, nous avons proposé une approche et de réaliser un outil basés sur l'utilisation de l'algorithme de classification hiérarchique. Trois étapes principales ont été fixées, à savoir:

- ✓ Analyser les codes sources: la première étape consiste à extraire toutes les informations qui permettent la réalisation du regroupement. Ces informations sont les classes Java et leurs interdépendances, c'est-à-dire les interactions entre elles.
- ✓ Regroupement: La seconde étape consiste au regroupement des classes en appliquant l'algorithme de regroupement hiérarchique.
- ✓ Enfin, l'identification des services candidats en appliquant l'algorithme de parcours d'identification.

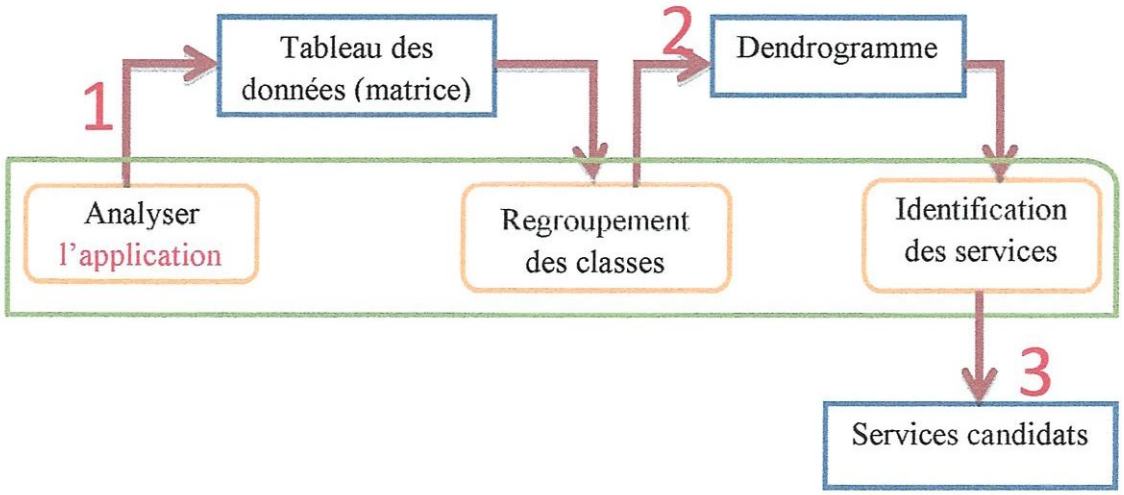


Figure 4. 1: Vue globale de l'application

4.3. Adaptation de l'algorithme de classification hiérarchique (ACH)

Dans le chapitre précèdent, nous avons présenté les grandes lignes de la classification hiérarchique ascendante. Nous avons dû adapter ce dernier selon le contexte propre à notre problématique.

Afin d'utiliser l'ACH, nous avons besoin de spécifier la population d'individus à regrouper ainsi que les variables et critères qui permettent d'exprimer et de mesurer la similarité (par défaut les différent types de distances) entre individus.

Dans notre cas les individus sont les composants de base d'une application OO objet qui sont les classes. Nous voulons les regrouper dans des ensembles qui formeront des services qui offrent des fonctionnalités homogènes.

Donc le critère de regroupement que nous allons utiliser ne seront pas les distances traditionnelles, mais une fonction qui permettra d'exprimer au mieux la similarité entre les classes du même groupe.

Nous voulons avoir à la fin un ensemble de classes représentant un service de qualité, ainsi, le groupe doit respecter les propriétés de qualité d'un bon service, savoir ***l'autonomie, la réutilisabilité, la Composabilité, la spécificité*** (caractéristique qui permet de vérifier que l'entité encapsule une fonctionnalité).

Pour rassembler les classes qui respectent les propriétés citées ci-dessous, nous avons besoin de rechercher les métriques qui peuvent les mesurer. Nous avons donc opter pour notre recherche celles de couplage et de cohésion. Le couplage défini prends les relations d'héritage et par conséquent la réutilisation.

Dans notre cas un service sera formé d'un ensemble de classes qui offrent des fonctionnalités homogènes, qui seront fortement couplés et cohésives entre eux et faiblement couplés avec leur environnement extérieur.

A partir de l'application OO nous allons récupérer les variables qui permettent de calculer ces deux métriques (couplage et cohésion), puis proposer de les combiner dans une fonction 'objectif' qui va permettre de calculer la similarité entre les différentes classes.

4.3.1. Structure d'une application orientée objet

Le module d'analyse permet d'extraire les informations sur les classes qui composent le projet. Du point de vue code source, nous trouvons les informations utiles dans les instructions simples et composées. Par instruction composée nous voulons dire les boucles (for, while), les instructions conditionnelles (if), les choix (switch), les blocs d'instructions de capture des exceptions, etc. Les instructions simples sont les affectations, les appels de méthodes, création d'instance, etc., selon la grammaire du langage.

En plus des instructions, les concepts de réutilisation par héritage et surcharge des méthodes (polymorphisme) offrent d'autres relations utiles car les méthodes d'une classe héritées sont accessibles sans qu'on crée une instance. Nous pouvons donc représenter un projet orienté objet comme ci-dessous.

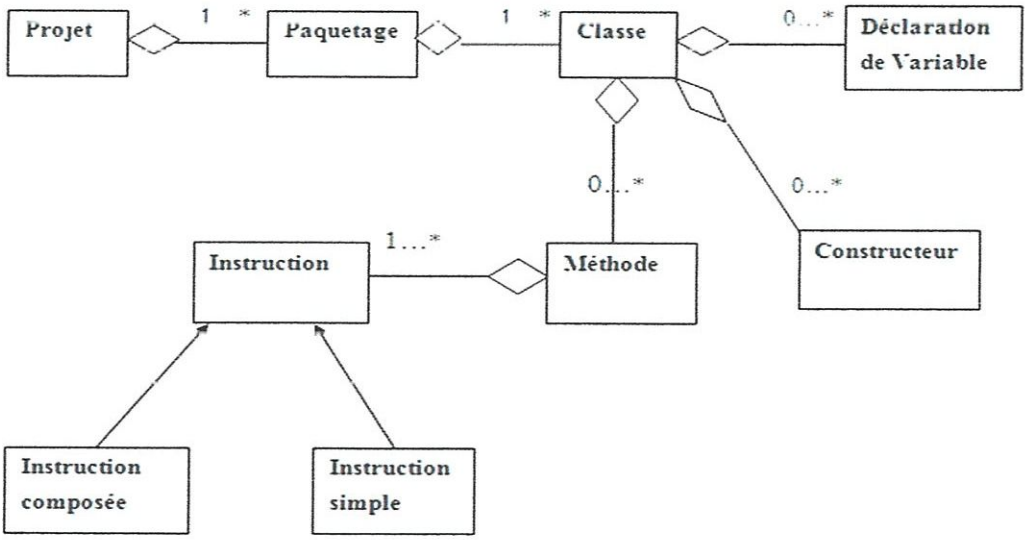


Figure 4. 2: Structure d'une application OO à analyser

Comme notre étude porte sur les classes. Les propriétés sont précisément celles de l'application. En effet, les propriétés de qualités de logiciels orientés objet liées à la réutilisabilité, l'autonomie, la composition, le couplage lâche et l'abstraction. Ces mêmes propriétés existent pour les SOA ce qui nous permet donc de penser les utiliser pour l'identification des services. Elles sont calculables en appliquant les métriques logicielles. Nous nous intéressons particulièrement aux métriques de couplage et de cohésion qui favorisent l'autonomie, la composition et la réutilisabilité.

4.3.2. Métriques de couplage

Le couplage a été défini, dans les approches traditionnelles, comme une mesure de l'interdépendance entre les différents modules d'une application [36].

Dans notre cas le couplage nous donne une idée sur l'autonomie, la réutilisabilité, l'encapsulation du groupe.

Nous verrons trois types de couplage qui presenté dans [40]:

- ✓ Couplage par interaction qui se réfère à l'existence de relation causé par des échanges de messages et des appels de méthodes.
- ✓ Couplage par composant qui montre l'existence de relation par la définition de type abstrait des données (ADT), soit comme attribut (variable globale), soit paramètre des méthodes ou par variable au sein des méthodes.
- ✓ Couplage par héritage qui se traduit par le concept d'héritage d'une classe.

Vu que le couplage se calcule entre les deux classes distinctes; sa valeur sera nulle pour la classe elle-même: $Couplage(A,B) = \begin{cases} x = o, \ si \ \ B \ n'apparait \ pas \ comme \ type \ \ dans \ A \\ x > 0, \ \ sinon \end{cases}$

Les informations qui nous permettent de calculer le couplage de la classe A par rapport à la classe B pour l'approche orientée objet sont dans les appels de méthodes, les types hérités (extends, implements), les paramètres d'appels et ainsi que les variables. Nous obtenons comme résultats n vecteurs dont la taille dépend du nombre de classes.

Supposons qu'on a un projet qui contient les classes A, B, C et D. La classe A sollicite 21 fois la classe B, 29 fois la classe C, 50 la classe E et jamais la classe F, on aura le vecteur représenté dans le tableau 4.1, ainsi de suite pour les autres classes. Si nous souhaitons comparer A et B sans aucune modification sur ces valeurs nous pouvons dire que le *couplage*(A,C) équivaut au *couplage*(B,C). Il est donc nécessaire d'utiliser un rapport donnant l'importance à la valeur proportionnellement au couplage d'une classe vers son environnement. Ce rapport permet d'avoir des valeurs proportionnelles au nombre total de relation participant au couplage. Cinq interactions parmi quinze aura la même importance qu'une interaction parmi trois. Nous verrons dans le tableau 4.2 l'avantage de ce rapport sur les nombres totaux d'une classe par rapport à son environnement. Ce rapport sera calculé par l'algorithme décrit par la figure 4.3.

