الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de L'enseignement Supérieure de la recherche scientifique

Université 8 Mai 45 –Guelma-

Faculté des Mathématiques, d'informatique et des Sciences de la Matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Un algorithme génétique pour l'identification des services

Encadré par : Mr Seridi Ali

Présenté par :

Hamici Khayreddine

Septembre 2021

Table des matières

Chapitre 02 : L'Algorithme Génétique

Chapitre 03 : La Conception

Chapitre 04 : L'implémentation

Liste des Figures

Liste de tableaux

Remerciement

Je tiens tout d'abord à remercier notre Dieu qui m'a donné la force, la santé, le courage, la persévérance et la patience afin de réaliser ce modeste travail.

Je remercie chaleureusement mes parents qui m'ont soutenu tout au long de mes études, parfois au prix de quelques sacrifices et sans qui je n'aurais pas pu atteindre ce niveau. Un grand merci à mes sœurs qui m'ont toujours encouragé pour poursuivre mes études.

Je tiens à remercier mon encadreur Monsieur Seridi Ali, pour sa supervision, sa présence, ses conseils précieux et pour les efforts qu'il avait consentis durant la rédaction tout le long de mon mémoire de Master.

Je désire aussi remercier les enseignants de l'université 08 MAI 1945, de Guelma (département d'informatique), qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

Mes remerciements vont aussi à tous les membres du jury, qui m'ont fait l'honneur de juger ce travail.

Finalement, Merci à tous mes amis, mes amis d'étude, merci à tous ceux qui m'ont aidé à réussir dans mes études.

DEDICACE

En premier lieu, à la lumière de mes yeux, à mes plus chers êtres au monde avec toute mon affection, toute ma gratitude, que Dieu les gardes, mes parents.

Toi Maman, la lumière de mes jours, ma vie et mon bonheur. Je te remercie aussi pour la force et la compassion que tu m'avais fait sentir.

Toi Papa, mon exemple éternel, mon soutien moral celui qui s'est toujours sacrifié pour me voir réussir. Grâce à ton amour et ta sollicitude j'arrive aujourd'hui au couronnement de mes efforts.

A mes chères sœurs Amira, Amel, Nessrine et Nihad.

A mon intime Aymen

A mon bijou de vie Imane

Au mari de ma sœur Walid.

A mes charmantes amies.

A toute la famille grande et petite.

A tous les étudiants de notre promo.

A tous ceux qui m'ont aidé de loin ou de près durant mes études.

Résumé

La réingénierie logicielle est un processus visant à convertir et à améliorer les logiciels à partir de modèles existants dans une nouvelle architecture afin d'assurer une meilleure maintenabilité et de gagner en performance, parmi les nouvelles architectures qui sont de plus en plus utilisées c'est l'architecture orientée service (SOA).

La migration vers la SOA en utilisant le processus de réingénierie s'avère nettement moins chère qu'un redéveloppement d'une nouvelle application.

A travers notre PFE, nous avons essayé de contribuer à l'automatisation du processus de réingénierie des systèmes existant vers une architecture SOA et plus particulièrement la phase d'identification des services, qui est une phase incontournable dans toute méthode de migration vers la SOA. Nous avons modélisé notre problème comme un problème de clustering, ce qui nous a mené à choisir l'algorithme génétique pour atteindre notre objectif.

Les mots clé

Architecture Orienté Service, Service Web, Heuristique, Méta Heuristique, La Réingénierie, La Migration Vers SOA**,** Identification Des Services, Application Orienté Objet, Couplage, Cohésion.

Abstract

Software reengineering is a process of converting and improving software from existing models into a new architecture to ensure better maintainability and gain performance. One of the new architectures that is increasingly used is service-oriented architecture (SOA).

Migration to SOA using the reengineering process is much less expensive than redeveloping a new application.

Through our project, we have tried to contribute to the automation of the process of reengineering existing systems towards an SOA and more particularly the service identification phase, which is an essential phase in any migration method towards SOA. We modeled our problem as a clustering problem, which led us to choose the genetic algorithm to achieve our goal.

The key words

Service Oriented Architecture, Web Service, Heuristics, Meta Heuristics, Reengineering, Migration to SOA, Service Identification, Object Oriented Application, Genetic Algorithm, Coupling, Cohesion.

الملخص

إعادة هندسة البرمجيات هي عملية تهدف إلى تحويل وتحسين البرامج من النماذج المتوفرة إلى بنية جديدة من أجل ضمان قابلية أفضل للصيانة وتحقيق مكاسب في الأداء، من بين البنى الجديدة التي يتم استخدامها بشكل متزايد هذه الأيام هي هندسة الخدمة الموجهة (SOA)

يعد الترحيل إلى SOA باستخدام عملية إعادة الهندسة أرخص بكثير من إعادة تطوير برنامج جديد.

من خلال المذكرة الخاص بنا، قمنا بالمساهمة في عملية إعادة هندسة الأنظمة الحالية نحو بنية SOA وبشكل أكثر تحديدًا مرحلة تحديد الخدمات، وهي مرحلة أساسية في أي طريقة للانتقال إلى SOA. لقد قمنا بنمذجة مشكلتنا كمشكلة تجميع، مما دفعنا إلى اختيار الخوارزمية الجينية لتحقيق هدفنا.

الكلمات الرئيسية

الهندسة الموجهة للخدمة، خدمة الويب، إعادة الهندسة، الانتقال إلى الخدمية، تحديد الخدمات، التطبيق الموجه للكائنات، الخوارزمية الجينية، التزاوج، التماسك.

Introduction générale

Dans l'industrie du logiciel, la qualité et les couts des logiciels sont considérés comme les paramètres les plus importants et les plus critiques à prendre en considération lors du processus de développement logiciel. Les logiciels sont conçus pour répondre aux attentes des parties prenantes le plus longtemps possible afin de rentabiliser les investissements déployés. Malheureusement après un certain temps d'exploitation du logiciel, on se trouve dans une situation où nous sommes obligés soit d'abandonner et de remplacer le logiciel à cause de sa technologie devenant obsolète et des coûts de maintenance devenant élevés, soit de le redévelopper en suivant une des techniques de réingénierie.

La réingénierie est l'une des méthodes qui sert à améliorer la qualité du logiciel. Elle consiste à l'amélioration et la transformation d'une application classique vers une autre application plus moderne.

Nous pouvons constater que les applications orientées objet occupent une large partie des applications disponibles sur le marché, malheureusement elles sont moins adaptées aux technologies web. Les faire migrer vers des architecture plus modernes et plus adaptées aux technologies web serait plus bénéfique. Parmi les architectures modernes on trouve l'architecture orientée service qui s'est imposée sur le terrain en raison de ses caractéristiques et ses avantages.

Il y a de nombreuses approches pour faire migrer une application orientée objet vers une architecture orientée service, parmi elles nous trouvons les approches ascendantes (bottom-up) qui consistent à démarrer du code source de l'application existante, analyser ses modules pour enfin identifier les services qui vont constituer la nouvelle application SOA.

Dans le cadre de notre travail nous avons proposé d'automatiser cette phase d'identification de service en ramenant le problème à un problème de clustering. Nous avons aussi opté pour l'utilisation de l'algorithme génétique (AG) pour regrouper les classes qui sont fortement couplés et cohérentes dans un seul groupe qui constitue le service.

Afin d'appliquer l'AG nous avons dû adapter tous les paramètres et proposer une fonction de fitness qui soit adéquate à notre problématique.

Notre mémoire est organisé en quatre chapitres, qui sont comme suit :

Chapitre 01 « la réingénierie logicielle vers la SOA »

Dans ce chapitre nous avons défini la réingénierie logiciel, l'architecture orientée service et ses concepts de base, ainsi que les concepts de la migration vers la SOA.

Chapitre 02 « Algorithme Génétique »

Ce chapitre contient une présentation de l'algorithme génétique en générale.

Chapitre 03 « Conception »

Dans ce chapitre nous avons expliqué le processus que nous avons suivi pour atteindre notre objectif et comment nous avons adapté l'algorithme génétique afin d'identifier les services.

Chapitre 04 « l'implémentation »

Dans ce chapitre nous avons présenté tous les outils de l'implémentation, ainsi que la présentation des différentes interfaces de notre application et enfin une discussion des résultats obtenus.

A la fin, nous avons clôturé ce mémoire par une conclusion générale et quelques perspectives.

Chapitre 01 : la réingénierie logicielle vers la SOA

1. Introduction

Un logiciel est le fruit de plusieurs années de développement, et d'énormes investissements financiers. Malgré cet investissement, certains logiciels sont abandonnés du fait des coûts parfois trop importants pour les maintenir. Ceci peut être dus à une perte du savoir-faire, à l'impossibilité de contacter les auteurs du code où à la technologie devenant obsolète. Dans certains cas nous préférons modifier ou maintenir le logiciel que de développer un nouveau et perdre le temps, l'argent. C'est ce cas où intervient la réingénierie logicielle.

2. La réingénierie logicielle

La réingénierie est un processus qui vise à réduire les coûts tout en augmentant la productivité et en fournissant des niveaux de service plus élevés. Dans le monde des affaires et des entreprises, elle est définie comme :

2.1 Définition de la réingénierie logicielle

La réingénierie logicielle (RL) est un processus qui vise à transformer et à améliorer les logiciels à partir de modèle existant. Le logiciel en question ne doit pas subir de régression fonctionnelle, mais doit améliorer les performances et l'évolutivité.

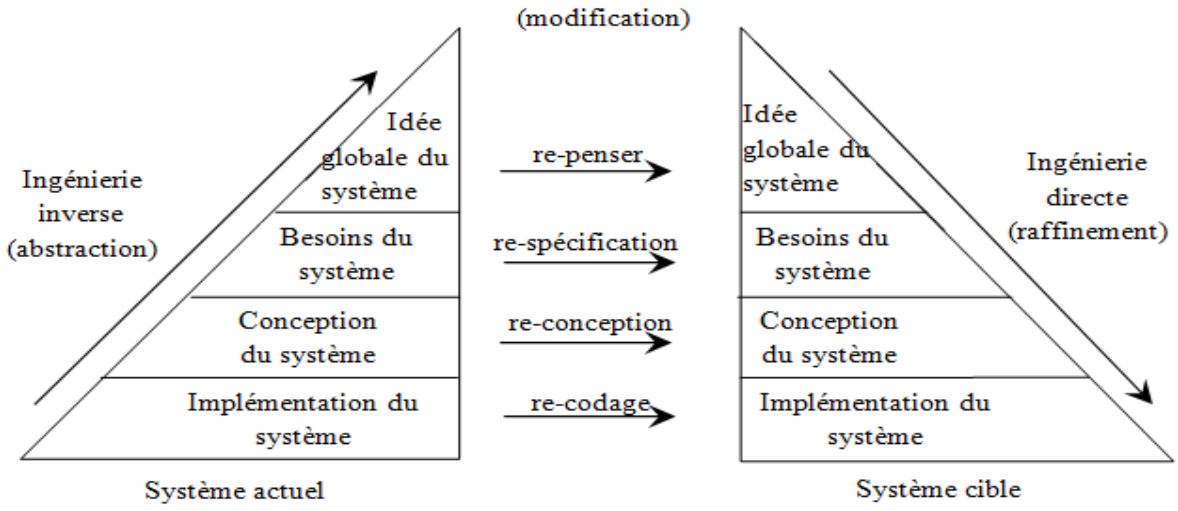


Figure 1.1 : Modèle général de la réingénierie du logiciel [20]

2.2 Domaines d'application

La réingénierie logicielle fait partie du domaine de la maintenance logicielle. C'est un processus de transformation utilisé pour l'analyse des besoins (définition des spécifications des problèmes, des objectifs, des contraintes...), le design (spécification des solutions) et l'implémentation logicielle (codage, réécriture, tests ...). Mais un logiciel n'est

pas seulement du code sur lequel la réflexion et l'intervention va porter. C'est aussi de la documentation, des représentations graphiques se rapportant au cahier des charges (aux spécifications), des données de tests, de validation et tout un ensemble de documentations et d'informations qui composent le logiciel lui-même et lui donne du corps. [17][18][19]

2.3 Processus utilisé

Il y a 3 processus essentielle pour la réingénierie de logicielle [1]

- La rétro-ingénierie : représente la première étape de la réingénierie qui consiste à effectuer le chemin inverse par rapport au processus de développement. On pourrait la définir comme le contraire de l'ingénierie autrefois nommée "forward engineering". Le reverse engineering est une partie indispensable de la maintenance logicielle, puisque celle-ci ne peut être réalisée sans une compréhension complète du système. Parfois, la compréhension d'un logiciel est si complexe en l'absence de toute documentation, que la tâche est presque impossible. C'est la raison pour laquelle la rétro-ingénierie est un processus qui a pour rôle de comprendre les différentes informations inconnues ou cachées du système logiciel.

- La redocumentation : constitue la forme la plus ancienne du processus de rétro-ingénierie puisque c'est la révision ou la création des représentations équivalentes à un code source donné.

- La restructuration : est la transformation d'une forme de représentation en une autre au même niveau d'abstraction en préservant l'interface externe du système. On peut utiliser la restructuration afin d'adapter un système à de nouvelles contraintes, ou comme maintenance préventive pour améliorer l'état physique du système en respectant par exemple des standards existants.

- Il y a aussi d'autre processus utilisé comme La mesure du logiciel (software measurement), Navigation... qui peuvent être utilisés dans une démarche de réingénierie logicielle.

2.4 Les risques reliés à la réingénierie logicielle

Malgré les efforts de la réingénierie pour atténuer les risques, réduire les coûts de fonctionnement et maintenir le système existant, il reste un domaine à risque. Être capable d'identifier les risques très tôt dans le processus pourrait aider à être plus efficace. Le tableau ci-dessous synthétisé par « Boby LAFORET » dans [2] définit les différents risques susceptibles d'être gérés au cours de la réingénierie.