Couplage	A	B	C	D	E	F
A	0	21	29	0	50	0
B	39	0	23	0	38	0
C	28	0	0	14	17	40
D	25	8	17	0	5	0
E	20	40	0	20	0	0
F	40	10	50	0	0	0

Tableau 4. 1 : Données brutes du couplage avant traitement

Voici un algorithme qui prend en entrée les données brutes et en sortie le tableau 1*n de réel, voici les résultats en appliquant l'algorithme figure 4.3, on obtient, le tableau suivant :

$Somme_A$ =0+21+29+0+50+0=100

A [1]=0/100=0 ;

B [2]= 21 /100=0.21, de même pour les autres, ligne numéro 1.

Couplage	A	B	C	D	E	F
A	0	0.21	0.29	0	0.5	0
B	0.39	0	0.23	0	0.38	0
C	0.28	0	0	0.14	0,17	0.4
D	0,25	0.08	0.17	0	0.5	0
E	0.2	0.4	0	0.2	0	0
F	0.4	0.1	0.5	0	0	0

Tableau 4. 2 : Données couplage après traitement

1. Entrée A : Tableau [n] entier
2. Sortie B : Tableau [n] réel ∈ [0,1]
3. Début
4. $Somme = \sum_i A[i]$
5. Pour i allant de 0 à n-1 faire
6. $B[i] = {}^{A[i]}/_{somme}$
7. Fin pour
8. Fin

Figure 4. 3: Algorithme de transformation de couplage

4.3.3. Métriques de cohésion

La cohésion a été définie, dans les cas des approches traditionnelles, comme une mesure du degré de liaison fonctionnelle des élements d'un module [36].

Dans notre cas la cohésion nous donne une idée sur l'homogénéité et la spécificité du groupe et permet de raffiner le choix des groupes déjà fortement couplés.

La métrique de cohésion peut être mesurée de deux façons différentes à savoir : Les relations directes et les relations indirctes [37].

✓ **Relations directes**

Deux méthodes M_i et M_j peuvent être directement connectées de plusieurs 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 (relation IM invocation de méthodes), ou partagent au moins un objet passé comme argument (relation CO paramètres objet en commun). Donc, deux méthodes peuvent être directement connectées par un ou plusieurs critères [37].

$$UA_{Mi} \cap UA_{Mj} \neq \emptyset \; ou \; IM_{Mi} \cap IM_{Mj} \neq \emptyset \; ou \; UCO_{Mi} \cap UCO_{Mj} \neq \emptyset$$

Considérant une classe C avec PUM = $\{M_1, M_2, \ldots, M_n\}$ l'ensemble des méthodes, le nombre maximum de paires de méthodes est:

$$Nbr = \frac{n * (\mathrm{n} - 1)}{2}$$

Soit un graphe non dirigé G_D dont les sommets sont les méthodes de la classe C. Il y'a un arc entre deux sommets si les méthodes correspondantes sont directement reliées. Soit E_D le nombre d'arcs dans le graphe. Le degré de cohésion dans la classe basé sur la relation directe entre ses méthodes est defini par:

$$DC_D = \frac{|E_D|}{n * (\mathrm{n} - 1)/2} \in [0,1] \quad (1)$$

DC_D donne le pourcentage de paires de méthodes, qui sont directement reliées. La métrique LCC_D (*Loose Class Cohesion*) de la classe C est ainsi donnée par :

$$LCC_D = 1 - DC_D \in [0,1] \; (1')$$

✓ **Relation indirecte**

Deux méthodes M_i et M_j peuvent être indirectement connectées si elles sont directement ou indirectement reliées à une méthode M_k. La relation indirecte est la fermeture transitive de la relation directe [37].Elles utilisent ce concept pour identifier les méthodes reliées indirectement. Ainsi, une méthode M_1 est indirectement connectée à une méthode M_k s'il y a une séquence de méthodes M_1, M_2,…, M_k telle que M_i est directement connectée à M_{i+1} (i=1, k-1). Considérons maintenant un graphe indirect G_I, où les sommets sont les méthodes de la classe C. Il y a un arc entre deux sommets si les méthodes correspondantes sont directement ou indirectement reliées (fermeture transitive du graphe G_D). Soit E_I le nombre d'arcs indirect dans le graphe. Le degré de cohésion dans la classe C basé sur les relations directes ou indirectes entre les méthodes est défini comme suit :

$$DC_I = \frac{|E_I|}{n * (n-1)/2} \in [0,1] (2)$$

DC_I donne les pourcentages de pair de méthodes, qui sont directement ou indirectement reliées. La métrique LCC_I de la classe C est ainsi défini par:

$$LCC_I = 1 - DC_I \in [0,1] \ (2')$$

Après avoir évalué les relations directes et indirectes des méthodes dans la classe C, la formule de la cohésion de cette classe est définie par :

$$DC_C = DC_D + DC_I = \frac{|E_D|}{[n * \frac{(n-1)}{2}]} + \frac{|E_I|}{[n * \frac{(n-1)}{2}]}$$

Par conséquent la cohésion de la classe C est :

$$Cohésion(C) = {|E_D| + |E_I|}\Big/{[n * \frac{(n-1)}{2}]} \in [0,1].$$

Les modules considérées pour calculer la cohésion sont les variables globales, les méthodes ou des constructeurs d'objet. Nous considérons les constructeurs comme méthodes afin d'integrer les relations indirectes qui existent entre les méthodes suites aux appels declarés dans les constructeurs. Malgré que les constructeurs ne font qu'initialiser les variables, ils peuvent comporter des instructions de toutes sortes. Nous preferons les integrer pour le calcul de cohésion afin d'ameliorer cette derniére.

✓ **Relation indirecte**

Deux méthodes M_i et M_j peuvent être indirectement connectées si elles sont directement ou indirectement reliées à une méthode M_k. La relation indirecte est la fermeture transitive de la relation directe [37].Elles utilisent ce concept pour identifier les méthodes reliées indirectement. Ainsi, une méthode M_1 est indirectement connectée à une méthode M_k s'il y a une séquence de méthodes M_1, M_2,…, M_k telle que M_i est directement connectée à M_{i+1} (i=1, k-1). Considérons maintenant un graphe indirect G_I, où les sommets sont les méthodes de la classe C. Il y a un arc entre deux sommets si les méthodes correspondantes sont directement ou indirectement reliées (fermeture transitive du graphe G_D). Soit E_I le nombre d'arcs indirect dans le graphe. Le degré de cohésion dans la classe C basé sur les relations directes ou indirectes entre les méthodes est défini comme suit :

$$DC_I = \frac{|E_I|}{n * (\mathrm{n}-1)/2} \in [0,1](2)$$

DC_I donne les pourcentages de pair de méthodes, qui sont directement ou indirectement reliées. La métrique LCC_I de la classe C est ainsi défini par:

$$LCC_I = 1 - DC_I \in [0,1] \ (2')$$

Après avoir évalué les relations directes et indirectes des méthodes dans la classe C, la formule de la cohésion de cette classe est définie par :

$$LCC_C = LCC_D + LCC_I = \frac{|E_D|}{[n * \frac{(n-1)}{2}]} + \frac{|E_I|}{[n * \frac{(n-1)}{2}]}$$

Par conséquent la cohésion de la classe C est :

$$Cohésion(C) - |E_D| + |E_I| \Big/ [n * \frac{(n-1)}{2}] \subset [0,1].$$

Les modules considérées pour calculer la cohésion sont les variables globales, les méthodes ou des constructeurs d'objet. Nous considérons les constructeurs comme méthodes afin d'integrer les relations indirectes qui existent entre les méthodes suites aux appels declarés dans les constructeurs. Malgré que les constructeurs ne font qu'initialiser les variables, ils peuvent comporter des instructions de toutes sortes. Nous preferons les integrer pour le calcul de cohésion afin d'ameliorer cette derniére.

La cohésion sera representé sous forme d'un tableau à n dimension qui depend du projet selectioné, voir tableau ci-dessous.

A	B	C	D	E	F
0.5	0.7	0.8	1	0.2	0.7

Tableau 4. 3 : Données de la cohésion

Après avoir défini les métriques de couplage et de cohésion, nous allons maintenant présenter comment les calculer.

4.3.4. Comment calculer ces metriques?

Les outils de mesure de métriques que nous avons rencontrés dans nos recherches ne nous offrent pas la possibilité de prendre les valeurs résultats comme variable d'entrée pour notre outil d'identification. En plus leur implémentation ne s'applique pas pour la plupart aux classes java mais aux classes compilées (fichier binaire .class). Nous proposons donc un outil d'analyse d'application composé principalement de trois modules afin de mesurer les métriques selon les définitions précédentes.

- ✓ Module d'extraction des instructions
- ✓ Module de calcul de couplage
- ✓ Module de calcul de cohésion

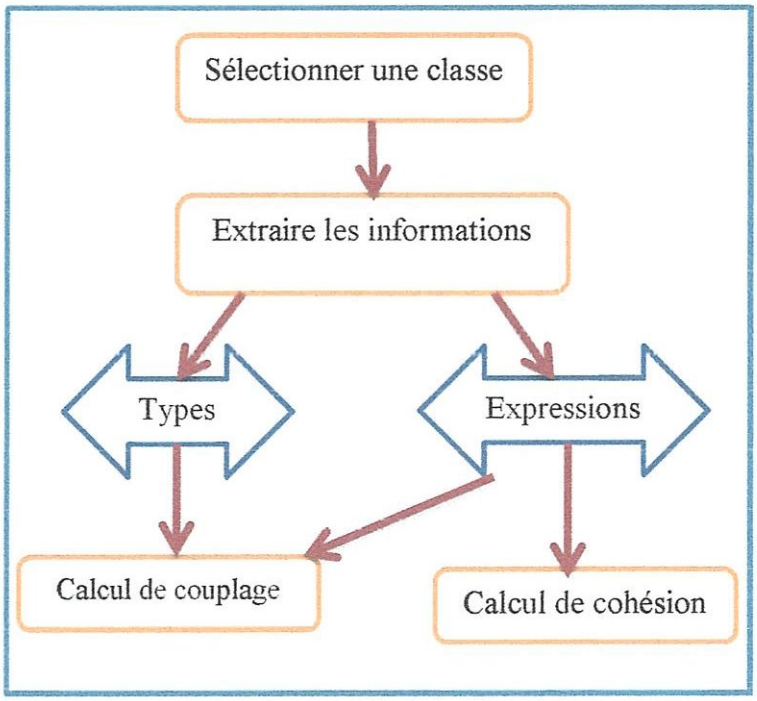


Figure 4. 4: Module d'analyse

Par la suite nous présentons la proposition d'une fonction 'objectif' permettant l'application de l'algorithme de regroupement hiérarchique. Cette fonction joue le rôle de distance de similarité.