Zone de risque	Risques
Processus	– Les coûts extrêmement élevés des manuels de réingénierie. – Avantages de coûts qui ne se réalisent pas dans le cadre de temps nécessaire – Dérives de l'effort de réingénierie – Le manque d'engagement de la direction à la solution de réingénierie continue – Sous-systèmes pour l'approche choisie de manière incorrecte – Gestion de configuration inadéquate – Le manque d'assurance de la qualité – Le manque de programme de mesures – Réingénierie sans aucun expert en application locale disponible – Système de réingénierie ne fonctionnant pas correctement – Masse de documentation dont le coût de la production est trop élevé – La technologie de réingénierie logicielle est insuffisante pour atteindre les objectifs du domaine – La fonctionnalité patrimoniale devient obsolète avant que la réingénierie s'achève
Rétro-ingénierie	– Le langage utilisé n'est pas conçu pour exprimer les informations abstraites nécessaires pour les exigences et la conception. – Difficulté de capture plus de conception et d'exigences à partir du code source – L`Objet capturé est incomplet ou incorrect – Les connaissances des règles d'affaires existantes sont perdues dans le code source – Recouvert des informations non utiles et non utilisées

Ingénierie	– Nouvelles exigences et fonctionnalités sont ajoutées. – Capturé des objets qui ne seront pas intégrés dans le nouveau système – Difficulté dans la migration des données existante – Le niveau de préparation et de rétro-ingénierie est insuffisant
Personnel	– Les personnels n'ont pas le niveau de compétences requises et n'ont pas assez d'expériences dans la réingénierie.
Outils	– Dépendance à des outils qui ne fonctionnent pas comme prévu – Des outils disponibles, mais nouveaux et immatures dont certaines fonctionnalités sont incomplètes. – Maturité – Stabilité du fournisseur de l'outil et de la qualité de l'outil
Stratégie	– Engagement prématuré d'une réingénierie pour un système au complet. – À défaut d'avoir une vision à long terme avec des objectifs intermédiaires. – Objectifs irréalistes. – L'approche choisie ne répond pas aux objectifs de l'entreprise, le budget ou Calendrier. – Pas de plan d'utilisation des outils de réingénierie.

Tableau 1.1 : Les risques potentiels de la réingénierie logicielle (Rosenberg et al., 1996)

2.5 Les avantages de la réingénierie logicielle

Les principaux avantages de la réingénierie logicielle sont les suivants (Abbas et al., 2012). [2]

- Augmenter la maintenabilité du logiciel : cet avantage permet d'avoir un système facile à modifier et corriger les erreurs, améliorer la performance et certains autres attributs. Faciliter l'adaptation du système dans un environnement changeant.
- Améliorer la performance du logiciel : Cet avantage permet d'économiser d'argent, d'augmenter la productivité, de diminuer le coût de matériel et de développement, garder une bonne relation avec la clientèle.

- Augmenter l'interopérabilité du logiciel : facilité la communication entre les différents systèmes existants.
- Diminuer la dépendance du système au personnel : réduire la coordination entre plusieurs équipements de développement géographiquement éloignés ou distancés aiderait à être plus efficace.
- Améliorer la testabilité : réduire la complexité du système et de ses composants permet de gagner en termes de coûts de test et de temps de test.

3. L'architecture orientée service

3.1 Définition

La SOA est un ensemble de principes architecturaux qui permettent de développer des systèmes modulaires basés sur des « services » ou des unités de fonctionnalités informatiques. Ces services, qu'ils soient métier ou techniques, sont offerts par une entité, le prestataire de services, et consommés par une autre [3].

3.2 Les caractéristiques de l'architecture orientée service

SOA possède les caractéristiques suivantes :

- **Réutilisation et composition :** Ceci est particulièrement puissant pour créer des nouveaux processus d'affaires rapide et fiable.
- **Recomposition :** La capacité de modifier les processus d'affaires existants ou d'autres applications basées sur l'agrégation des services.
- **La capacité à progressivement changer le système :** Commutation des prestataires de services, l'extension des services, en modifiant les fournisseurs de services et les consommateurs. Tous ces éléments peuvent être faits en toute sécurité, grâce au couplement bien contrôlé.
- **La capacité à construire progressivement le système :** Cela est particulièrement vrai pour toute intégration basée sur SOA [21].

3.3 Les acteurs de l'architecture orientée service

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

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

Le répertoire de services a un rôle primordial dans la SOA. C'est lui qui reçoit la requête du consommateur, lui qui découvrira le service idoine, et lui qui agira en tant que proxy (intermédiaire) entre consommateur et service. En s'assurant que les fournisseurs de services informent régulièrement les répertoires de leurs nouveautés, le consommateur peut constamment profiter de celles-ci sans pour autant devoir mettre à jour ses méthodes [W2].

3.4 Les bénéfices de l'architecture orientée service

SOA avec sa nature faiblement couplée permet aux entreprises de :

- Brancher de nouveaux services.
- Améliorer les services existants de façon granulaire pour répondre aux nouveaux besoins.
- Offrir la possibilité de rendre les services consommables à travers différents canaux.

3.5 Les avantages de l'architecture orientée service [W1]

- 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.

3.6 Les inconvénients de l'architecture orientée service [W3]

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

4. La réingénierie vers SOA

L'architecture orientée services (SOA) rend le développement d'applications plus flexible et adaptable que les applications créées à l'aide d'architectures traditionnelles. Par conséquent, les applications SOA sont plus faciles à modifier et à adapter. La SOA peut également permettre un meilleur alignement entre les processus métier et les services ou applications informatiques. Cet alignement a pour effet d'un coté de minimiser l'écart sémantique entre les modèles de processus métier et les logiciels des applications réels et d'autre coté d'optimiser l'efficacité et l'agilité de l'entreprise. La réingénierie des anciens systèmes vers une SOA permet d'une part de bénéficier des avantages de cette architecture et d'autre part d'allonger la vie de l'exploitation du logicielle qui a demandé beaucoup d'investissements.

4.1 Les différentes approches pour la réingénierie vers SOA

Il existe quatre approches pour migrer vers la SOA lesquelles :

4.1.1 Top Down

Dans ce type de projet d'intégration le système d'information de l'entreprise est vu comme un super bloc applicatif. Cette approche est descendante et globalisante dans la mesure où elle touche à la globalité de l'entreprise. Elle vise à créer un socle commun d'intégration aligné sur le métier de l'entreprise.

Le point de départ de cette approche est la définition et la formalisation des processus métiers, qui représentent les objectifs fonctionnels de l'entreprise, pour descendre ensuite au travers des différentes couches du système afin de définir les services nécessaires à la réalisation de ces processus. Cette approche peut être appliquée dans le

cadre d'un système urbanisé ou pour démarrer un nouveau projet. Cependant, elle est très coûteuse, très longue (plusieurs années) et trop risquée, puisqu'elle cause la reconstruction de tout ou d'une partie de système d'information [14] [15]

4.1.2 Bottom Up

A l'inverse de Top-Down, cette approche prône une phase de conception ascendante. Une analyse de l'existant (cartographie applicative) permet de déterminer les fonctions existantes du SI. Ainsi cartographiées, il est possible d'identifier les fonctions du SI qui sont éligibles au rang de service. Une fois ces fonctions mises en mode service, elles sont exploitables au sein de services à forte valeur ajoutée et / ou de processus métiers.

Cette approche peut séduire par plusieurs aspects :

- Elle contraint à réaliser une cartographie du SI qui, si elle est tenue à jour et publiée, faciliterait la réutilisation de services.
- Le coup du ticket d'entrée peut paraître moins élevé que pour une démarche Top Down.
- Elle présente cependant de sérieux défauts :
- Elle bloque le pilotage de la SOA par les besoins métiers.
- Elle ne favorise pas la mise en place d'un effort transverse : elle ne permet pas de sortir de la « culture projet ». En effet, les services émergeants de cette approche restent très fortement couplés à leur application d'origine.
- Elle rend très difficile la justification de l'investissement auprès du métier, qui n'en verra les bénéfices qu'une fois les services auront été rendus exploitables au sein de processus métiers. [4]

4.1.3 Middle Out

Par opposition à l'approche *Outside In*, cette méthode propose de commencer au milieu « en anglais : In the middle », c'est-à-dire là où le métier et les technologies d'information (ou, IT pour *Information Technology*) parlent le même langage (en tout cas presque). Elle s'attaque donc d'emblée à ce qui reste un des facteurs limiteurs à l'adoption des SOA : La compréhension du métier de l'entreprise par les maîtrises d'œuvre et inversement, la compréhension des contraintes IT par les maîtrises d'ouvrage. Une fois les différentes parties sont d'accord sur un premier socle de services "métiers" nécessaires :

- Les maîtrises d'ouvrage engagent un chantier "*Middle Up*" pour spécifier les processus métiers.
- Les maîtrises d'œuvre engagent un chantier "*Middle Down*", pour spécifier le socle de services de plus bas niveau permettant la réalisation des services métiers.

Cette approche limite par contre le pilotage de la SOA par les besoins métiers, puisque le point de départ est l'identification des services métiers nécessaires et non la définition des processus réalisant le métier de l'entreprise [4] [16].

4.1.4 Outside In (Meet In The Middle)

Cette approche préconise de mener en parallèle :

- Un chantier Top Down pour définir les processus métiers et les services de plus haut niveau nécessaires à leur réalisation.
- Un chantier Bottom Up afin de cartographier l'existant applicatif dont dispose l'entreprise pour supporter les services métiers à forte valeur ajoutée.

Une fois ces deux chantiers en phase finale, commence la délicate étape de l'accostage. Son objectif est de « réconcilier » les résultats des deux approches afin de déterminer comment seront réalisés les processus métiers. Il faut, pour cela, croiser les besoins en services (exprimés par le chantier Top Down) et le patrimoine applicatif (cartographié par le chantier Bottom Up).

De par sa nature, cette approche réunit les bénéfices des approches Top Down et Bottom Up. Elle permet de piloter la SOA par les besoins métiers tout en facilitant la réutilisation de services et la capitalisation sur l'existant. Mais elle cumule également les travers des deux approches, notamment en induisant un très fort risque d'effet tunnel. [4]

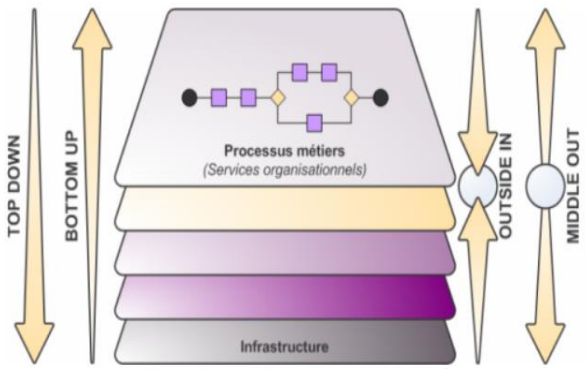


Figure 1.2 différentes approches pour la réingénierie vers SOA [W4]

5. Conclusion

La réingénierie est l'examen, l'analyse, et la restructuration d'un logiciel existant pour le reconstituer sous une nouvelle forme et implémenter ainsi cette nouvelle forme. Nous avons présenté dans ce chapitre la réingénierie logicielle en général, les domaines d'application avec les risques et les processus utilisé. Ensuite nous avons présenté l'architecture orientée service (SOA) avec ces caractéristiques, bénéfices, avantages et inconvénients. En fin nous nous sommes concentrés sur la réingénierie vers une SOA avec les différentes approches existantes.

Le chapitre suivant, sera consacré à la présentation d'un des algorithmes de regroupement, qui est l'algorithme génétique que nous avons utilisé pour la réingénierie des applications orientées objet vers une SOA.

Chapitre 02 : Algorithme Génétique

1. Introduction

L'optimisation combinatoire définit un cadre formel pour de nombreux problèmes dans l'industrie, la finance ou la vie quotidienne. Les problèmes d'optimisation combinatoire sont généralement définis comme le problème de la sélection de la meilleure solution à partir d'un ensemble très large mais limité de solutions.[40] Il existe de nombreuses méthodes pour résoudre ses problèmes parmi ces méthode les méthodes heuristiques et les méthodes méta-heuristiques.

1.1 Définition

Heuristique : Une heuristique est généralement une stratégie de solution empirique qui utilise la structure du problème. Généralement sous-optimal, mais converge rapidement vers une bonne solution.

Deux types d'heuristiques sont principalement utilisés :

- Construire des heuristiques (telles que les méthodes gloutonnes), qui construisent itérativement des solutions,
- L'heuristique descendante consiste à trouver l'optimum local à partir de la solution donnée.

Méta-heuristique : la méta-heuristique est un algorithme d'optimisation qui vise à résoudre des problèmes d'optimisation difficiles pour lesquels il n'existe pas de méthode classique plus efficace. Elle utilise deux approches principales pour résoudre un problème :

- La première est nommée « approche à population ». Elle désigne les algorithmes qui traitent plusieurs solutions à la fois. Elles maintiennent et améliorent plusieurs solutions candidates en même temps.: (Algorithme Génétique (AG), Algorithme de colonie de fourmis (ACO), Optimisation par Essaim Particulaire (PSO), etc...).|28]
- La seconde est nommée "approche de trajectoire". Elle représente les algorithmes qui font évoluer une fonction objectif unique à chaque itération. La stratégie est basée sur la recherche locale : (Recherche Tabou(RT), Recuit Simulé (RS), etc).|28]

2. L'algorithme génétique

2.1 Historique

En 1859, le naturaliste Charles Darwin a publié son célèbre ouvrage L'origine des espèces, qui présentait une théorie visant à expliquer le phénomène de l'évolution. Cette théorie révolutionnaire a provoqué plusieurs débats à l'époque mais elle est maintenant largement acceptée. Environ 100 ans plus tard, elle a inspiré le professeur John Holland qui a tenté d'implémenter artificiellement des systèmes évolutifs basés sur le processus de sélection naturelle. Ces travaux ont ensuite mené aux algorithmes génétiques, qui sont maintenant utilisés pour obtenir des solutions approchées pour certains problèmes d'optimisation difficiles. [26]

2.2 Définition

Les algorithmes génétiques (GA) sont des méta-heuristiques relativement anciennes. Ils ont été introduits à la fin des années 60 par John Holland [27]. Eux aussi ont souvent été utilisés dans le domaine du génie logiciel. Ils sont basés sur la théorie de l'évolution de Darwin : les individus s'affrontent pour survivre et les plus adaptés ont les plus grandes chances de survivre et de se reproduire. L'idée de base des GA est de démarrer à partir d'un ensemble de solutions initiales et d'utiliser des mécanismes d'évolution inspirés par la biologie pour créer de nouvelles et potentiellement meilleures solutions.