4.4. Définition de la fonction 'objectif'

Nous avons défini les caractéristiques pour chaque classe que nous allons maintenant exploiter. Chaque projet a un nombre donné de fichiers sources, en fonction de la taille du projet. Le module d'analyse nous donne pour chaque classe, en sortie le couplage sous forme d'un vecteur, et une seule valeur pour la cohésion. Ces valeurs servent pour le regroupement de ces classes.

- ✓ **Distance entre classes** : Nous avons procédé par le test des distances existantes pour chercher une distance de similarité qui prend en compte l'hypothèse ci-dessous.
  Soient les classes A et B, leur distance est grande dans les cas suivant
  - Une classe A hérite de la classe B ou appelle les méthodes déclarées dans la classe B.
  - si leur couplage est élevé et la moyenne de leur cohésion est élevée.

  Malheureusement aucune des distances existantes ne s'est avérée appropriée à notre cas. Ce qui nous a menés à proposer notre propre distance sous forme de fonction.

- ✓ Proposition : Soit une fonction ***S*** définie ci-dessous, elle prend le couplage et la cohésion de deux éléments et retourne une valeur, selon notre hypothèse, qui traduit la similarité attendue. Soient A et B deux classes, la formule de notre fonction est définie par :

$$S(A,B) = \frac{1}{[Couplage(A,B) + Couplage(B,A) + Cohésion(A) + Cohésion(B)}$$

La fonction 'objectif' **S** que nous avons proposée, suppose que la plus petite valeur dans la matrice triangulaire signifie la ***similarité élevé***.

- ✓ **Choix de l'agrégation:**

  Les deux classes les plus proches seront remplacées suivant une agrégation avec la valeur moyenne ou minimale. En comparant les deux choix nous constatons une

amélioration en appliquant l'agrégation moyenne: la classe la plus proche des deux sera ajoutée au cluster mais la valeur minimale prend en considération seulement une classe au détriment de l'autre. Nous verrons par la suite une illustration à travers un exemple.

4.5. Algorithme de classification hiérarchique ascendant (ACHA)

L'algorithme suivant permet de réaliser les regroupements des entités classes par l'algorithme de regroupement hiérarchique ascendant suivant [38].

1. Clustering_hiérarchique(fichiers code) : Arbre dendro ;
2. Classes = extraction_information(code) ;
3. Clusters = classes ;
4. Tant que |Clusters|> 1
5. (C1,C2) = clusterProche (clusters) ;
6. C3= Clusters(C1,C2) ;
7. enleve(C1,clusters) ;
8. enleve(C2,clusters) ;
9. ajoute(C3,clusters) ;
10. dendro(0,clusters) ;
11. retour dendro ;

Figure 4. 5: Algorithme de regroupement hiérarchique

L'algorithme reçoit en entrée des clusters dont les données sont dans une matrice triangulaire ; on applique la fonction *clusterProche* pour sélectionner les deux éléments les plus proches. Cette fonction clusterProche dépend du type d'agrégation choisie et ces Types dépendent des données à traiter. Les éléments sélectionnés sont regroupés et remplacés par un nouvel élément (cluster) qui est créé en appliquant le type d'agrégation.

Le résultat de l'application de l'**ACHA** est une représentation graphique appelée dendrogramme qui nous donne une idée sur la répartition des classes suivant le degré de

similarité. Malheureusement les groupes ne sont pas encore formés, cela peut se faire de manière manuelle en analysant le dendrogramme.

Dans notre approche, nous voulons automatiser au maximum l'identification des groupes afin de les proposer comme services candidat. Dans ce qui suit, nous allons présenter comment identifier les classes qui peuvent jouer le rôle de bon services.

4.6. Identification des services candidats

Apres l'application de l'ACHA qui nous donne le dendrogramme, nous pouvons lancer une recherche sur ce dernier. Cette étape permet de regrouper les classes qui peuvent former un bon service candidat.

4.6.1. Les caractéristiques d'un service candidat

Les caractéristiques à prendre en compte pour les services candidats sont les suivantes :

- ✓ Il a des entrées et des sorties claires.
- ✓ Il est utilisé pour calculer un résultat donné.
- ✓ Il respecte la notion de couplage forte et de cohésion élevée à l'intérieur du service et du couplage lâche vers l'extérieur.
- ✓ Il a un degré d'autonomie et de réutilisation.

L'ensemble des services choisis sont ensuite bien spécifiés en définissant pour chaque service les entrées et les sorties, ainsi que les informations de qualité attendue [39].

4.6.2. Algorithme d'identification

L'algorithme suivant prend en entrée un dendrogramme et en sortie donne les classes qui peuvent être un bon service. Cet algorithme s'appelle l'algorithme de partitionnement ou de sélection [39].

L'algorithme de sélection des nœuds suivant consiste en un parcours en profondeur du dendrogramme (paramètre *dendro*). A chaque nœud, on compare le résultat de la fonction objectif de ce nœud avec celle de ces deux fils (ligne 8). Si le résultat de la fonction objectif de ce nœud est inférieur à la moyenne de ses deux fils, alors l'algorithme traite le premier fils

(ligne 11, 12 et 5), sinon le nœud en question est identifié comme un service et ajouté à la partition (variable R, ligne 9) et l'algorithme traite le nœud suivant [39].

1. *Selection_clusters (arbre dendro) : Partition R ;*
2. *Pile parcoursClusters ;*
3. *empile (racine (dendro), parcoursClusters) ;*
4. *tant que ! vide (parcoursClusters)*
5. *Cluster pere =depile (parcours Clusters)*
6. *Cluster f1 =fils1 (pere, dendro) ;*
7. *Cluster f2 = fils2 (pere, dendro) ;*
8. *Si S (pere) > moyenne(S (f1, f2))*
9. *ajouter (pere, R) ;*
10. *Sinon*
11. *empile (f1, parcoursClusters) ;*
12. *empile (f2, parcoursClusters) ;*
13. *retour R ;*

Figure 4. 6: Algorithme de partitionnement de dendrogramme

4.7. Exemple théorique

L'exemple suivant illustre le principe de l'algorithme de classification hiérarchique et d'identification.

- ✓ Soit la matrice de distance comportant cinq individus (classes orientées objet). Cette matrice a été calculée à partir de données de couplage Tableau 4.2 et cohésion Tableau 4.3 définit dans le paragraphe (cf.4.3.2). Nous appliquons la fonction objectif sur les données afin d'établir une matrice triangulaire qui sera utilisée comme entrée de l'algorithme de regroupement hiérarchique. Cette première étape consiste à rechercher la plus petite distance dans la matrice. Deux agrégations sont appliquées sur la matrice, agrégation minimale et moyenne. Celle-ci se situe entre les classes A et C.

$$S(A,B) = \frac{1}{0.21 + 0.39 + 0.5 + 0.7} = 0.55$$

	A	B	C	D	E	F
A	0	0.55	0.53	0.51	0.68	0.625
B	0.55	0	0.57	0.54	0.56	0.66
C	0.53	0.57	0	0.44	0.85	0.41
D	0.51	0.54	0.44	0	0.64	0.58
E	0.68	0.56	0.85	0.64	0	1.11
F	0.625	0.66	0.41	0.58	1.11	0

Tableau 4. 3 : Choix de la plus petite distance dans la matrice des distances

✓ La seconde étape consiste à créer une nouvelle matrice dans laquelle les anciens clusters A et C sont supprimés. En revanche un nouveau cluster M_1 contenant A et C est ajouté. Il s'agit maintenant de calculer la distance entre le nouveau cluster et les anciens. Pour cela, en utilisant le lien simple, il faut calculer la distance minimale entre A et C et les autres clusters. Le résultat de ce calcul est :

Agrégation minimale						Agrégation moyenne				
	A	B	CF	D	E	A	B	CF	D	E
A	0	0.55	0.53	0.51	0.68	0	0.55	0.57	0.51	0.68
B	0.55	0	0.57	0.54	0.56	0.55	0	0.62	0.54	0.56
CF	0.53	0.57	0	0.44	0.85	0.57	0.62	0	0.516	0.98
D	0.51	0.54	0.44	0	0.64	0.51	0.54	0.516	0	0.64
E	0.68	0.56	0.85	0.64	0	0.68	0.56	0.98	0.64	0

Tableau 4. 4 : Nouvelle matrice suite prenant en compte le nouveau cluster M_1

✓ L'opération précédente est répétée tant qu'il reste au moins deux clusters. Cela donne les matrices suivantes.

Agrégation minimale					Agrégation moyenne				
	A	B	CFD	E	AD	B	CF	E	
A	0	0.55	0.51	0.68	0	0.549	0.548	0.669	AD
B	0.55	0	0.54	0.56	0.549	0	0.62	0.56	B
CFD	0.51	0.54	0	0.64	0.548	0.62	O	0 .98	CF
E	0.68	0.56	0.64	0	0.669	0.56	0.98	0	E

Tableau 4. 5 : Matrice réalisée lors de la seconde itération

✓ L'opération précédente est répétée au moins deux clusters. Cela donne la matrice suivante.