2.3 Principe de base

Les algorithmes génétiques sont des approches d'optimisation qui utilisent des techniques dérivées de la science génétique et de l'évolution naturelle : la sélection, la mutation et le croisement. [5] Pour utiliser ces approches, on doit disposer des éléments suivants : [6]

- Le codage d'un élément de population : une fonction qui permet de modéliser les données du problème réel dans des données utilisables par l'algorithme génétique.
- Une fonction qui génère la population initiale : la génération de la population initiale est importante puisque cette génération représente le point de départ de l'algorithme et son choix influe sur la rapidité et l'optimalité de la solution finale.

- Une fonction à optimiser (la fonction objective ou fitness) : une fonction qui retourne une valeur d'adaptation pour chaque individu. Cette valeur permet de déterminer la solution pertinente puisque le problème se restreint à chercher le groupe d'individus qui ont les valeurs optimums.
- Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.
- Des paramètres de dimensionnement : taille de la population, un critère d'arrêt, une probabilité pour appliquer les opérateurs de croisement et de mutation.

2.4 Caractéristiques des algorithmes génétiques

2.4.1 La Population initial

C'est lors de l'initialisation de l'algorithme génétique que les individus originaux sont générés. Un chromosome est une suite de gêne, chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus. [7]

2.4.2 Le codage

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée). [6]

Généralement on a trois types de codage :

Codage binaire : Chaque gène dispose du même alphabet 0, 1. Les chromosomes sont représentés par une suite de 0 et 1.

Codage réel : Contrairement au codage binaire, il y a des cas ou le codage binaire est impossible, dans ces cas chaque chromosome est représenté par une série de valeurs quelconque du contexte de problème.

Ce type de codage peut être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle [6]

Exemple :

Codage binaire 10010010 10001010

Codage réel 2358HY 8965Sk

Gene1 Gene2

Codage à l'aide d'arbres syntaxiques : Ce type de codage utilise une structure arborescente. Il peut être utilisé lorsque l'ampleur du problème ou de la solution n'est pas limitée. Son inconvénient est qu'il peut trouver de grands arbres de solutions difficiles à analyser.

2.4.3 La fonction objective

C'est une fonction qui associe une valeur de performance à chaque individu ce qui offre la possibilité de le comparer à d'autre individus et permet à l'algorithme génétique de déterminer qu'un individu sera sélectionné pour être reproduit ou pour déterminer s'il sera remplacé. [8]

2.4.4 Les opérateurs génétiques

Pour construire une population k+1 à partir d'une population k on utilise l'opération de sélection, le croisement et la mutation.

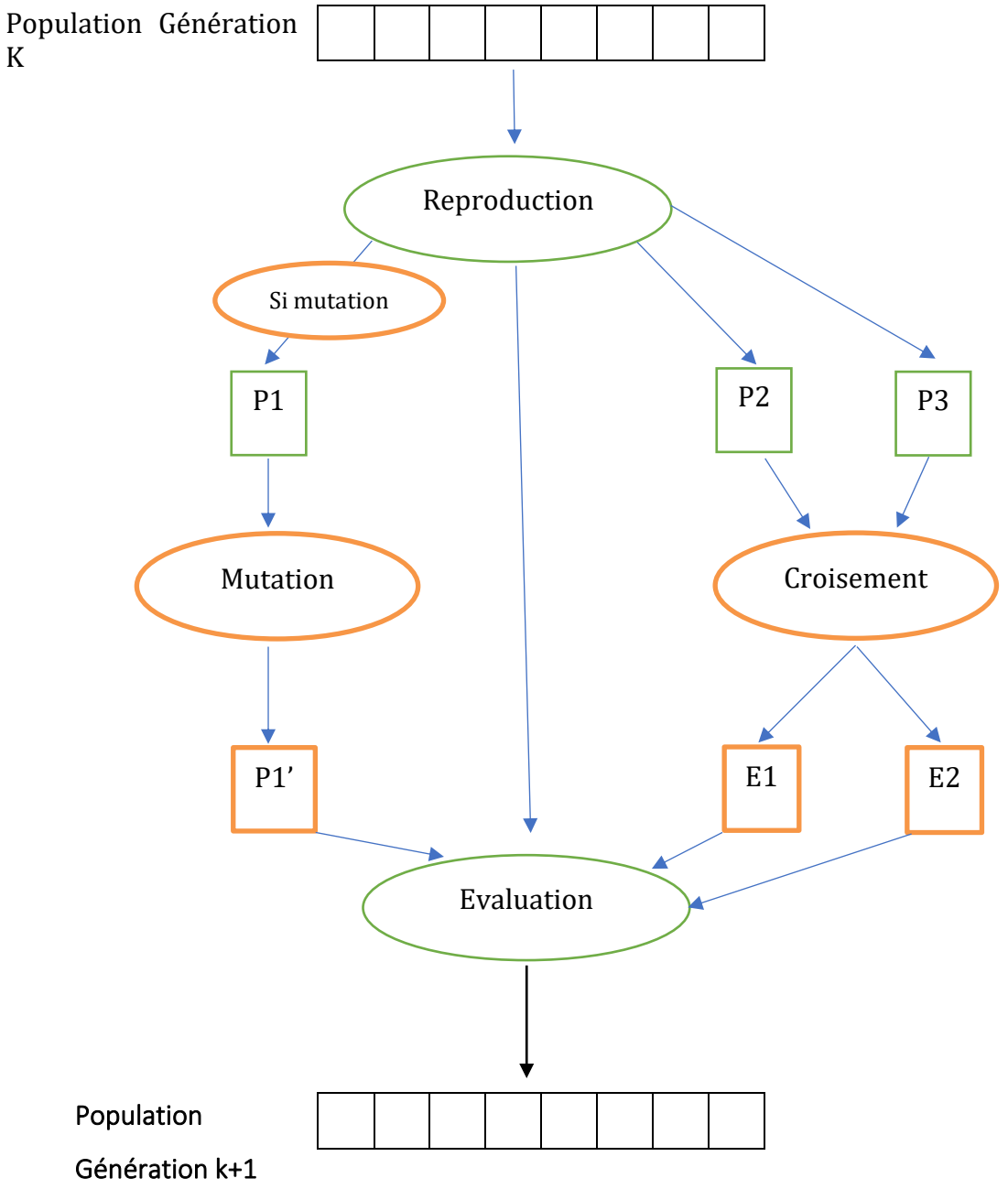


Figure 2.1 Principes générale de l'évolution d'une population d'un algorithme génétique [9]

2.4.4.1 La sélection

Les opérateurs de sélection sont utilisés à deux reprises dans l'algorithme génétique. En effet, il faut d'abord sélectionner les chromosomes qui vont être soumis aux opérateurs de croisement et de mutation, puis il faut sélectionner ceux qui seront conservés dans la génération suivante. Ces deux sélections ne partagent donc pas les mêmes objectifs et ne sont pas soumis aux mêmes critères de choix. [10]

La première sélection permet d'identifier les individus susceptibles d'être croisés dans une population. Il existe plusieurs techniques de sélection. Nous présentons ici les quatre les plus utilisées parmi elles : [11]

Sélection par rang : consiste à ranger les individus de la population dans un ordre croissant ou décroissant, selon l'objectif (fonction *fitness*).

Sélection par roulette : elle consiste à créer une roue de loterie biaisée pour laquelle chaque individu de la population occupe une sélection de la roue proportionnelle à sa valeur d'évaluation.

Sélection aléatoire : cette sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité uniforme $1/P_{taille}$ d'être sélectionné, où P_{taille} est le nombre total d'individus dans la population (la taille de la population).

Sélection par tournoi : le tournoi le plus simple consiste à choisir aléatoirement un nombre k d'individus dans la population et à sélectionner celui qui a la meilleur performance. Les individus qui participent à un tournoi sont remis ou sont retirés de la population, selon le choix de l'utilisateur. Avec le tournoi binaire, sur deux individus en compétition, le meilleur gagne avec une probabilité $p \in [0{,}5 ; 1]$.

La seconde sélection sert à déterminer les chromosomes qui sont conservés dans la génération suivante (clonage). Le choix de cet opérateur a une influence directe sur la complexité de l'algorithme [10]**.** Elle va sélectionner les chromosomes non sélectionnés par la première sélection et les sauvegarder pour les remettre dans la nouvelle population générée.

2.4.4.2 Le croisement

Les opérateurs de croisement permettent de combiner les individus pour obtenir de nouvelles combinaisons de gènes avec un meilleur résultat pour notre fonction objectif. Ils explorent ainsi l'espace de recherche accessible à partir de l'ensemble des gènes disponibles [10]. Il a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants.[5]

Plusieurs opérateurs de croisement sont proposés, nous citons ici les plus utilisés : [11]

- **Croisement en 1-point :**

Consiste à diviser chacun des deux parents en deux parties à la même position, choisie au hasard et à recopier la partie inférieure du parent à l'enfant et à compléter les gènes manquants de l'enfant à partir de l'autre parent en maintenant l'ordre des gènes.

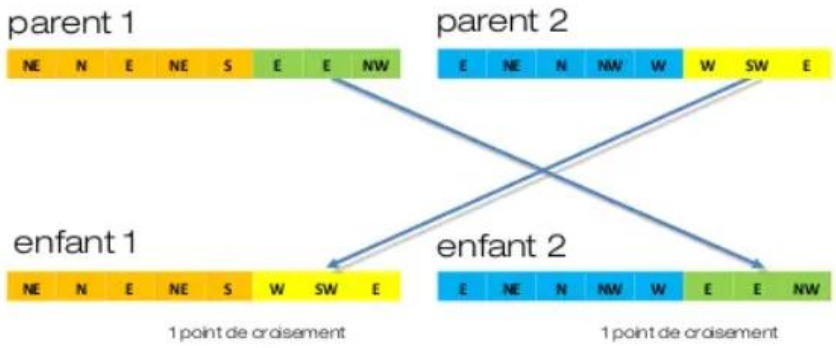


Figure 2.2 Croisement en 1-point de deux chromosomes

- **Croisement en 2-points**

Ce type de croisement est utilisé en choisissant aléatoirement 2 points de coupure pour dissocier chaque parent en 3 fragments. Les 2 fragments en extrémités pour le *Parent1* (respectivement *Parent2*) sont copiés à l'*Enfant1* (respectivement *Enfant2*). On complète la partie restante de l'*Enfant1* par les éléments du *Parent2* et la partie restante de l'*Enfant2* par les éléments du *Parent1* en balayant de gauche à droite et en ne reprenant que les éléments non encore transmis.

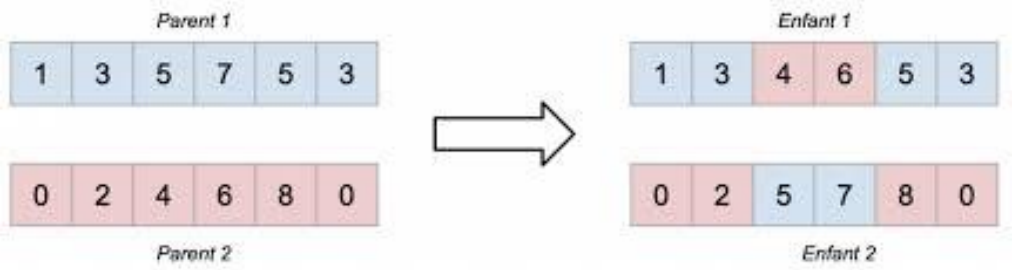


Figure 2.3 Croisement en 2-point de deux chromosomes

Il existe d'autres opérateurs de croisement qui sont proposés dans la littérature comme le croisement PMX(*partial-mapped crossover*), OX (*order crossover*), CX (*cycle crossover*), JOX (*job-based order crossover*), ER (*edge recombination rossover*), etc.

2.4.4.3 La mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. L'opérateur de mutation peut consister à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire. [5]

Le rôle de cet opérateur est de modifier aléatoirement, avec une certaine probabilité, la valeur d'un composant de l'individu. [12]

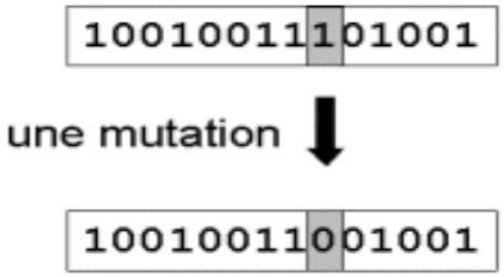


Figure 2.4 Principe de mutation [6]

2.5 Critère d'arrêt

Le test d'arrêt joue un rôle très important dans le jugement de la qualité des individus. Les critères les plus courants pour arrêter les processus sont : [11]

- Arrêt après un nombre fixé à priori de générations.
- Arrêt lorsque la population cesse d'évoluer ou n'évolue plus suffisamment.
- Arrête après certain nombre d'itérations ou un certain temps de calcul.