Agrégation minimale					Agrégation moyenne		
	CFDA	B	E		A et D, C et F	B	E
CFDA	0	0.54	0.64	A et D, C et F	0	0.586	0.826
B	0.54	0	0.56	B	0.586	0	0.56

E	0.64	0.56	0		E	0.826	0.56	0

Tableau 4. 6 : Matrice réalisée lors de la troisième itération

Agrégation minimale				Agrégation moyenne		
	CFDAB	E			A et D, C et F	B et E
CFDAB	0	0.56		A et D, C et F	0	0.705
E	0.56	0		B et F	0.705	0

Tableau 4. 7 : Matrice réalisée lors de la quatrième itération

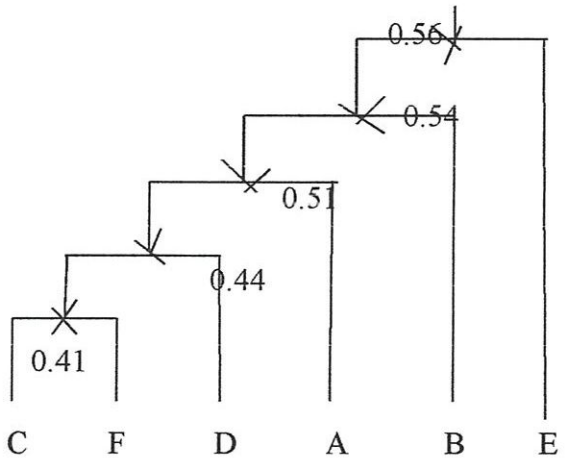


Figure 4. 7: Dendrogramme pour l'agrégation minimale

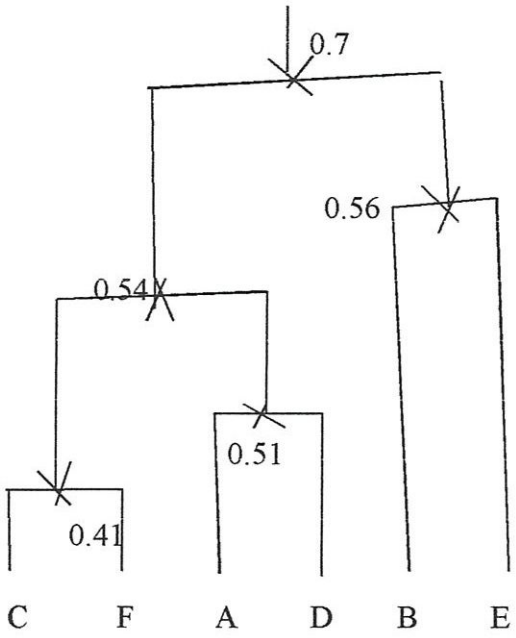


Figure 4.8 : Dendrogramme pour l'agrégation moyenne

Intérpretation: Avec l'agrégation moyenne,l'algorithme de classifiction hiérarchique donne de meilleur résultats par rapport à l'agregation minimale.

Dans de nombreuses applications utilisant l'algorithme de classification hiérarchique comme la classification d'espèces animales, de structure chimique,..., les experts doivent sélectionner manuellement le niveau le plus adéquat de regroupement dans le dendrogramme pour faciliter leur tâche de recherche. Ainsi ils fournissent une hiérarchie de profondeur moindre par rapport au dendrogramme initial. Il faut donc contrôler ce paramètre afin d'afficher des clusters avec une profondeur maximale définie a priori [32].

4.8. Conclusion

A travers ce chapitre nous avons présenté le cadre auquel appartient notre étude puis la problématique que nous avons choisi de traiter. Les objectifs ont été mis en avant et qui consiste à automatiser l'identification des services web à partir d'une application OO. Nous avons montré le fonctionnement de notre outil et spécifié les méthodes choisies, afin de le mettre en œuvre. A noter que toutes les méthodes utilisées dans ce chapitre sont validées d'une manière empirique.

Le prochain chapitre sert à présenter les techniques utilisées pour implémenter l'application conçue dans ce chapitre ainsi qu'une étude de cas sur un exemple concret.

CHAPITRE 5

IMPLEMENTATION

5.1. Introduction

L'objectif de ce chapitre est de mettre en application les objectifs du projet. Au niveau de ce chapitre, nous allons décrire les différents aspects techniques liés à l'implémentation de notre outil. Nous rappelons que l'objectif est l'identification des services à travers l'analyse du code orientée objet et regroupement des classes fortement couplées et fortement cohésives sous forme de services de bonne qualité.

Nous commencerons par la présentation de l'environnement logiciel sur lequel l'application a été réalisée. Ensuite, nous présenterons un exemple concret montrant le fonctionnement et les interfaces graphiques de notre outil.

5.2. Les outils de développement

Dans cette première partie, nous présentons les outils logiciels qui nous ont permis de réaliser notre application.

5.2.1. Environnement de programmation

Nous avons préféré le langage Java. C'est un langage bien connu à l'heure actuel du fait de son succès planétaire. Il implémente le concept oriente objet.

Conçue par James Gosling en 1994 chez Sun, c'est un langage portable qui ne dépend pas d'une plateforme donnée. Il peut être utilisé sous n'importe quel système d'exploitation à savoir Windows, Macintosh, Linux, etc. Java est donc un langage multiplateforme qui permet aux développeurs d'écrire un code qu'ils peuvent exécuter sur divers plateformes.

5.2.2. L'EDI Eclipse pour Java

C'est un environnement de référence pour la programmation orientée objet. Il est riche par sa bibliothèque contient des fonctions divers telles que les fonctions standards, le système de gestion des fichiers, les fonctions multimédia et beaucoup d'autres fonctionnalités.

C'est la plateforme utilisée pour réaliser notre outil.

C'est un environnement de développement intégré abrégé souvent IDE (en anglais, Integrated Developement Environement) qui fournit un éditeur de code et un ensemble d'options qui améliorent considérablement le temps d'implémentation.

C'est l'entreprise IBM qui l'a développé. IBM l'a rendu open source et depuis, son évolution est maintenant gérée par la fondation Eclipse. Il est distribué depuis Septembre 2005 sous la licence CPL (Common Public Licence).

5.2.3. L'analyseur syntaxique ASTParser

ASTParser est un analyseur syntaxique qui permet de construire à partir d'un code source java un arbre syntaxique abstrait, Abstract Syntax Tree.

AST est la façon dont le compilateur voit le code source, tous les fichiers source Java sont entièrement représentés sous forme d'arborescence de nœuds AST. Ces nœuds sont tous les sous-classes de la classe ASTNode. Chaque sous-classe est spécialisée dans un élément du langage de programmation Java. Par exemple, il y a des nœuds pour les déclarations de méthodes (MethodDeclaration), déclaration de variable (VariableDéclaration), etc.

Il a fallu plusieurs téléchargements avant de le choisir pour notre projet. Il n'est pas le seul analyseur mais sa particularité est qu'il s'applique aux fichiers sources.java. Nous avons téléchargé JDepend mais il ne s'applique qu'aux fichiers binaires.class.

Ensuite, il fallait connaitre comment l'utiliser. Il n'y a pas d'exemple ou de cours assez explicite pour son utilisation [42]. Nous avons procédé au départ par essai-erreur afin de savoir comment l'intégrer à notre projet.

ASTParser nous a été d'une grande utilité puisqu'il nous a permis de parcourir syntaxiquement les applications java et d'extraire les éléments essentiels dans le calcul des métriques utilisées dans notre fonction 'objectif'.

5.2.4. L'outil Stan

Malgré qu'il ne soit pas intégré dans notre outil, c'est une référence à ne pas omettre. Il n'est pas disponible dans une version open source. Il permet d'analyser un projet développé en Java et offre des résultats sur les métriques d'une façon graphique et paramétrable. Il fut l'outil d'analyse de projet informatique qui nous a inspiré et orienté dans nos recherches: il nous a permis de réfléchir sur la forme de solution à proposer jusqu'au choix d'utiliser ASTParser.

Il est libre en téléchargement pour les projets développés dans le langage Java de moins de 500 fichiers sources. La version professionnelle est nécessaire au-delà.