3. Algorithme génétique en générale

Algorithme génétique
1 initialisation : générer une population initiale P de solutions de taille taille(P) = n ; 2 Coder la population avec une méthode de codage ; Répéter 3 Calculer la valeur fitness de chaque chromosome ; 4 Sélection : choisir selon un pourcentage de sélection des solutions par une technique de Sélection ; 5 Croisement : combiner chaque deux par deux les solutions sélectionner pour former deux solution enfants E1 et E2 ; 6 Mutation : Choisir selon un pourcentage de mutation un chromosome C pour former un nouveau Chromosome C' ; 7 Remplacer E1 et E2 par ces parents et C' par C dans P ; 8 Vérifier P ; Jusqu'à critère d'arrêt satisfait ;

4. Domaine d'Application des AG

Les algorithmes génétiques peuvent être une bonne solution pour résoudre certains problèmes. nous pouvons citer: quelques domaines où iks sont appliqués :

- **Domaine des jeux :**

L'algorithme génétique peut être utilisé pour résoudre le problème du jeu seul. En fait, l'apprentissage via un système de score est très adapté au monde du jeu, car le score est l'élément central de tout jeu et peut classer les joueurs. Il devient plus facile de développer un algorithme génétique, puisque la fonction d'évaluation a été calculée à travers le jeu,

Voici des exemples de jeux qui utilisent l'algorithme génétique : [28]

- Flappy Bird
- Mario

- **Domaine de recherche :**

Comme le problème de voyageur de commerce qui vise à optimiser le trajet de façon à ce que le chemin soit le plus court possible. De plus, ce type de problème peut être facilement codé sous la forme d'un algorithme génétique. L'idée de base est d'utiliser la longueur du chemin comme fonction d'évaluation.

- **La sécurité :**

En raison du développement des applications multimédia, les images, les vidéos et les audios sont transférés d'un endroit à l'autre sur Internet. Il a été constaté dans la littérature que les images sont plus sujettes aux erreurs pendant la transmission. Par conséquent, des techniques de protection des images telles que le cryptage, le filigrane et la cryptographie sont nécessaires. Les techniques classiques de cryptage d'images nécessitent des paramètres d'entrée pour le cryptage. Une mauvaise sélection des paramètres d'entrée génère des résultats de cryptage inadéquats. L'AG et ses variantes ont été utilisées pour sélectionner les paramètres de contrôle appropriés. Kaur et Kumar [29]

- **Domaine de la médecine :**

Les algorithmes génétiques ont été appliqués à l'imagerie médicale, comme la détection des bords en IRM et la détection des nodules pulmonaires dans les images de tomodensitométrie [30, 35]. Dans [32], les auteurs ont utilisé une technique de correspondance de gabarit avec GA pour détecter les nodules dans les images CT. Dans [31] ont utilisé une méthode de croissance de régions basée sur les AG pour détecter les tumeurs cérébrales. Les AG ont été appliqués à des problèmes de prédiction médicale à partir de sujets pathologiques. Dans [32] ils ont utilisé des AG pour résoudre les problèmes qui se posent en biomécanique. Il est utilisé pour prédire les pathologies lors d'un examen.

5. Conclusion

L'optimisation combinatoire fournit un ensemble de solutions (précises et approximatives) qui peuvent résoudre des problèmes difficiles dans un court laps de temps, et a une bonne qualité de solution. Nous avons cité le panorama méta-heuristique pour la réingénierie d'une application orientée objet vers une architecture orientée service, cette méta-heuristique est appelée algorithme génétique. Le concept de cet algorithme permet de développer des solutions très efficaces à un problème donné, dans un temps assez court. Le principal avantage de cet algorithme est qu'il n'a pas besoin d'exemples de départ pour apprendre, aucune base n'est requise pour l'apprentissage.

Dans le chapitre suivant nous allons expliquer comment nous avons adapté l'algorithme génétique pour la réingénierie d'application OO vers une SOA et plus précisément pour l'identification des services.

Chapitre 03 : La Conception

1. Introduction

Les systèmes logiciels doivent continuer à évoluer pour répondre aux nouvelles demandes. Aujourd'hui, la technologie SOA et les services Web sont de nouvelles approches de concevoir des logiciels interopérables qui permettent l'intégration d'applications métier internes et externes. La migration des systèmes logiciels existants vers une architecture orientée services (SOA) est un moyen de moderniser ces systèmes. La SOA permet de développer des systèmes inter-organisationnels complexes en intégrant et en combinant des services réutilisables, distribués, relativement indépendants et généralement hétérogènes. [33]

Notre travail rentre dans le cadre de la réingénierie des applications orientée objet vers une nouvelle architecture orientée service.

Il existe de nombreuses stratégies et méthodes pour migrer vers la SOA, malgré qu'elles puissent diverger, elles s'accordent toutes sur l'importance d'une étape de base, qui est l'étape d'identification des services.

Les applications orientées objet paressent prédisposées pour la migration vers une architecture SOA, puisqu'ils sont déjà composés de modules réutilisables qui sont les classes. Malheureusement les classes sont généralement de granularité fine, fortement couplées et difficilement dissociables de leur environnement.

Le but principal de notre travail est d'automatiser la phase d'identification de service à partir d'applications orientées objet afin de faciliter la migration vers une nouvelle architecture SOA.

2. L'objectif global

Une application orientée objet est composée d'un ensemble de classes. Afin de migrer vers une architecture orientée services, nous devons combiner des classes fortement couplées et fortement cohésives. Ces classes peuvent former un module autonome, réutilisable et distribuable. Ce nouveau module composé de classes et appelé service et doit impérativement respecter les propriétés des services de la SOA.

Pour atteindre cet objectif, nous avons pensé à modéliser notre problème à un problème de regroupement. Nous avons commencé à explorer les algorithmes de classification

pouvant nous permettre d'atteindre cet objectif. Puis nous nous sommes intéressés plus particulièrement aux algorithmes de classification non supervisés, car nous ne pouvons pas connaître à l'avance le nombre des services identifiés.

Afin d'automatiser l'identification des services à partir du code source objet, nous avons choisi d'adapter l'algorithme génétique décrit dans le chapitre précédent pour trouver les meilleurs regroupements de classes qui peuvent constituer des services de qualité qui respectent les caractéristiques de la SOA.

3. Travaux dans le contexte

Puisque nous avons choisi de modéliser notre problématique sous forme de problème de regroupement en utilisant l'algorithme génétique (AG), nous avons commencé à explorer les travaux effectués dans ce domaine. Plusieurs idées ont été proposées, nous citions quelques-unes :

Dans le domaine de clustering, [34] a proposé une nouvelle combinaison d'algorithmes génétiques et de K-means. L'algorithme génétique est capable de partitionner un ensemble de données en une série de groupes qui ne sont pas connus à l'avance. L'approche montre une représentation basée sur les étiquettes et utilise la technique des K-means pour améliorer la génération de la progéniture à l'aide de croisements basés sur les groupes. Cela signifie que la méthode a avancé une population de chromosomes, chacun représentant une division d'objets en une variété différente de clusters. Un croisement basé sur les groupes a été amélioré avec un opérateur de K-means en une étape, puis une approche de mutation afin d'assigner à nouveau les objets en fonction des clusters de la mauvaise gamme d'objets à l'imitation des clusters calculés jusqu'à présent.

Dans [35] Ils ont proposé une nouvelle technologie de modèle de type minoritaire basée sur l'algorithme génétique et le regroupement de Kmeans. La dimension de reconnaissance de la classe minoritaire est d'autant plus importante qu'elle est commune. Ils utilisent des classifieurs de configuration typiques et des erreurs pour améliorer les performances globales de classification des classes minoritaires. Cette méthode utilise l'algorithme K-means pour regrouper un petit nombre de types d'échantillons, puis utilise un algorithme génétique pour gagner de nouveaux échantillons dans chaque cluster, et utilise un trieur, SVM et KNN pour une confirmation efficace. Les résultats expérimentaux confirment que la formation d'un grand nombre d'échantillons peut favoriser de

meilleurs résultats de classification, ce qui est très différent de la croissance et de l'augmentation des échantillons minoritaires.

Dans [36] ils ont proposé l'outil de clustering Bunch qui utilise le hillclimbing et les algorithmes génétiques pour regrouper les classes/interfaces en clusters. L'entrée de l'outil est un graphe basé sur les classes dans lequel les nœuds représentent les classes/interfaces et les arêtes représentent les dépendances entre ces classes/interfaces. Ces classes/interfaces sont regroupées sur la base d'une fonction de qualité de modularisation. Cette fonction mesure la qualité des solutions de clustering du graphe d'entrée de manière quantitative en tant que compromis entre l'inter connectivité (c'est-à-dire les dépendances entre les classes/interfaces de différents clusters) et l'intra connectivité (c'est-à-dire les dépendances entre les classes/interfaces du même cluster).

Dans le domaine d'identification des services, [37] ont utilisé des algorithmes génétiques pour identifier les services dans le code source hérité. Ils ont proposé une technique d'identification basée sur les arbres de recouvrement. Ils ont utilisé ces représentations pour fournir aux développeurs un ensemble de solutions possibles pour le problème d'identification. Ils ont également utilisé un algorithme génétique multi-objectif pour affiner l'ensemble initial de décompositions de services. L'algorithme génétique multi-objectif s'est appuyé sur une fonction d'adaptation qui prend en compte un ensemble d'objectifs de gestion pour obtenir une solution optimale au problème d'identification des services.

Dans [38], ils ont également proposé une approche d'identification des services basée sur l'algorithme génétique. Cependant, ils ne prennent en compte que la cohésion fonctionnelle d'un ensemble de modules des systèmes existants.

Le principe de l'algorithme génétique est resté toujours le même et basé sur trois étapes principales qui sont la sélection, le croisement et la mutation.

Nous avons remarqué que la différence entre les travaux de recherche cités, est dans la définition de chaque caractéristique de l'algorithme.

Travail	L'initialisation				
	Codage	Population	Sélection	Croisement	Mutation
[34]	Un chromosome est constitué d'une chaîne de longueur égale au nombre n d'objets. Chaque objet est situé à une position i du chromosome et est associé à un nombre entier dans l'alphabet {2, ... , kmax}, où kmax est le nombre maximum de clusters possibles et chaque entier est l'étiquette d'un cluster.	/	/	Choisir une position de couper aléatoire h entre 1 et n. Générer les enfants. Appliquer KMO aux enfants. Si un cluster singleton est présent, assigner son objet à l'un des clusters existants au hasard.	Vérifie si la distance entre xi et le cluster Ca auquel il appartient est supérieure à la distance entre xi et un autre cluster Cb. Dans ce cas, xi est retiré de Ca et ajouté à Cb.
[35]	Binaire	/	/	Pourcentage 80%	/
[36]	/	N	Roulette (avec la fonction Modularisation Qualité)	/	/
[38]	Binaire	/	Roulette (avec la fonction Jaccard distance metric)	Pourcentage 70%	Pourcentage 1%

Tableau 3.1 l'initialisation des paramètres de l'AG dans les travaux cités.

Dans le reste du chapitre nous allons présenter notre approche pour l'identification des services en utilisant l'AG.

4. Processus d'identification des services

Pour atteindre notre objectif, il faudrait bien répartir l'ensemble des fonctionnalités fournies par une application orientée objet globale en un ensemble de fonctionnalités cohérentes contenues dans des modules appelés services.

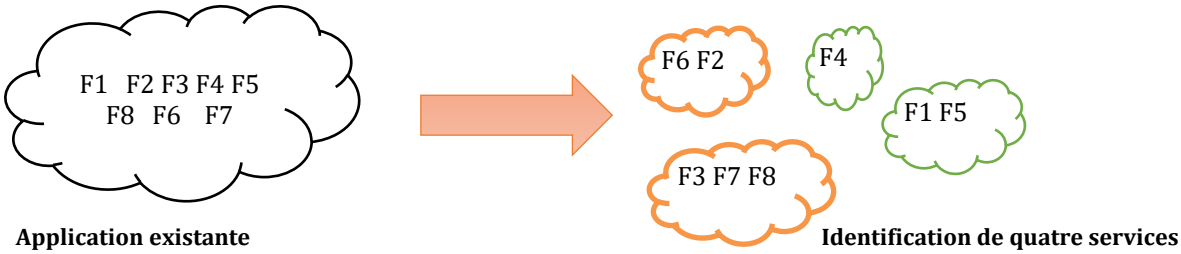


Figure 3.1 l'identification des services

Allons de ce principe nous avons fixé les phases suivantes :

Phase 1 : La mise en correspondance Objet-Service : Un travail préalable avant de définir le processus d'identification du service est indispensable : il s'agit d'identifier les caractéristiques des deux éléments de base du formalisme, classe (ou objet) et service, afin de faire la correspondre entre ces derniers.

Nous avons constaté que le service est un module qui fournit un ensemble de fonctionnalités cohérentes, en général de granularité assez large. Ces fonctionnalités sont fortement couplées entre elles mais faiblement couplé avec l'extérieur, ce qui donne la possibilité au service d'être autonome et facilement réutilisable.

De ce fait nous avons proposé que le service soit composé d'une ou de de plusieurs classes qui fournissent des fonctionnalités cohérentes, fortement couplées entre elles et qui sont faiblement couplées avec le reste des classes. Ce qui nous ramène à regrouper les classes qui remplissent les conditions citées pour former le service.

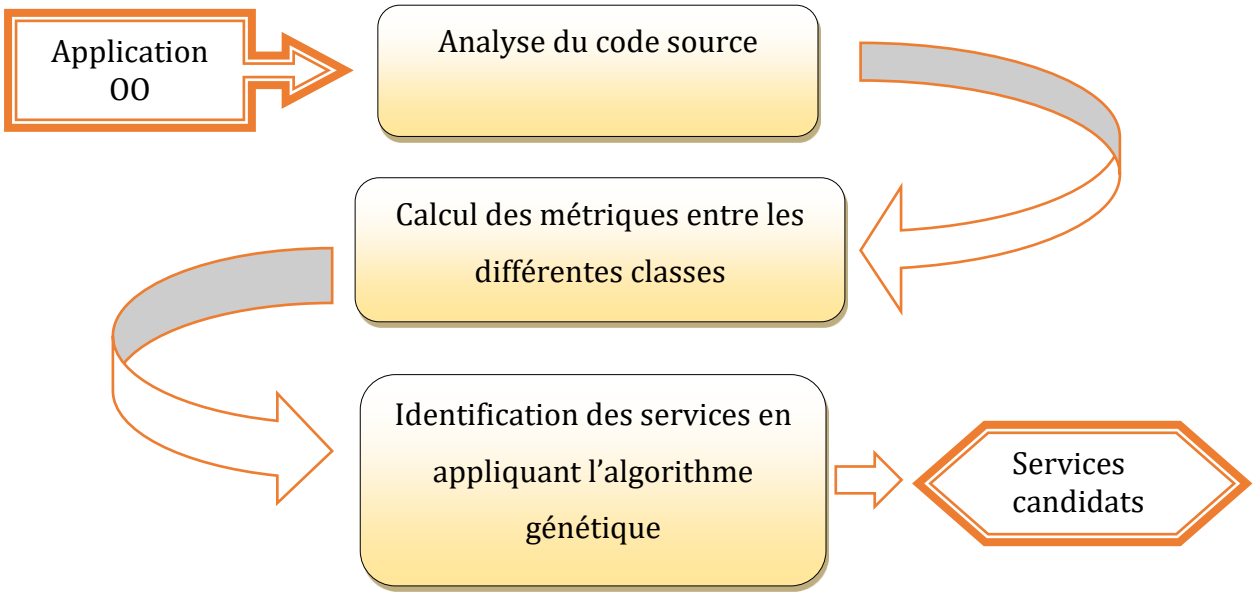


Figure 3.2 : Etapes d'identification des services.

Phase 2 : Analyse du code source : Dans cette étape nous procédons à l'analyse de l'application orientée objet afin d'extraire toutes les informations qui vont nous aider à étudier les relations entre les classes, les expressions, les appels des méthodes, les utilisations des attributs…etc, afin de décider des groupes de classes qui vont former un service de qualité.

Les informations extraites du code source seront utilisées pour calculer la fonction basée sur le couplage et la cohésion, ce qui permet d'indiquer le degré de proximité entre les classes (elle sera détaillée ultérieurement).

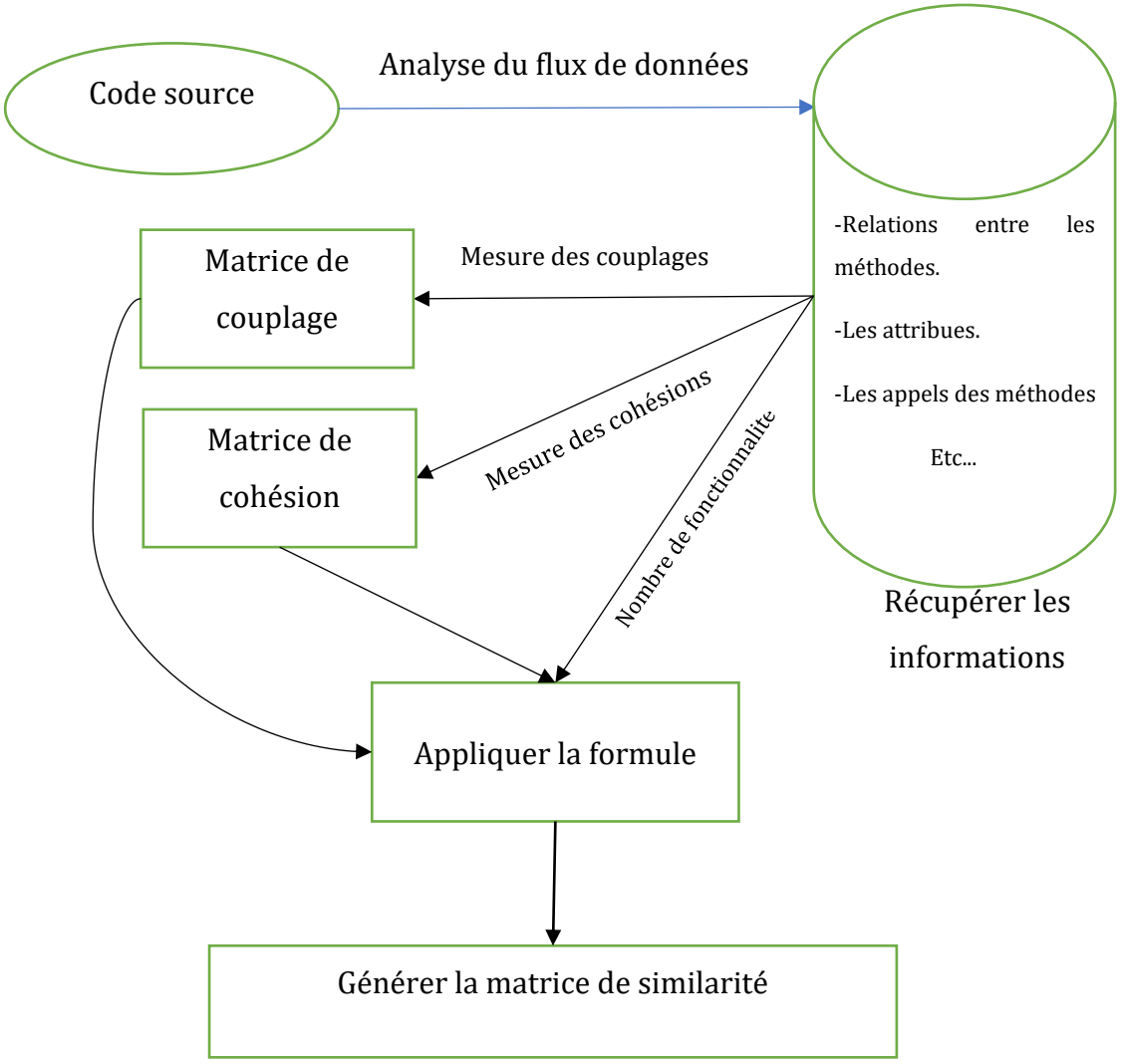


Figure 3.3 Structure d'analyse de code source.

Phase 3 : Identification des services : Dans cette phase, nous appliquons l'algorithmes génétique pour décomposer l'application OO en un ensemble de services, en regroupant les classes qui sont fortement reliées (couplées) et offrant des fonctionnalités cohérentes.

5. Application de l'algorithme génétique

Cette étape c'est l'étape la plus importante dans notre travail, elle porte sur l'adaptation de l'algorithme génétique afin d'identifier les services candidats d'une application.

On a déjà présenté l'algorithme génétique et ses composants essentiels dans le chapitre précédent, dans ce chapitre nous allons adapter cet algorithme pour trouver une solution optimale à notre problème d'identification de services.

5.1 Le codage proposé

Dans un algorithme génétique, un individu représente une solution au problème. Dans notre problème nous proposons de représenter un individu par un ensemble de gènes où chacun comporte un groupe de classes qui constituent un service potentiel ou candidat. Chaque gène est présenté par un ensemble d'entiers et chaque entier correspond à un identificateur d'une classe.

Le nombre de gènes est variable mais l'ensemble des éléments qui composent les gènes est toujours fixe de taille n qui est le nombre de classes de l'application OO.

Donc dans notre cas nous avons choisi le codage réel.

Dans l'exemple suivant nous présentons un individu (chromosome) composé de trois gènes (services) où chaque gène à des classes différentes :

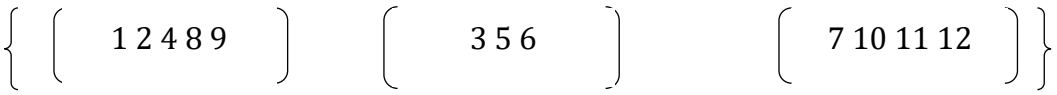


Figure 3.4 Le codage d'un individu

Chaque nombre est l'identificateur d'une classe et chaque gène est un service.

5.2 Initialisation des paramètres de l'Algorithme Génétique

Paramètres	Valeur	Valeur par défaut
Taille de la population	N Nombre de classes	N Nombre de classes
Nombre de générations	Fixé par l'utilisateur	10 *N
Pourcentage de Sélection	Fixé par l'utilisateur	50% de N
Pourcentage de mutation	Fixé par l'utilisateur	1% du Nombre de générations

Tableau 3.2: Valeurs des paramètres de l'algorithme génétique

5.3 Population initiale

Le choix de la population initiale joue un rôle important dans la vitesse de convergence de l'AG, elle doit être la plus représentative possible de l'ensemble des individus.

Nous avons choisi de créer une population assez diversifiée allant de services composés d'une seule classe jusqu'à un service composé de l'ensemble des classes de l'application OO initiale, et la taille de la population doit être au minimum au nombre des classe de l'application.

Nous avons suivi les étapes suivantes pour créer la population initiale :

Si N = le nombre de classe de l'application OO alors ;

- Chaque gène représente un service ;
- Chaque service peut être composé de 1 à N classes ;
- Le nombre de gène est variable, mais le totale des éléments des gènes égale toujours à N ;
- Une classe doit paraitre une seul fois dans un individu ;
- Toutes les classes doivent paraitres dans un individu ;
- Le premier individu : sera composé de N gènes (service) d'une classe.
- Le deuxième individu : sera composé de N/2 services où chaque service contient deux classes ;
- Le troisième individu : sera composé de N/3 services où chaque service contient trois classes ;
- Jusqu'à le N ième individu : qui sera composé d'un seul service qui contient toutes les classes.

Notre population sera représentée comme suit :

Pop= { I1= [1] [2] [3] [4] [5] [6] …… [N]
I2= [1 2] [3 4] [5 6] …… . [N-1 N]
I3= [1 2 3] [4 5 6] … [N-2 N+1 N]
. . .
In= [1 2 3 4 5 6 …… … … … N] }

Figure 3.5 La population initial

Tel que chaque ligne représente un individu.

Nous avons choisi cette forme de la population initiale afin de couvrir toutes les possibilités pour un service.

5.4 Fonction objective proposée (Fitness)

Après la création de la population initiale, il faut évaluer les chromosomes pour déterminer ceux qui vont sélectionner après.

Pour évaluer les chromosomes nous avons besoin d'une fonction Fitness.

Dans notre travaille nous avons proposé une fonction pour calculer la valeur fitness de chaque chromosome. cette dernière utilise la formule proposée dans le travail de [21].

Voici la fonction utilisée pour calculer la valeur fitness de chaque chromosome.

$$Fitness(X) = \frac{\sum_{i=1}^{Nbgene}(\mathrm{F}(\mathrm{G}_i)}{Nbgene}$$

Ou X c'est un chromosome,

Nbgene c'est le nombre de gènes (services) dans le chromosome X.

$F(G_i)$, c'est la moyenne des valeurs de similarité entre toutes les classes du Gène i par paire

$F(G_i)$= Moyenne ($M(a_i,b_i)$)

$M(a_i,b_i)$: c'est la valeur de similarité entre la classe a_i et la classe b_i qui appartiennent au même gène i

La valeur de similarité entre la classe A et B est calculée à partir du couplage, de la cohésion et de la moyenne des interface exposées vers l'extérieur.

$$M(A,B) = \frac{NB\_FCT\ (A,B) + (1 - Cohésion\ (A,B)) + (1 - Couplage(A,B))}{3}$$ [21]

Ou A, B : deux classes.

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

Cohésion(A,B) : La cohésion interne entre A, B.

Couplage(A,B) : Le couplage entre A, B.

Toutes les valeurs de similarité entre les classes seront calculées et sauvegardées dans la matrice carrée de similarité, puis elles vont être utilisées dans notre fonction fitness pour calculer les valeurs fitness des chromosomes.

Exemple d'application :

On a l'individu suivant : X = {[1 2 5 6 9] [11 3 4 7]} on va calculer sa valeur Fitness comme suit :

Cet individu est composé de deux gène G1= [1 2 5 6 9] et G2= [11 3 4 7] premièrement on va calculer la somme des valeurs des similarités pour chaque gène.

$$F(G1) = \frac{(\mathrm{M}(1,2) + \mathrm{M}(1,5) + \mathrm{M}(1,6) + \mathrm{M}(1,9) + \mathrm{M}(2,5) + \mathrm{M}(2,6) + \mathrm{M}(2,9) + \mathrm{M}(5,6) + \mathrm{M}(5,9)\ \mathrm{M}(6,9))}{1+2+3+4}$$

$$F(G2) = \frac{(\mathrm{M}(11,3) + \mathrm{M}(11,4) + \mathrm{M}(11,7) + \mathrm{M}(3,4) + \mathrm{M}(3,7) + \mathrm{M}(4,7)\ /(1+2+3).}{1+2+3}$$

Donc $Fitness(X) = \frac{\mathrm{F(G1)}+F(G2)}{2}$

5.4.1 Formule de similarité

Comme nous l'avons mentionné, notre fonction fitness s'est basée sur la formule de similarité proposée par [21], nous détaillons dans ce qui suit les éléments de cette formule :

$$M(A,B) = \frac{NB\_FCT\ (A,B) + (1 - Cohésion\ (A,B)) + (1 - Couplage(A,B))}{3}\ [21]$$

5.4.2 Calcul du couplage

Le couplage est le degré d'interdépendance entre modules. Cette notion est très utilisée dans le génie logiciel et en particulier dans le paradigme objet [10].

On dit qu'une classe est couplée avec une autre classe si :

- Elle est en relation avec une ou plusieurs parties de code source de celle-ci.
- Une classe est hériter d'autre classe. (Couplage fort)

Il existe plusieurs formules qui permettant de mesurer le couplage entre les classes, mais [21] ont combiné entre trois formules dans une formule, ces trois formules sont :

- NIH_ICP [22] : mesure le nombre d'appels de méthodes des classes avec lesquelles une classe n'a pas de relation d'héritage.