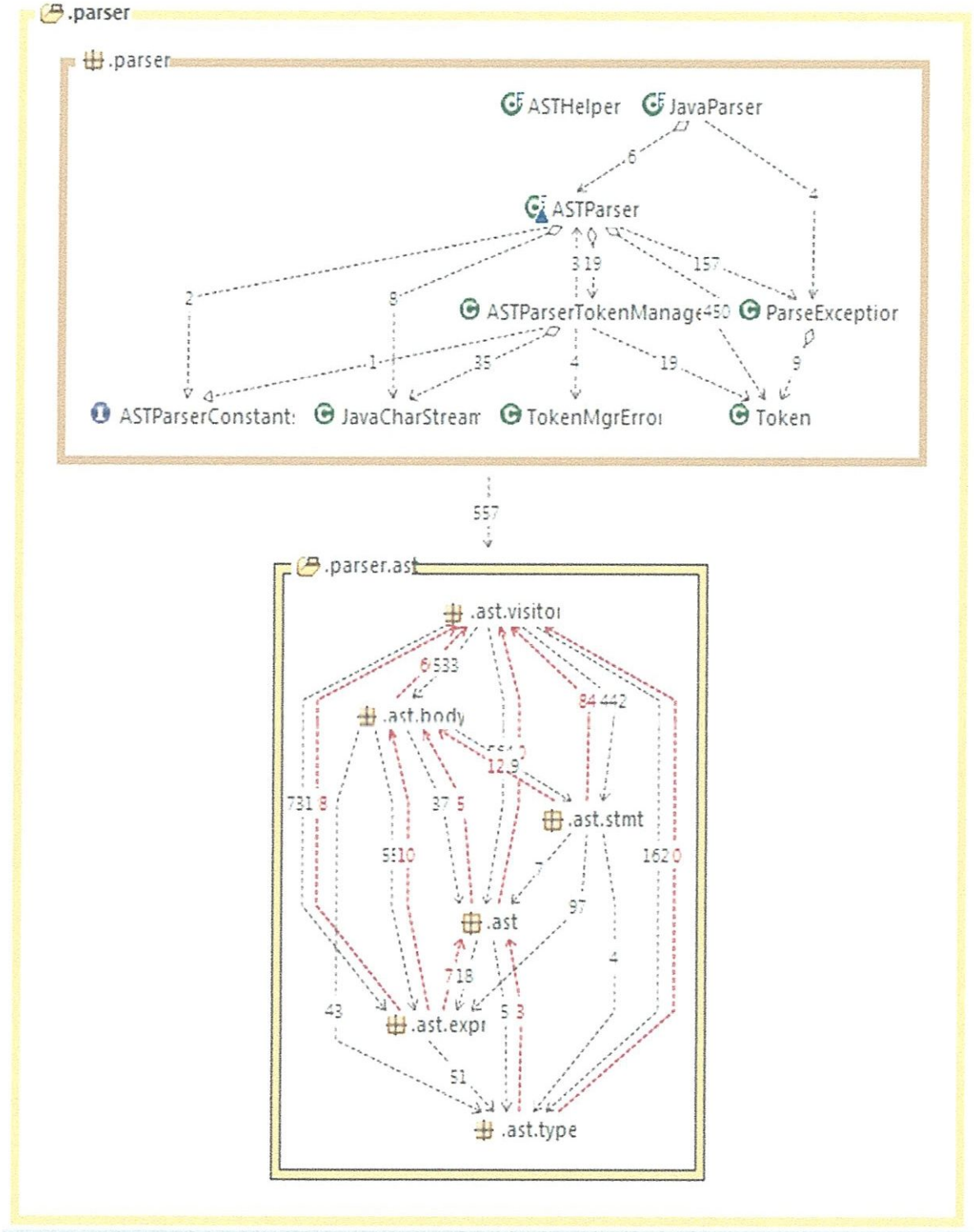


Figure 5. 1: Analyse d'un projet avec Stan

5.2.5. Présentation STATISTICA

STATISTICA est un système complet et intégré d'analyse des données, de représentation graphique, de gestion de base des données et de développement d'applications personnalisées, offrant une large gamme de procédures élémentaires ou avancées pour les sciences, le data mining et toutes les applications industrielles ou commerciales[45].

STATISTICA offre non seulement des procédures généralistes pour les statistiques, les graphiques, et la gestion analytique des données, mais également un ensemble complet de méthodes spécialisées en analyse des données (par exemple, le data mining, les sciences sociales, la recherche biomédicale, ou les applications industrielles et commerciales). Tous les outils analytiques de la gamme STATISTICA sont des composantes intégrées du logiciel. Ces outils peuvent être pilotés par divers interfaces utilisateur, notamment [45]:

Nous avons utilisé STATISTICA pour deux raisons :

- ✓ Première raison, pour vérifier si les fonctions existantes de distance de dissimilarité peuvent nous servir pour traiter nos données. Et on a constaté que ces fonctions ne nous donnent pas les meilleurs résultats, en utilisant nos données.
- ✓ Deuxième raison, comparer le résultat obtenu par la 'fonction objectif' qu'on a proposé avec celles de STATISTICA.

5.3. Architecture générale de l'application

L'architecture générale de notre outil est composée principalement des deux composants, à savoir : Le composant d'analyse et le composant d'identification des services.

- ✓ Le composant d'analyse qui contient le package Analyseur, le package couplage, le package cohésion et ASTParser. Le package analyseur prend en entrée un projet code source Java et en sortie donne une matrice brute calculée à partir du package couple, package cohésion et ASTParser. Et puis cette matrice sera traitée par la 'fonction objectif' définie dans le chapitre conception qui est une étape intermédiaire avant le regroupement et puis une matrice triangulaire sera trouvée. Cette matrice triangulaire sera utilisée comme entrée dans le composant d'identification.
- ✓ Le composant d'identification des services continue le processus. Ce composant contient le package regroupement. Ce package utilise cette matrice triangulaire, en choisissant sa valeur minimale et puis choisir l'agrégation de son choix ; soit

l'agrégation minimale ou soit l'agrégation maximale. En appliquant cette agrégation, la sortie sera un dendrogramme. Et puis, le processus continue par le package identification qui permet de chercher sur le dendrogramme les parties qui peuvent être considérées comme services potentiels. D'où la fin du cycle de notre application.

Légende :


Entrée


Sortie


Fonction Package

Analyseur ASTParser


Appliquer étape intermédiaire

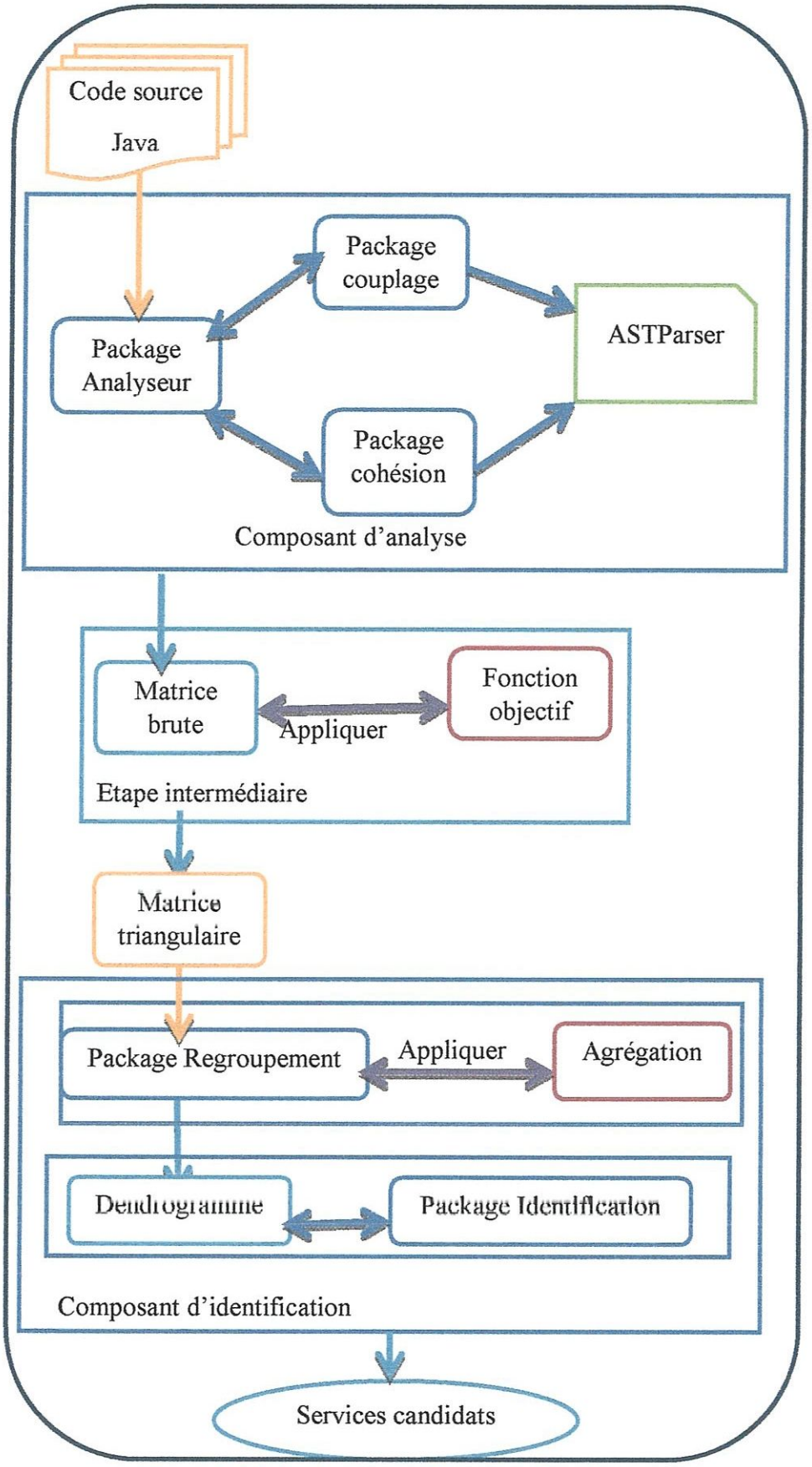


Figure 5. 2: Architecture globale de l'application

5.4. Description de l'application

Cette partie est consacrée à la présentation, en termes d'image de capture écran, les principales interfaces de l'application.

✓ **Interface principale**

Cette interface principale contient permet d'ouvrir un projet java afin de l'analyser.

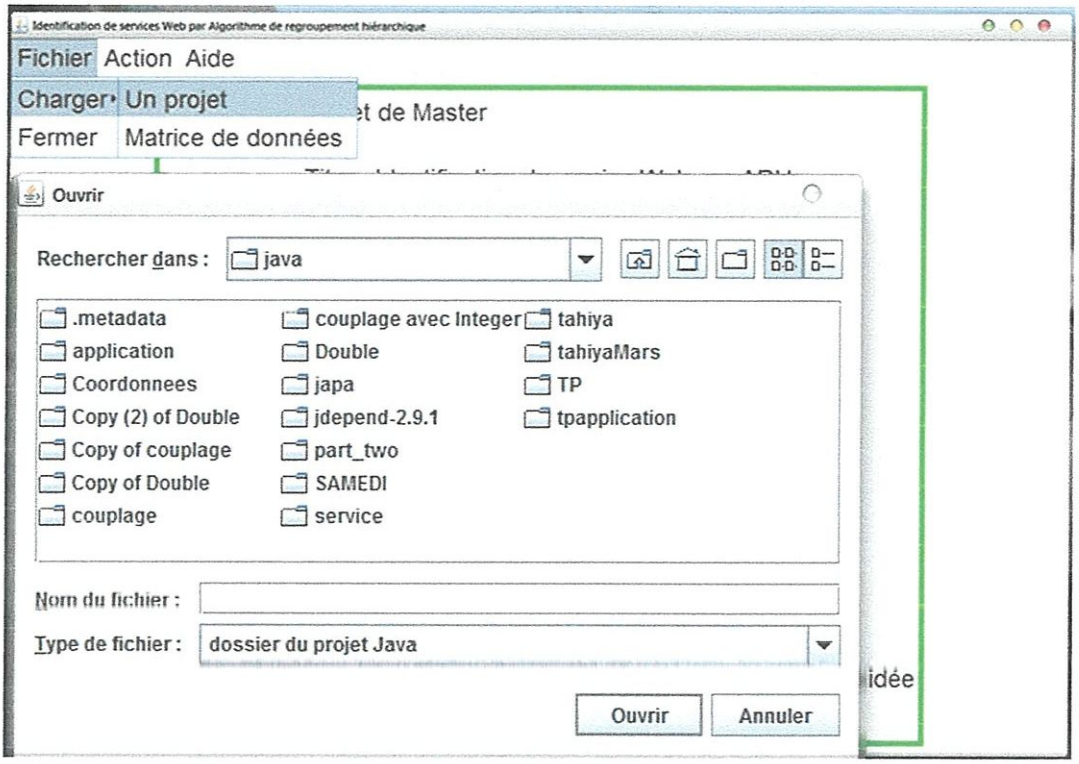


Figure 5. 3: Interface principale de l'application

Apres avoir charger le projet, l'étape suivante est l'étape d'analyse.