- DAC [23] : mesure le nombre d'attributs de la classe qui ont pour type une autre classe.
- OCMIC [24] : mesure le nombre des paramètres qui ne sont pas de type a ou primitifs dans les méthodes de la classe a.

D'après [21] la première formule proposée pour calculer le couplage :

Couplage (A, B) = NIH_ICP (A, B) + DAC (A, B) + OCMIC (A, B).

Ou A, B sons deux classes. (Dans le cas de l'héritage la valeur de couplage doit être la plus grande valeur).

Cette formule ne permet pas de déterminer le dégrée de dépendance d'une classe envers son environnement c'est pour cette raison [21] ont calculés le couplage par rapport au nombre totale des relations d'une classe sans compter les relations d'héritage.

La formule finale pour calculer la matrice de couplage proposé par [21].

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

Figure 3.6 La formule de couplage proposé dans [21]

Ou Ai représente une classe.

5.4.3 Calcul de la cohésion

La cohésion mesure la force de la collaboration au sein d'un ensemble d'éléments. Elle doit mesurer la cohésion d'un ensemble de classes ainsi que la cohésion d'une classe [10].

La mesure de cohésion prend en compte les relations directes et les relations indirectes.

- On dit que deux méthodes sont directement connectées si :
  - Elles partagent au moins une variable d'instance en commun (relation UA : usage d'attributs).
  - Interagissent au moins avec une méthode de la même classe (relation IM invocation de méthodes).
- On dit que deux méthodes sont indirectement connectées si :

- Elles sont directement ou indirectement reliées à une méthode.

D'après [25] la cohésion est basée sur

- Les relations directs est définir par : DC_D=|ED| / [n*(n-1)/2].
- Les relations indirectes est défini par : DC_I= |EI| / [n*(n - 1) / 2].

Tel que

DC_D : donne le pourcentage de paires de méthodes publiques qui sont directement reliées.

DC_I : donne le pourcentage de paires de méthodes publiques qui sont directement ou indirectement reliées.

n c'est le nombre des méthodes.

ED le nombre des relations directe.

EI le nombre des relations indirecte.

La formule proposée dans [21] pour calculer la cohésion d'une classe C :

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

5.4.4 La moyenne des interface exposées vers l'extérieur

C'est le nombre des fonctionnalités public n'étant pas appelé à l'intérieur du service diviser par le nombre total des méthodes. Le nombre des fonctionnalités est calculer ou milieu de calcul de la cohésion.

5.5 La Sélection

Pour chaque individu nous allons lui calculer sa valeur fitness puis trier les individus selon l'ordre décroissant des résultats. Après nous avons choisi la par rang qui a pour rôle de sélectionner les individus en fonction de leurs Fitness.

Le calcul de la valeur fitness, nécessite le calcul de la formule des similarités entre classes basés sur les métriques de couplage, de cohésion et du nombre de fonctionnalités exposées vers l'extérieur.

Exemple :

Voici la population suivante qui est composée de six individus (6 classes) :

Pop = {
I1= [1] [2] [3] [4] [5] [6]
I2= [1 2] [3 4] [5 6]
I3= [1 2 3] [4 5 6]
I4= [1 2 3 4] [5 6]
I5= [1 2 3 4 5] [6]
I6= [1 2 3 4 5 6]
}

Figure 3.7 La population initial

Nous allons calculer la valeur Fitness de chaque individu dans la population.

Fitness(I1) = 0

Fitness(I2) = 0,9

Fitness(I3) = 0,7

Fitness(I4) = 0,3

Fitness(I5) = 0.1

Fitness(I6) = 0.5

Nous allons faire une sélection avec un pourcentage de 50% de la taille de la population.

- Trier les individus selon l'ordre décroissant de leurs Fitness (I2, I3, I6, I4, I5, I1)
- Sélectionner quatre individus (Le pourcentage 50% nous donne juste trois individus, nous ajoutons un individu car le croisement exige un nombre pair des individus). Après la sélection nous avons obtenir (I2, I4, I6, I3) sur lesquels nous allons appliquer le croisement.

5.6 Le Croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. Pour effectuer ce croisement sur des chromosomes constitués de M gènes, ont divisé les deux parents P1, P2 sur la moitié de chaque individu. On échange ensuite les deux sous-chaînes premières de chacun des deux chromosomes, ce qui produit deux enfants.

Nous avons choisi de faire le croisement a un point, plus précisément au point du milieu de chaque chromosome. Nous avons utilisé la division entière pour déterminer le point de croisement des individus.

Après la génération des nouveaux enfants, nous pouvons trouver des classes redondantes ou des classes manquantes, ce qui nécessite une opération de vérification et de rectification des nouveaux chromosomes

Voici ci-dessous un schéma descriptif du croisement : (croisement entre les chromosomes I2, I3 de la population de la (figure 3.3)).

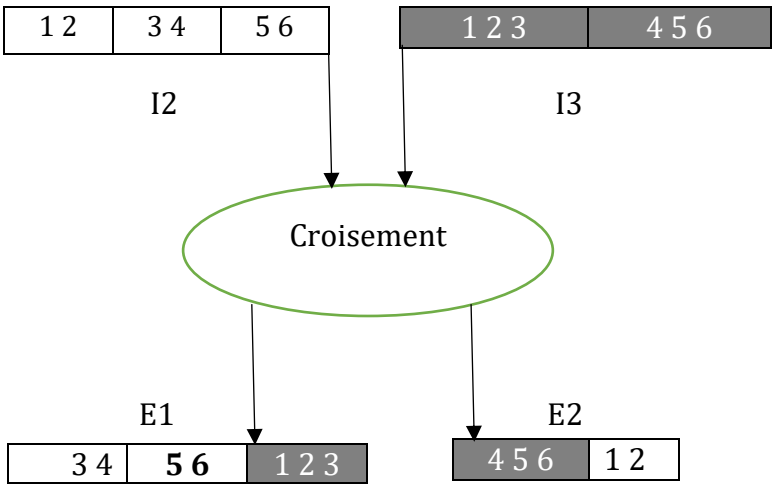


Figure 3.8 croisement a un point

Une étape de vérification et de rectification est nécessaire dans ce cas. Puisqu'une redondance et un manque sont apparue.

5.7Mutation

La mutation a pour but de garantir l'exploration de l'espace d'état et améliorer la qualité de la nouvelle génération.

Pour faire la mutation, nous tirons aléatoirement un gène dans le chromosome puis nous le remplaçons avec une valeur de façon aléatoire.

Nous avons proposé une fonction qui prend en charge cette opération et qui procède aussi à la vérification et la rectification du nouveau chromosome, si une redondance ou un manque de classe est détecté.

Exemple :

Nous avons tiré aléatoirement le chromosome I4 dans la population de la (figure 3.3).

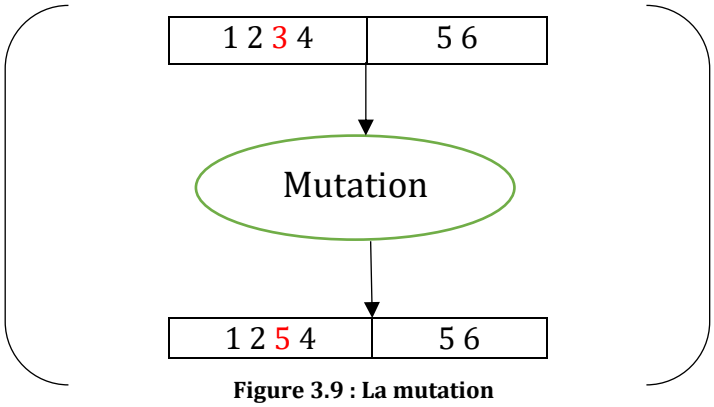


Figure 3.9 : La mutation

Dans ce cas nous avons besoin de rectifier le chromosome puisqu'une redondance et un manque sont apparues.

Remarque :

Les individus croisés et les individus mutés sont ensuite vérifiés avant l'insertion dans la nouvelle population. Pour vérifier les chromosomes on utilise l'idée de [10] qui consiste à supprimer la redondance des classes ou ajouter les classes manquantes dans le chromosome comme un nouveau service (nouveau gène).

Exemple :

a) Cas de redondance de classe

Dans (Figure 3.9) il y a une redondance de classe 5, voici le chromosome avant et après la vérification

Avant :

1 2 5 4	5 6

Après :

1 2 5 4	6

La suppression d'allèle dans un gène se fait comme suite :

- Si nous avons trouvé deux gènes de même taille (nombre des classes) on supprime un allèle quelconque

- Si nous avons trouvé deux gènes de différente taille on supprime l'allèle du gène qui a la taille inférieure ce qui permet de conserver des gènes (services) avec le maximum de classes.

b) Cas de manque de classes

Dans le chromosome précédent il y a un manque de la classe 3, après la vérification on procède à son ajout dans un nouveau gène :

Avant :

1 2 5 4	6

Après :

1 2 5 4	6	3

6. Conclusion

Dans ce chapitre nous avons expliqué comment faire migrer une application orientée objet vers une architecture orientée service en utilisant l'algorithme génétique.

Nous avons ramené le problème à résoudre à un problème de regroupement. Puisque le nombre de groupes résultants n'est pas connu à l'avance nous avons dû choisir un algorithme de regroupement non supervisé, dans notre cas l'AG. Le plus grand défi a été rencontré dans la modélisation de la population et dans la définition de la fonction fitness qui doit fournir une mesure qui représente la qualité d'un service qui respecte au mieux les exigences de la SOA.

Un autre enjeu consistait à garder la taille du chromosome fixe (pour les nombres des classe) tout en utilisant un nombre de gènes variable, ce qui nécessitait une constante vérification après chaque opération de croissement et de mutation.

Afin de garder la solution la plus flexible possible nous avons choisi de permettre le changement de tous les paramètres suivant les besoins de l'utilisateur, tout en proposant des valeurs par défaut.

Les détails techniques de l'implémentations de notre solution feront l'objet du chapitre suivant.

Chapitre 04 : L’implémentation

1. Introduction

Dans ce dernier chapitre, nous allons présenter les détails d'implémentions de notre travaille. Notre but est d'automatiser l'identification des services à partir d'une application orientée objet et plus particulièrement en Java.

Tout d'abord nous présenterons les outils de développement que nous avons utilisé pour implémenter notre application qu'on a baptisé SIGA (Service Identifier based on Genetic Algorithme), ensuite nous allons présenter les interfaces et les fonctionnalités de notre logiciel afin de faciliter son utilisation.

Enfin, nous présenterons un exemple à travers lequel nous comparerons les résultats suivant des différents paramètres de l'algorithme génétique.

2. Les outils de développement

2.1 Java

Nous avons choisi le langage Java car il est plus simple, avec une syntaxe beaucoup plus lisible que C, C++ ou n'importe quel autre langage. Java est complètement orienté objet, il vous permet et vous pousse même à développer vos applications d'une façon orientée objet et vous permet d'avoir une application bien structurée, modulable, maintenable beaucoup plus facilement et de façon efficace. Cela augmente une fois de plus votre productivité. Il est aussi assez puissant et fiable avec l'éditeur Eclipse, qui offre des interfaces faciles à manipuler

2.2 ASTParser

ASTParser est un analyseur syntaxique qui génère un arbre syntaxique abstrait (AST) qui représente entièrement la source du programme. L'AST est constitué de plusieurs nœuds contenant chacun des informations connues sous le nom de propriétés structurelles. L'analyseur syntaxique (ASTParser) permet d'analyser et de découper les différents éléments du langage Java. Il permet ainsi d'extraire les différentes composantes d'une classe donnée (le nom des classes, les importations, les méthodes, les appels, les variables, etc).

Nous avons utilisé cette bibliothèque afin de récupérer toutes les informations possibles sur une application java pour calculer le couplage et la cohésion entre les classes. Les deux métriques sont utilisées pour calculer la fonction objective dans l'algorithme génétique.

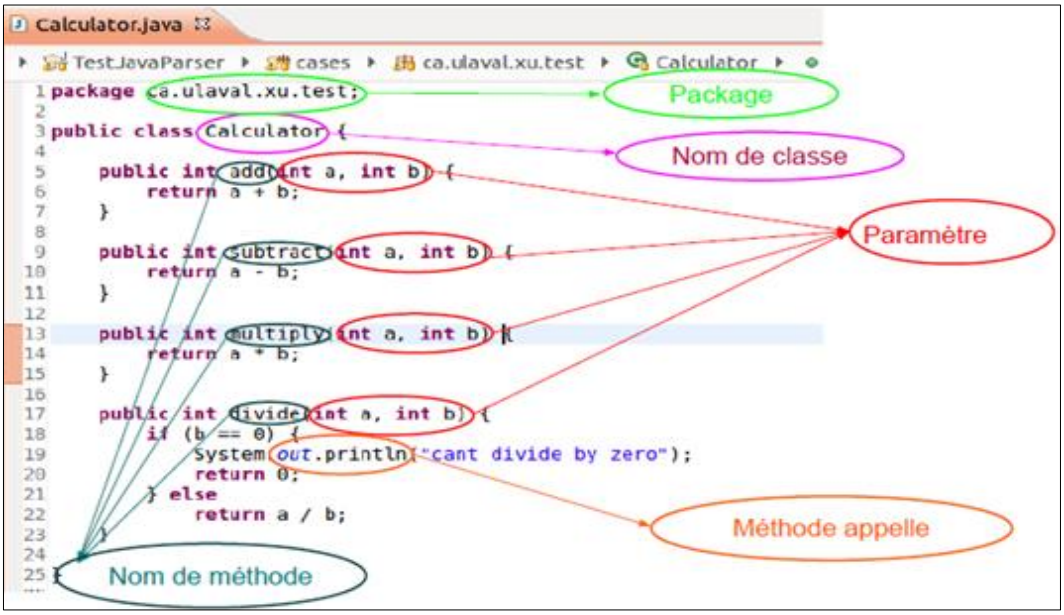


Figure 4.1 : Les composantes d'une classe java.