✓ **Etape d'analyse**

Cette étape permet d'analyser le projet afin d'évaluer les métriques de couplage et la cohésion sous forme matricielle. Voire figure 5.4.

La sortie d'analyse est une matrice qui s'appelle matrice bruite. Cette matrice permet d'interpréter les données à partir des valeurs de couplage et de cohésion sous forme matricielle. La diagonale est les valeurs de cohésion et les restes sont les valeurs de couplage.

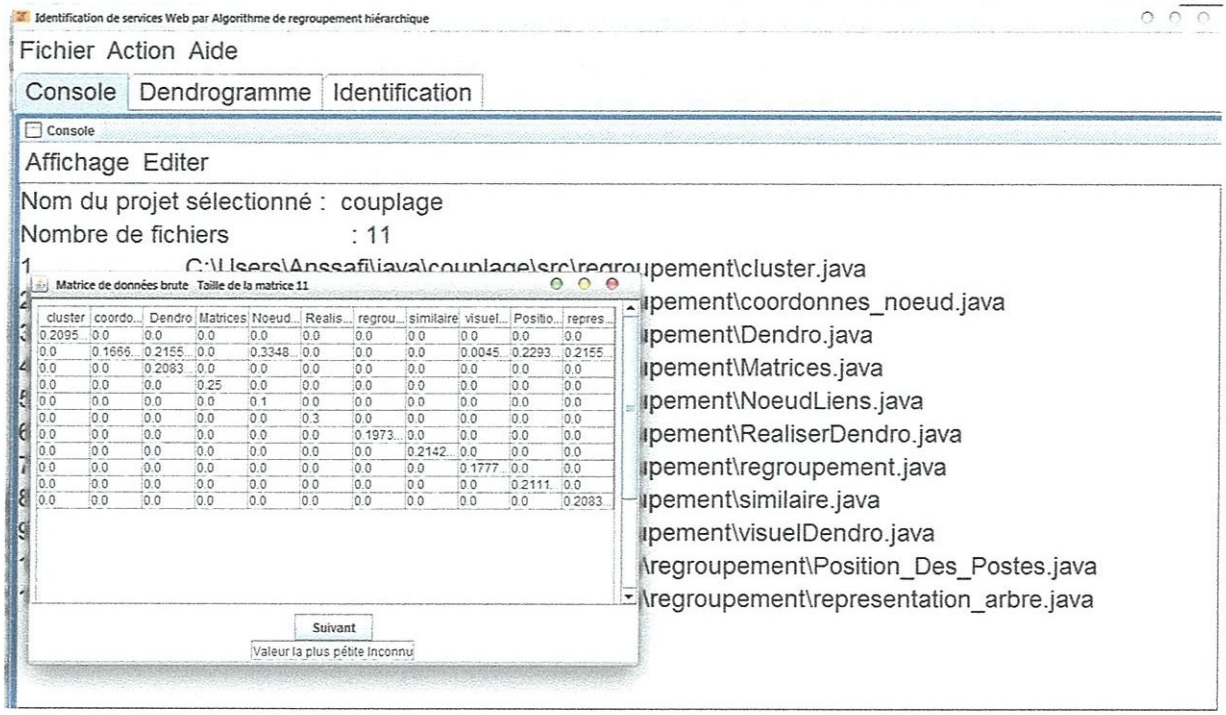


Figure 5. 4: Résultat d'analyse

En cliquant sur le bouton suivant, la matrice triangulaire s'affiche afin de commencer l'étape de regroupement.

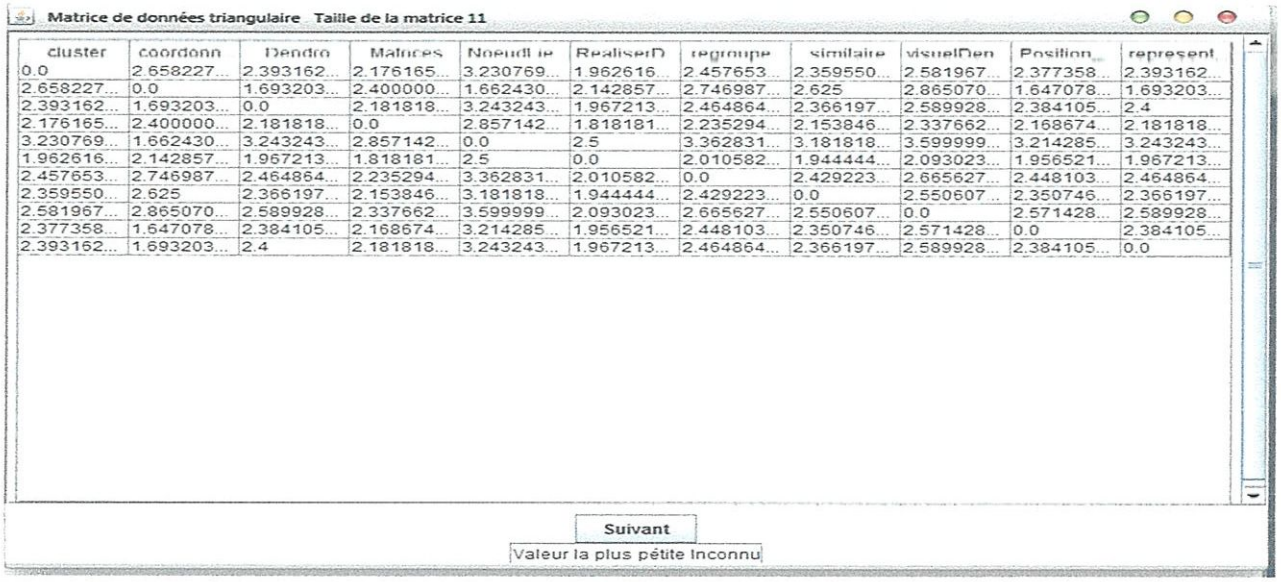
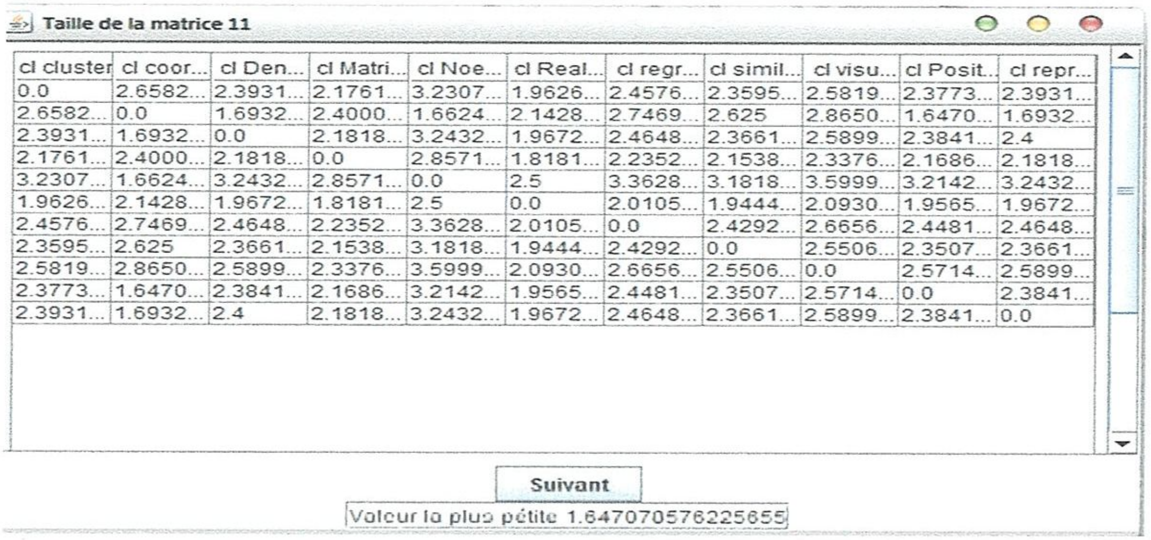


Figure 5. 5: Matrice triangulaire

En cliquant sur le bouton suivant ça permet de commencer le regroupement de classe.

✓ L'étape de regroupement

Cette étape commence avec une matrice triangulaire, voir figure 5.7.