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

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

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

Tableau 4.1 : Quelques éléments en java et leur correspondance en ASTParser.

3. L'architecture globale de notre application

L'utilisateur choisit au début soit de charger un projet java (orienté objet), soit de charger directement une matrice de similarité ou bien introduire une matrice de similarité à partir de l'interface. Si l'utilisateur charge un projet java, ce projet vas être analysé par l'analyseur ASTParser qui va extraire toutes les informations nécessaires sur la relation des éléments de l'application (classes, méthodes, attributs, appels, …) et les envoyer au package de couplage et de cohésion afin de calculer les métriques qui vont être utilisées dans la formule de similarité entre classes.

Ainsi nous aurons deux matrices une pour la cohésion, et une pour le couplage. Une fonction objective est appliquée à ces matrices tout en récupérant d'autres information sur les interfaces externe afin de générer une nouvelle matrice de similarité qui contient la similarité entre toutes les classes.

Pour identifier les services l'utilisateur doit exécuter le package d'identification de service. Ce package permet d'appliquer l'algorithme génétique pour proposer à la fin à l'utilisateur des services candidats.

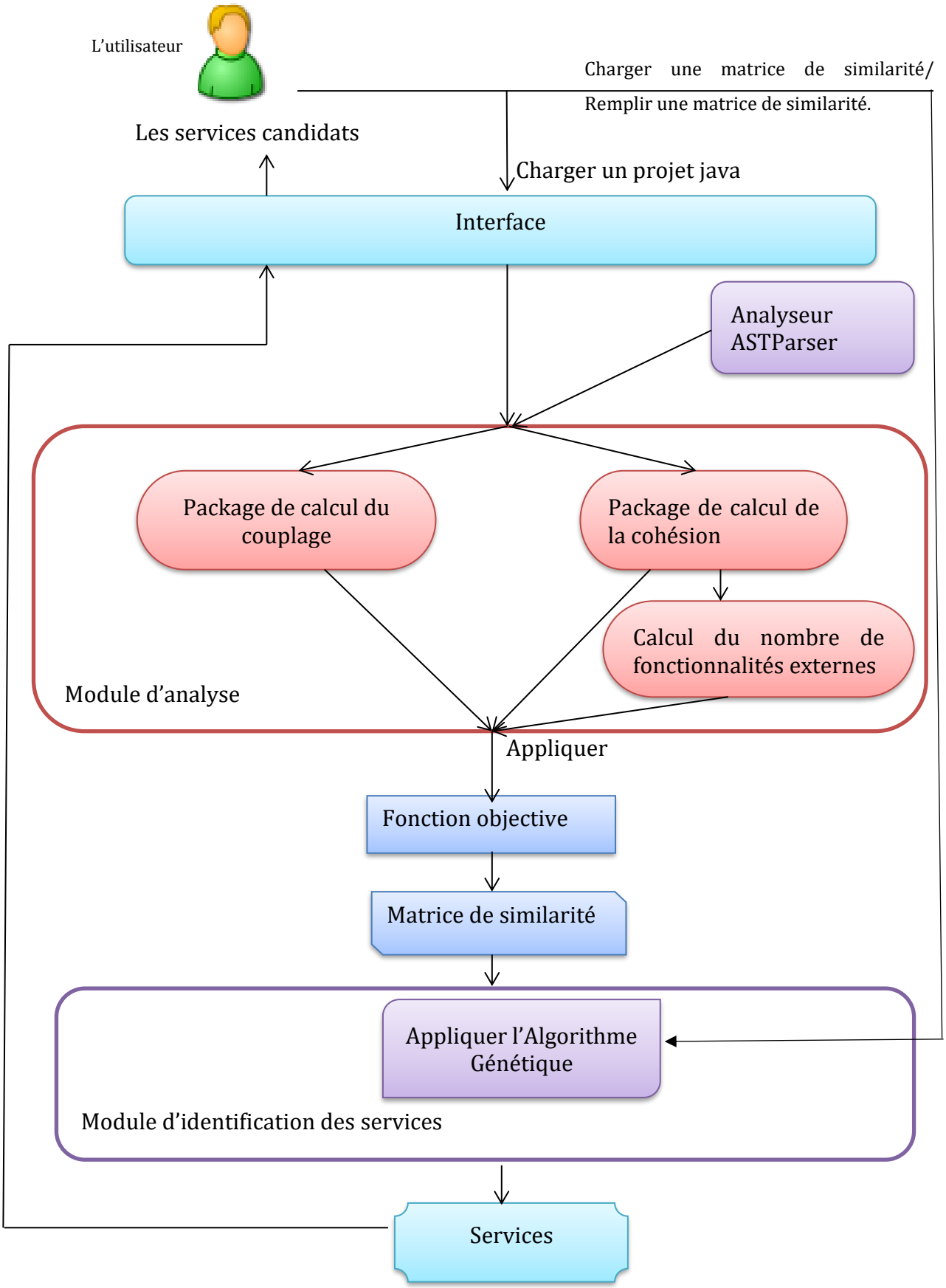


Figure 4.2 : Architecture globale de notre application.

4. L'interface de l'application

Notre interface est composée de trois parties : une pour l'accueil, la deuxième pour charger l'application, et la dernière pour lancer l'algorithme génétique qui vas identifier les services. (Voir figure 5.4) :

Figure 4.3 : L'interface de notre application.

4.1 Les options d'accueil

Composée de huit fonctionnalités :

- Charger un projet java : 
- Charger une matrice de similarité à partir d'un fichier texte : 
- Remplir une matrice de similarité : 
- Le guide d'utilisation : 
- Calculer le temps d'exécution : 
- Sauvegarder une matrice : 

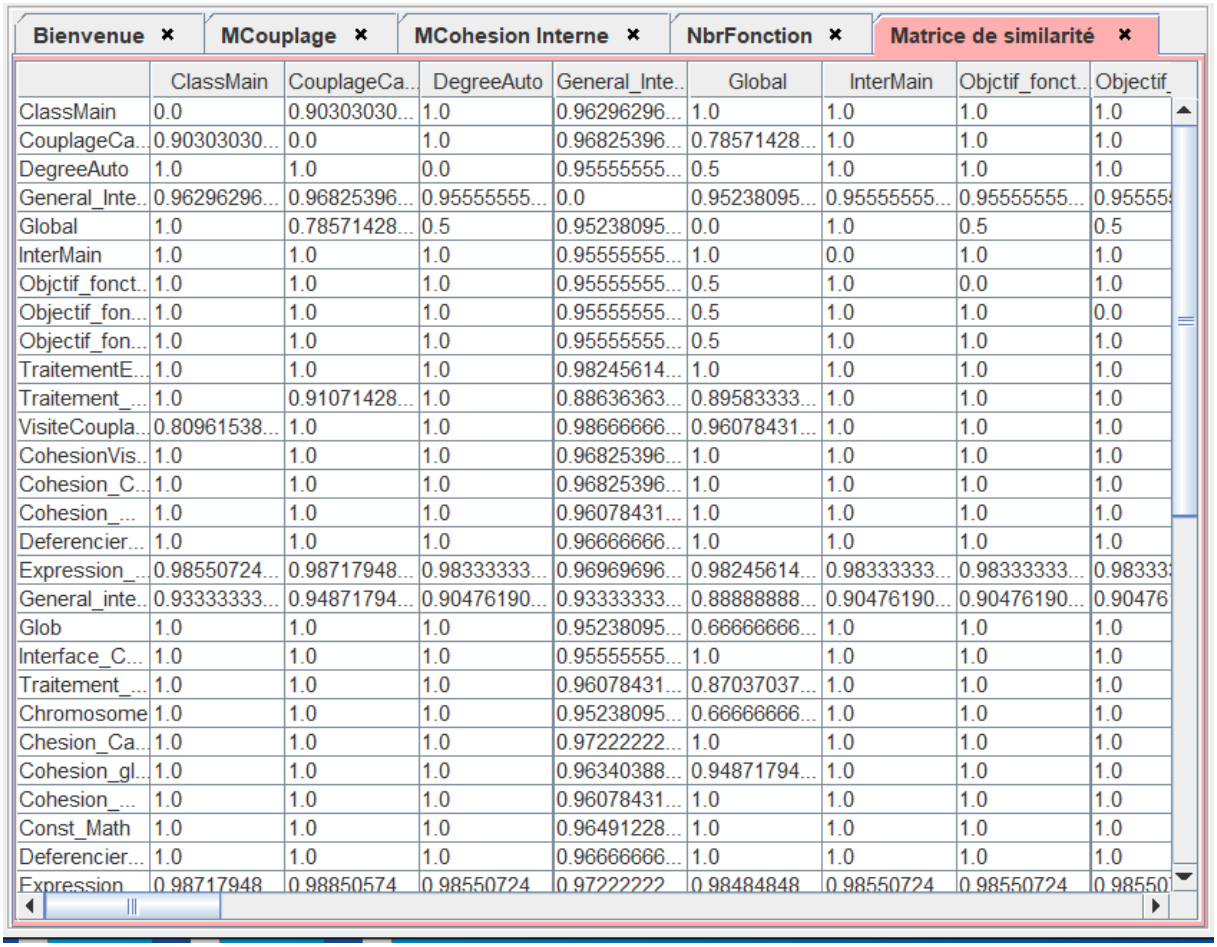
- Restaurer les fenêtres précédentes : 
- Quitter l'application : 

La partie suivante détaille chaque fonctionnalité.

4.1.1 Charger un projet java

Après le chargement du projet, le nombre total des classes est affiché.

Puis quand l'analyseur commencera à filtrer tous les fichiers source et de calculer le couplage, la cohésion, le nombre des fonctionnalités externes ainsi que le calcul de la matrice de similarité (figure 4.5). Puis toutes les informations seront affichées dans des fenêtres séparées ce qui permet à l'utilisateur de les parcourir en parallèle.