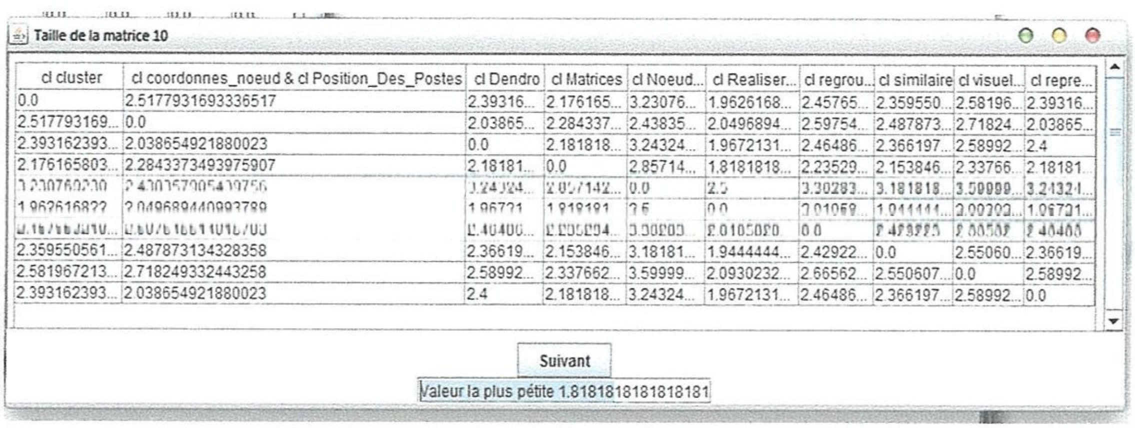
Taille de la matrice 11

cl cluster	cl coor...	cl Den...	cl Matri...	cl Noe...	cl Real...	cl regr...	cl simil...	cl visu...	cl Posit...	cl repr...
0.0	2.6582...	2.3931...	2.1761...	3.2307...	1.9626...	2.4576...	2.3595...	2.5819...	2.3773...	2.3931...
2.6582...	0.0	1.6932...	2.4000...	1.6624...	2.1428...	2.7469...	2.625	2.8650...	1.6470...	1.6932...
2.3931...	1.6932...	0.0	2.1818...	3.2432...	1.9672...	2.4648...	2.3661...	2.5899...	2.3841...	2.4
2.1761...	2.4000...	2.1818...	0.0	2.8571...	1.8181...	2.2352...	2.1538...	2.3376...	2.1686...	2.1818...
3.2307...	1.6624...	3.2432...	2.8571...	0.0	2.5	3.3628...	3.1818...	3.5999...	3.2142...	3.2432...
1.9626...	2.1428...	1.9672...	1.8181...	2.5	0.0	2.0105...	1.9444...	2.0930...	1.9565...	1.9672...
2.4576...	2.7469...	2.4648...	2.2352...	3.3628...	2.0105...	0.0	2.4292...	2.6656...	2.4481...	2.4648...
2.3595...	2.625	2.3661...	2.1538...	3.1818...	1.9444...	2.4292...	0.0	2.5506...	2.3507...	2.3661...
2.5819...	2.8650...	2.5899...	2.3376...	3.5999...	2.0930...	2.6656...	2.5506...	0.0	2.5714...	2.5899...
2.3773...	1.6470...	2.3841...	2.1686...	3.2142...	1.9565...	2.4481...	2.3507...	2.5714...	0.0	2.3841...
2.3931...	1.6932...	2.4	2.1818...	3.2432...	1.9672...	2.4648...	2.3661...	2.5899...	2.3841...	0.0

Suivant

Valeur la plus pétite 1.647070576225655

Figure 5. 6 : Matrice réalisée avant la première étape



Taille de la matrice 10

cl cluster	cl coordonnes_noeud & cl Position_Des_Postes	cl Dendro	cl Matrices	cl Noeud...	cl Realiser...	cl regrou...	cl similaire	cl visuel...	cl repre...
0.0	2.517793169336517	2.39316...	2.176165...	3.23076...	1.9626168...	2.45765...	2.359550...	2.58196...	2.39316...
2.517793169...	0.0	2.03865...	2.284337...	2.43835...	2.0496894...	2.59754...	2.487873...	2.71824...	2.03865...
2.393162393...	2.038654921880023	0.0	2.181818...	3.24324...	1.9672131...	2.46486...	2.366197...	2.58992...	2.4
2.176165803...	2.2843373493975907	2.18181...	0.0	2.85714...	1.8181818...	2.23529...	2.153846...	2.33766...	2.18181...
3.230769230	2.430357905439756	3.24324...	2.857142...	0.0	2.5	3.30283...	3.181818...	3.59999...	3.24324...
1.962616822	2.049689440993789	1.96721...	1.818181...	2.5	0.0	2.01058...	1.944444...	2.09302...	1.96721...
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0	[illegible]	[illegible]	[illegible]
2.359550561...	2.487873134328358	2.36619...	2.153846...	3.18181...	1.9444444...	2.42922...	0.0	2.55060...	2.36619...
2.581967213...	2.718249332443258	2.58992...	2.337662...	3.59999...	2.0930232...	2.66562...	2.550607...	0.0	2.58992...
2.393162393...	2.038654921880023	2.4	2.181818...	3.24324...	1.9672131...	2.46486...	2.366197...	2.58992...	0.0

Suivant

Valeur la plus pétite 1.8181818181818181

Figure 5. 7 : Matrice réalisée après la première étape

La première étape à regrouper la clMatrice et clRealiserdendro est la plus petite valeur est 2.06.Le bouton suivant permet de continuer les étapes de regroupements et la sortie sera la représentation de dendrogramme, voir figure 5.8.

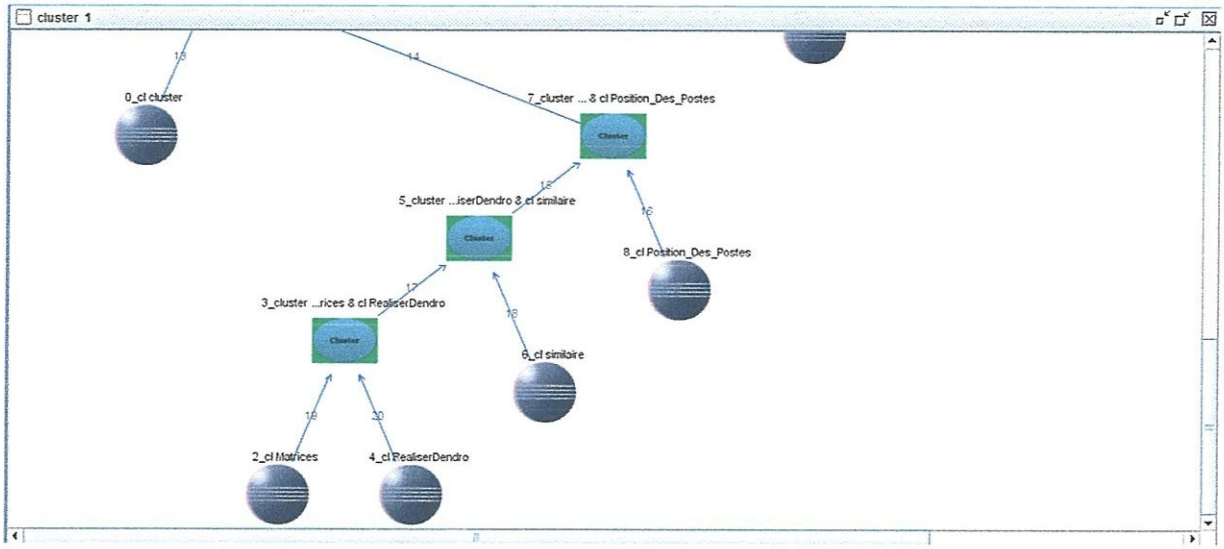


Figure 5. 9: dendrogramme

L'étape qui suit est l'étape d'identification.

- ✓ **L'étape d'identification**

L'identification des classes qui peuvent être considérées comme des services candidats, ça se fait par deux méthodes à savoir la méthode de recherche par nom d'une classe et la méthode de recherche par la distance. Comme option de recherche de services dans le dendrogramme nous proposons une méthode de parcours en profondeur. Nous soulignons que ces deux méthodes de recherches sont semi-automatiques.

- ✓ La méthode de recherche parle nom d'une classe nous permet d'identifier un service à partir de deux classes données. A la sortie, elle va identifier le service dans laquelle les deux classes appartiennent.
- ✓ La méthode de recherche par la distance, on fixe une valeur et puis elle regroupe les classes dans laquelle cette distance appartient.

On la lance en passant par le menu Afficher dans l'option identifier, voir fenêtre 5.10.

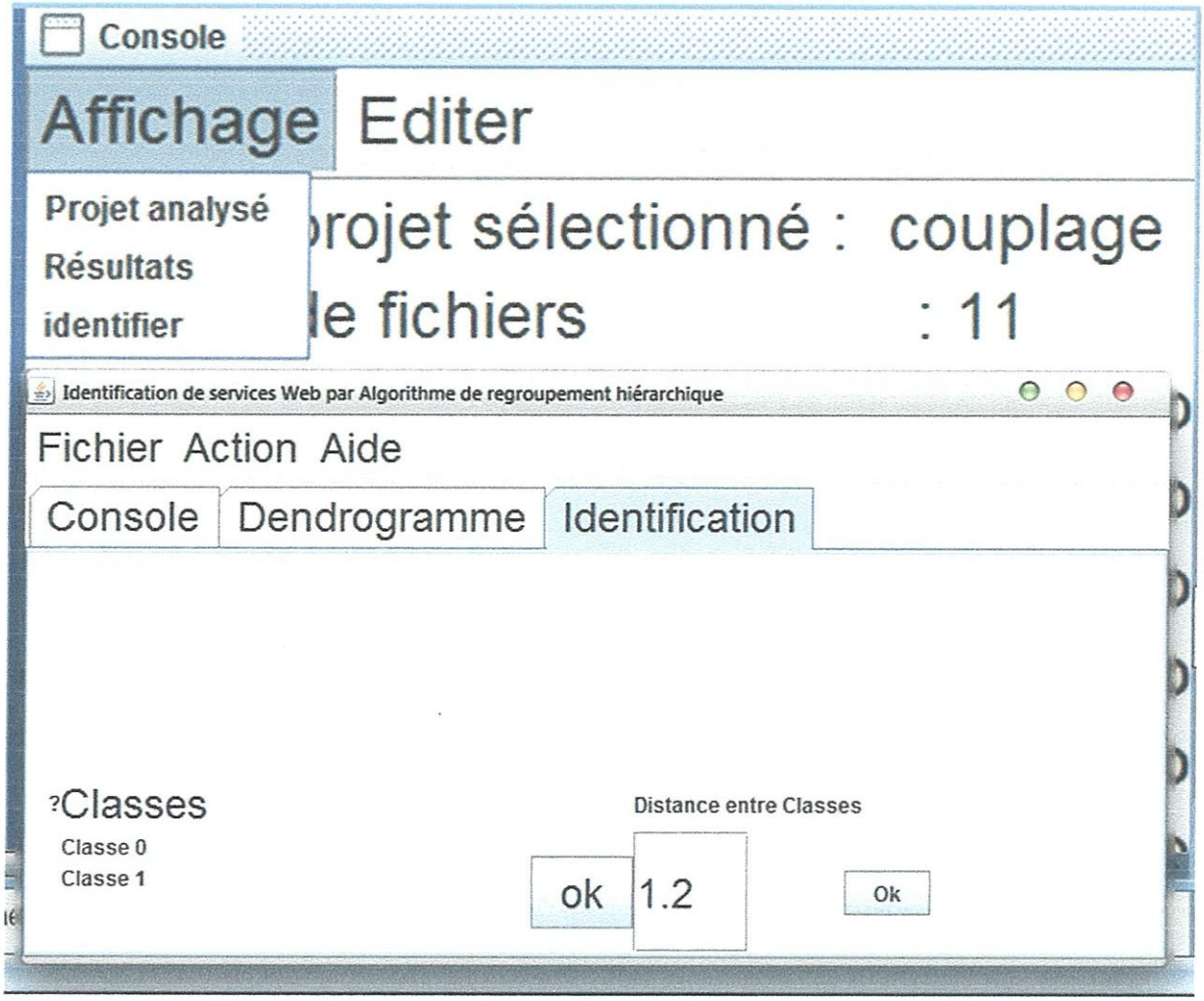


Figure 5.10:Fenêtre pour l' identification de services

5.5. Conclusion

Dans ce chapitre nous avons présenté les outils utilisés pour réaliser notre outil d'identification de services web. Nous avons aussi présenté le fonctionnement général de notre application.

Des imprimes écrans ont aussi été insérés afin de montrer les différentes options de notre outil.

REFERENCES BIBLIOGRAPHIQUES

CONCLUSION GENERALE ET PERSPECTIVES

Conclusion générale et perspectives

Ce projet que nous avons eu l'occasion de conduire à bout, rentre dans le cadre de la réingénierie des systèmes orientés objet vers de nouvelles architectures et plus particulièrement l'architecture orientée service. Il vise à automatiser l'identification des services web à partir du code source des applications JAVA.

La réalisation d'un outil pour l'approche d'identification de services web par l'algorithme de regroupement hiérarchique nous a permis de revoir en détail les notions rencontrées dans notre cursus académique , il nous a permis de revoir :

- ✓ les notions de métriques logicielles étudiées dans le module de génie logiciel avancé,
- ✓ le regroupement hiérarchique introduit dans le module d'Analyse de données,
- ✓ l'approche orientée objet et le langage de programmation Java utilisé dans les travaux pratiques.

En plus, nous avons observé les contraintes qui nous ont permis d'acquérir pas mal de choses.

L'utilisation de l'algorithme de regroupement hiérarchique nécessite plus de tests comme toute méthode d'exploration. Le résultat de son déroulement est en fonction des choix liés à la formule de la distance de similarité et de l'indice d'agrégation.

Les distances disponibles dans la littérature n'étant pas adéquates à notre cas d'étude, nous avons dû proposer notre propre fonction de similarité.

Il n'est pas facile de valider ses résultats automatiquement ; une analyse par un expert du système patrimonial est extrêmement souhaitable.

Avec l'espoir que notre contribution dans l'objectif final qu'est la migration d'une application orientée objet vers l'approche orientée services sera de l'ordre satisfaisant.

Nous pouvons suggérer comme perspective :

Un travail minutieux sur l'extraction des données liées au système patrimonial, dans un cadre extra académique, pour donner le temps aux tests d'exploration et valider les choix:

- ✓ des variables qui représentent les services à extraire,
- ✓ de la distance de similarité, la fonction objectif et,
- ✓ de l'indice d'agrégation.

REFERENCES BIBLIOGRAPHIQUES

✓ Webographie

[1] deptinfo.unice.fr/~baude/WS/CoursICAR-WS.pdf le 04 juin 2014

[3] http://msdn.microsoft.com/fr-fr/library/bb469924.aspx#wsdlexplained_topic03 le 04 juin 2014

[4] http://blog.xebia.fr/2009/04/29/soa-du-composant-au-service-lautonomie/ le 04 juin 2014

[5]http://www.microsoft.com/france/entreprises/plateformeapplicative/scenarios/details/business-processandsoa.aspx le 04 juin 2014

[7] www.zdnet.fr/.../soa-comprendre-l-approche-orientee-service-39206712 le 04 juin 2014

[9]http://www.commentcamarche.net/contents/web-services/soa-architecture-orientee-services.php3 le 04 juin 2014

[10] http://www.w3.org/ World Wide Web Consortium le 04 juin 2014

[11] http://genexo-a-m.googlecode.com/files/SOA%20final.doc le 04 juin 2014

[12]http://www.cetic.be/Architectures-Orientees-Services?lang=fr le 04 juin 2014

[13]http://www.entreprise-business.com/architecture-orientee-service-soa le 04 juin

[14]http://glossaire.infowebmaster.fr/xml/ le 04 juin 2014

[17]www.lirmm.fr/~seriai/encadrement/m2recherche/2012/sujet3.html le 04 juin 2014

[15] http://msdn.microsoft.com/fr-fr/library/bb469912.aspx#_Toc478383487 le 04 juin 2014

[21] fr.wikipedia.org/wiki/Réingénierie_logicielle le 04 juin 2014

[29]http://blog.xebia.fr/2007/08/16/mise-en-oeuvre-dune-soa-les-cles-du-succes/ le 04 juin 2014

[33]http://www.statsoft.fr/concepts-statistiques/classifications/exemple1-classification-ascendante-hierarchique.html le 04 juin 2014

[34] URL:http://nico.bonnel.googlepages.com/A2C1-0607CoursClustering.pdf le 04 juin 2014

[42] www.programcreek.com le 04 juin 2014

[43] http://genexo-a-m.googlecode.com/files/SOA%20final.doc le 04 juin 2014

✓ Néographie

[44] www.statsoft.fr /pdf/STATISTICA-prise-en.pdf le 04 juin 2014

✓ Bibliographie

[2] Cyrielle Lablanche, Florens Seine, Sébastien Gastaud : Les Web services. Université de Nice-Sophia Antipolis. Licence d'informatique, 2005

[6] livre blanc : Moteur d'innovation: les architectures orientées services 2007

[8] Livre blanc BEA : SOA et virtualisation : quelle complémentarité 2008

[16]Lionel Tricon, Initiation au couple gagnant WSDL/SOAP, Article publié dans GNU/Linux Magazine France, numéro 76 (Octobre 2005).

[18] Jean-François couturier élaboration et expérimentation d'une méthodologie agile permettant la migration vers une architecture orientée services en plus à l'aide d'openup. Mémoire de fin d'étude en Maitrise Université de Québec, octobre 2011

[19] Gilbert Raymond SOA : Architecture Logique : Principes, structures et bonnes pratiques, livre blanc

[20] Chikofsky EJ, Cross II JH. Reverse Engineering and Design Recovery: Taxonomy.

IEEE Software; 7(1):13–17, Janvier 1990

[22]A.Benzeltout et W.Yahyaui: Migration d'une application orientée objet vers une architecture orietée service. Mémoire de fin d'études Master Université 8 Mai 1945 Guelma, Juin 2012.

[23] M. Brodie and M. Stonebraker: Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach, Morgan Kaufmann San, Francisco1995.

[24] Bisbal, J.; Lawless, D.; Bing Wu; Grimson, J.; "Legacy information systems: issues and directions," Software, IEEE, vol.16 , no.5, pp.103-111, Septembre /Octobre 1999.

[25] Amjad Umar and Adalberto Zordan. Reengineering for service oriented architectures: A strategic decision model for integration versus migration. J. Syst. Softw. 82, 3 (Mars 2009

[26]Marchetto, A., Ricca,F.: From objects to services: toward a stepwise migration approach for java applications. STTT 11(6) (2009) 427-440

[27]A. Almonaies, J.R. Cordy and T.R. Dean, "Legacy System Evolution Towards Service-Oriented Architecture", Proc. SOAME 2010, International Workshop on SOA Migration and Evolution, Madrid, Spain, March 2010, pp. 53-62.

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

[30]. P. Berkshire: Survey of Clustering Data Mining Techniques, 2002

[31]. A. K. Jain, M. N. Murty, P.J. Flynn: Data Clustering: A Review, ACM Computing Serveys, Journal Vol. 31, No.3, September 1999

[32] Nicolas Bonnel, Acquisition de connaissances à partir de données : méthodes numériques, (2007).

[35] 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, Conférence Morocco Octobre 13-15 2011

[36]W. STEVENS, G. MYERS and L. CONSTANTINE, «Structured Design» IBM Systems Journal, Vol. 13, No 2, 1974.

[37] Lazhar Sadaoui : Evaluation de la cohésion des classes : une nouvelle approche basée sur la classification. Mémoire de fin d'année Université du Québec à trois rivière Juin 2010.

[38]Sylvain CHARDIGNY Extraction d'une architecture logicielle à base de composants depuis un système orienté objet. Une approche par exploration Nantes. Thèse Doctorat octobre 2009.

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

[40] Sukainah Husein et Alen Oxley: A Coupling an Cohesion Metrics Suite For Object-Oriented Software. Article PETRONAS UTP Bandar Seri Iskandar, Malaysia, 2009

[41] Seridi Ali Réingénierie des systèmes existants vers la SOA: application au domaine ubiquitaire. Rapport de stage Université Mines de Douai juin 2008

[45] PALLAVIDINO Luc Vue Unifiée d'un ensemble documentaire Paul Sabatier Rapport de stage Université TOULOUSE, Juin 2008.