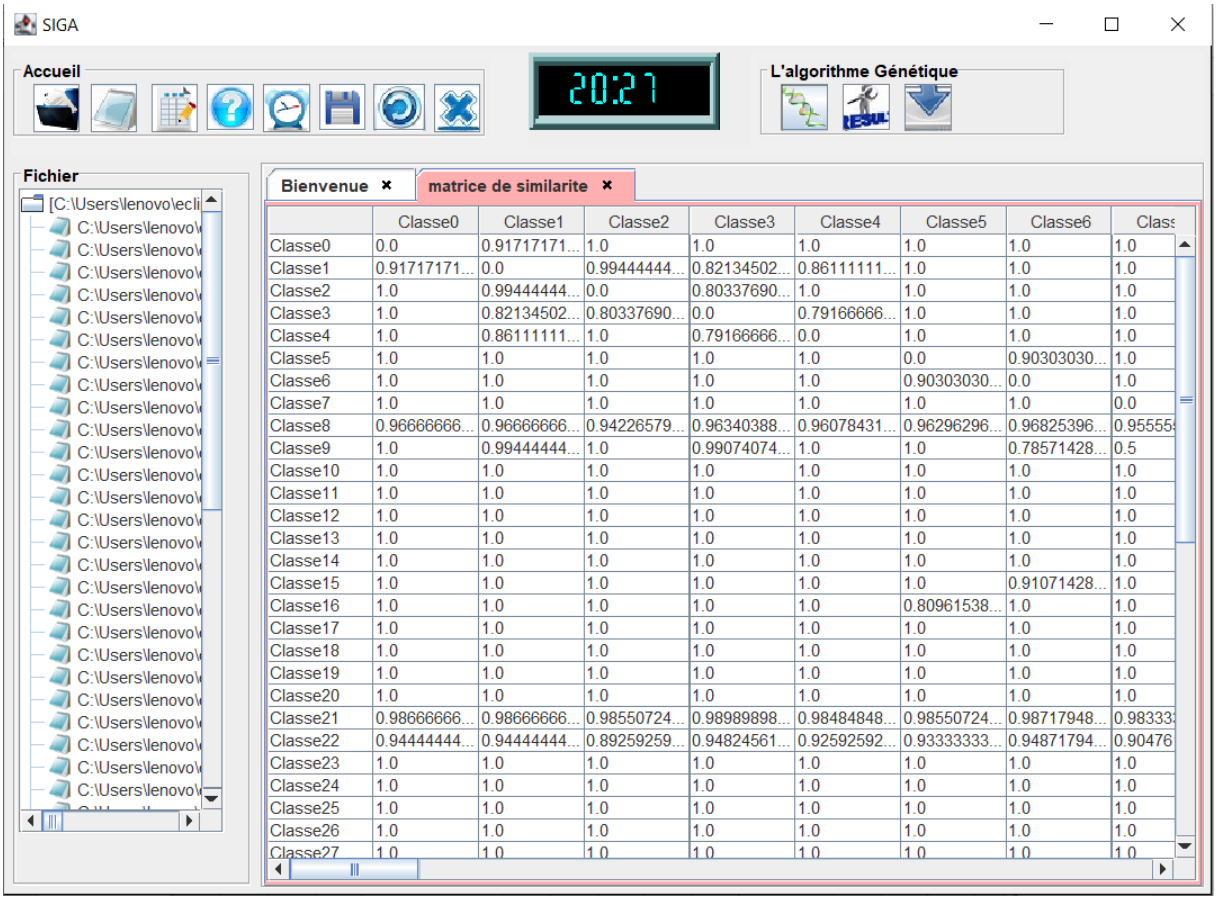


	ClassMain	CouplageCa..	DegreeAuto	General_Inte..	Global	InterMain	Objctif_fonct..	Objectif_
ClassMain	0.0	0.90303030...	1.0	0.96296296...	1.0	1.0	1.0	1.0
CouplageCa...	0.90303030...	0.0	1.0	0.96825396...	0.78571428...	1.0	1.0	1.0
DegreeAuto	1.0	1.0	0.0	0.95555555...	0.5	1.0	1.0	1.0
General_Inte..	0.96296296...	0.96825396...	0.95555555...	0.0	0.95238095...	0.95555555...	0.95555555...	0.95555
Global	1.0	0.78571428...	0.5	0.95238095...	0.0	1.0	0.5	0.5
InterMain	1.0	1.0	1.0	0.95555555...	1.0	0.0	1.0	1.0
Objctif_fonct..	1.0	1.0	1.0	0.95555555...	0.5	1.0	0.0	1.0
Objectif_fon...	1.0	1.0	1.0	0.95555555...	0.5	1.0	1.0	0.0
Objectif_fon...	1.0	1.0	1.0	0.95555555...	0.5	1.0	1.0	1.0
TraitementE...	1.0	1.0	1.0	0.98245614...	1.0	1.0	1.0	1.0
Traitement_...	1.0	0.91071428...	1.0	0.88636363...	0.89583333...	1.0	1.0	1.0
VisiteCoupla...	0.80961538...	1.0	1.0	0.98666666...	0.96078431...	1.0	1.0	1.0
CohesionVis..	1.0	1.0	1.0	0.96825396...	1.0	1.0	1.0	1.0
Cohesion_C...	1.0	1.0	1.0	0.96825396...	1.0	1.0	1.0	1.0
Cohesion_...	1.0	1.0	1.0	0.96078431...	1.0	1.0	1.0	1.0
Deferencier...	1.0	1.0	1.0	0.96666666...	1.0	1.0	1.0	1.0
Expression_...	0.98550724...	0.98717948...	0.98333333...	0.96969696...	0.98245614...	0.98333333...	0.98333333...	0.98333
General_inte..	0.93333333...	0.94871794...	0.90476190...	0.93333333...	0.88888888...	0.90476190...	0.90476190...	0.90476
Glob	1.0	1.0	1.0	0.95238095...	0.66666666...	1.0	1.0	1.0
Interface_C...	1.0	1.0	1.0	0.95555555...	1.0	1.0	1.0	1.0
Traitement_...	1.0	1.0	1.0	0.96078431...	0.87037037...	1.0	1.0	1.0
Chromosome	1.0	1.0	1.0	0.95238095...	0.66666666...	1.0	1.0	1.0
Chesion_Ca...	1.0	1.0	1.0	0.97222222...	1.0	1.0	1.0	1.0
Cohesion_gl...	1.0	1.0	1.0	0.96340388...	0.94871794...	1.0	1.0	1.0
Cohesion_...	1.0	1.0	1.0	0.96078431...	1.0	1.0	1.0	1.0
Const_Math	1.0	1.0	1.0	0.96491228...	1.0	1.0	1.0	1.0
Deferencier...	1.0	1.0	1.0	0.96666666...	1.0	1.0	1.0	1.0
Expression	0.98717948	0.98850574	0.98550724	0.97222222	0.98484848	0.98550724	0.98550724	0.98550

Figure 4.4 : Fenêtre des résultats d'analyse d'un projet java.

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

Le calcul de la matrice de similarité prend beaucoup de temps à cause de calcul des matrices de couplage et de cohésion. C'est pourquoi nous avons ajouté cette fonction qui permet à l'utilisateur d'importer une matrice carrée symétrique de similarité sous la forme texte pour éviter le calcul de matrice de couplage et de cohésion sur la même application à analyser et gagner ainsi un peu de temps (figure 5.6).



	Classe0	Classe1	Classe2	Classe3	Classe4	Classe5	Classe6	Class
Classe0	0.0	0.91717171...	1.0	1.0	1.0	1.0	1.0	1.0
Classe1	0.91717171...	0.0	0.99444444...	0.82134502...	0.86111111...	1.0	1.0	1.0
Classe2	1.0	0.99444444...	0.0	0.80337690...	1.0	1.0	1.0	1.0
Classe3	1.0	0.82134502...	0.80337690...	0.0	0.79166666...	1.0	1.0	1.0
Classe4	1.0	0.86111111...	1.0	0.79166666...	0.0	1.0	1.0	1.0
Classe5	1.0	1.0	1.0	1.0	1.0	0.0	0.90303030...	1.0
Classe6	1.0	1.0	1.0	1.0	1.0	0.90303030...	0.0	1.0
Classe7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
Classe8	0.96666666...	0.96666666...	0.94226579...	0.96340388...	0.96078431...	0.96296296...	0.96825396...	0.95555
Classe9	1.0	0.99444444...	1.0	0.99074074...	1.0	1.0	0.78571428...	0.5
Classe10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe12	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe13	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe14	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe15	1.0	1.0	1.0	1.0	1.0	1.0	0.91071428...	1.0
Classe16	1.0	1.0	1.0	1.0	1.0	0.80961538...	1.0	1.0
Classe17	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe18	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe19	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe20	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe21	0.98666666...	0.98666666...	0.98550724...	0.98989898...	0.98484848...	0.98550724...	0.98717948...	0.98333
Classe22	0.94444444...	0.94444444...	0.89259259...	0.94824561...	0.92592592...	0.93333333...	0.94871794...	0.90476
Classe23	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe24	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe25	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe26	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Classe27	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Figure 4.5 : Importation d'une matrice de similarité à partir d'un fichier texte.

4.1.3 Remplir une matrice de similarité

Nous avons ajouté cette option dans le cas où l'utilisateur voudrais explorer des exemples théoriques qu'il veut tester et qui n'a pas pu les avoir à travers des cas réels. Cette option donne à l'utilisateur la possibilité de fixer les dimensions de la matrice, puis de remplir les valeur (figure 4.7). La matrice remplie doit être une matrice symétrique.

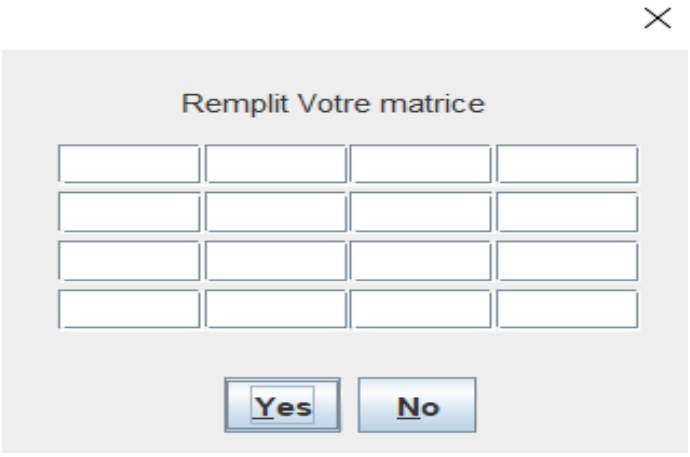


Figure 4.6 : Matrice carrée de taille 4*4.

4.1.4 Le guide d'utilisation

Le guide,affiche une petite description de notre projet, ainsi qu'une aide à l'utilisateur pour comprendre comment fonctionne notre application (figure 4.8).

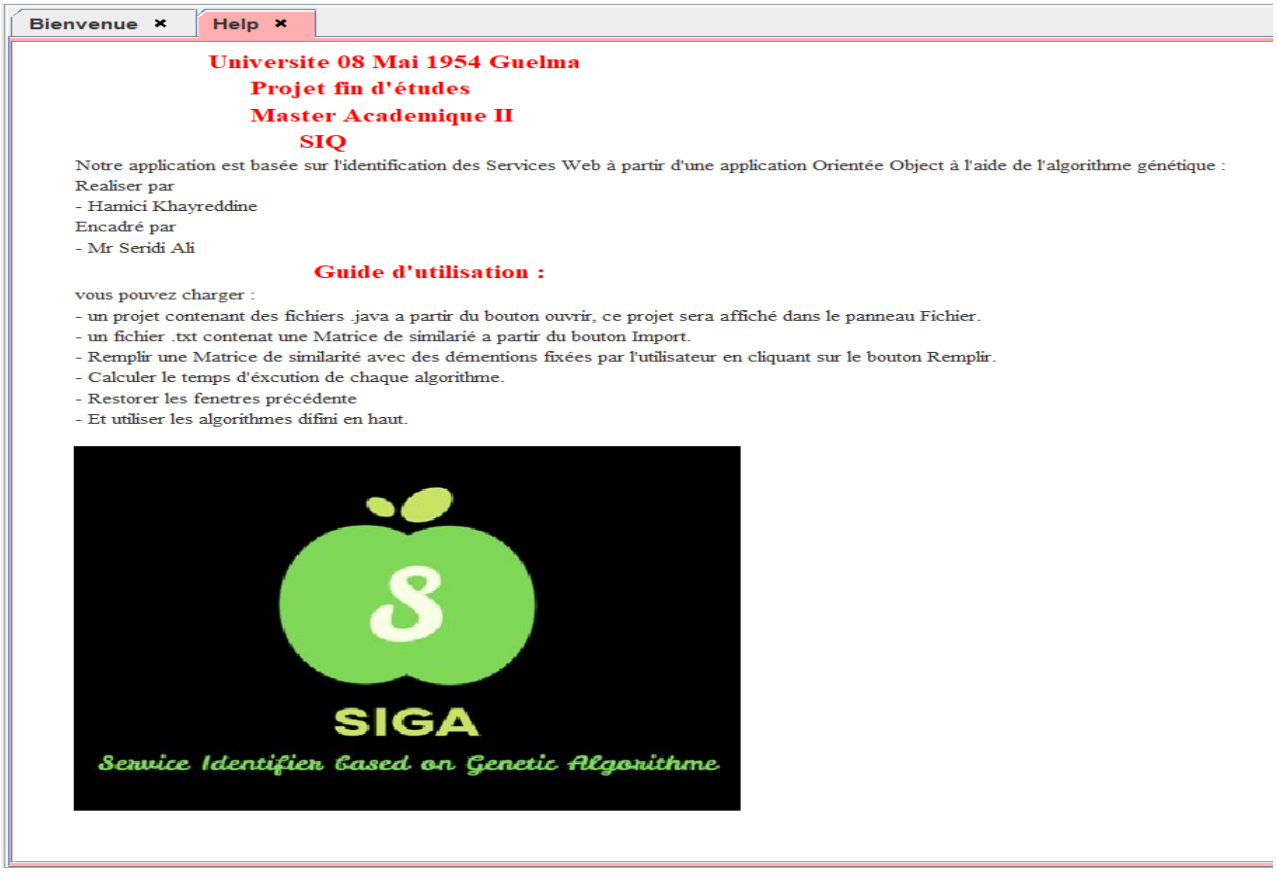


Figure 4.7 : Le guide d'utilisation.

4.1.5 Le calcul du temps d'exécution

Après l'exécution de l'algorithme génétique nous pouvons cliquer sur cette option pour voir le temps écoulé pendant l'exécution en nanoseconde (figure 4.9), afin de pouvoir comparer la vitesse de convergence des différents cas.

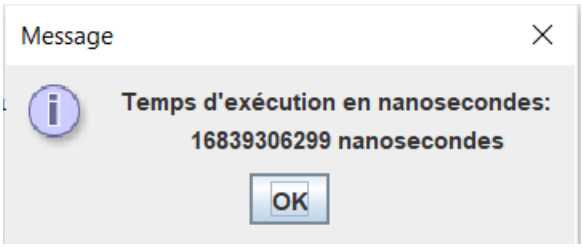


Figure 4.8 : Le temps d'exécution.

4.1.6 Sauvegarder une matrice

Cette option permet de sauvegarder la matrice de similarité dans le cas où nous voulons utiliser la même application et lui appliquer l'AG avec d'autre paramètres. La sauvegarde se fait sous forme de texte, lorsque la matrice est sauvegardée une boite de dialogue sera affichée (figure 5.10).

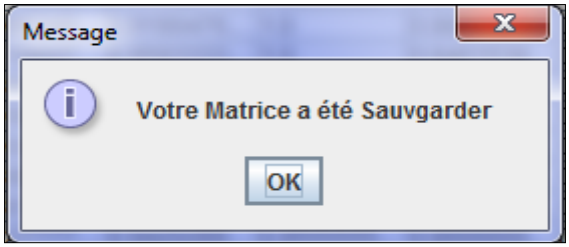


Figure 4.9 : Boite de confirmation de sauvegarde d'une matrice.

4.2 L'application de l'algorithme Génétique

Après la génération de la matrice de similarité nous pouvons aller sur la partie de l'exécution de l'algorithme génétique. Cette partie est composée de :

- L'exécution de l'algorithme génétique pour l'identification des services : 
- Résultats d'exécution d'un projet plusieurs fois avec différents paramètres : 
- Exporter les résultats des exécutions : 

4.2.1 L'exécution de l'algorithme génétique

La première étape consiste à fixer les valeurs des paramètres de l'AG (figure 4.11).

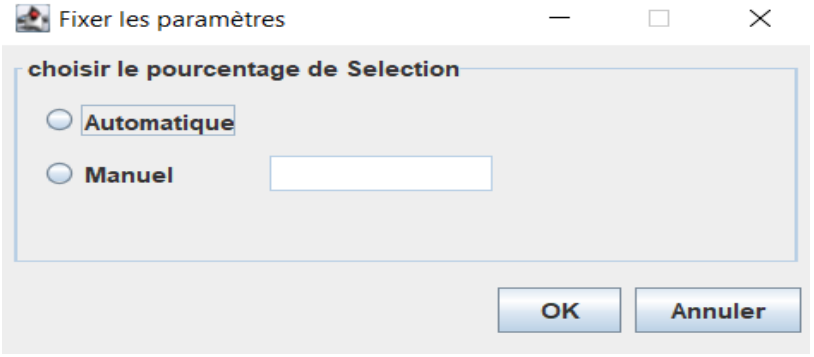


Figure 4.10 : Boite du choix du pourcentage de Sélection.

Si l'utilisateur choisi le pourcentage automatique du paramètre de sélection, l'algorithme utilise par défaut un pourcentage de 50% de la population.

Après avoir fixé le pourcentage de sélection, d'autres paramètres doivent être fixés tel que le nombre de génération (critère d'arrêt) et le nombre des itérations pour lancer la mutation (figure 4.12).

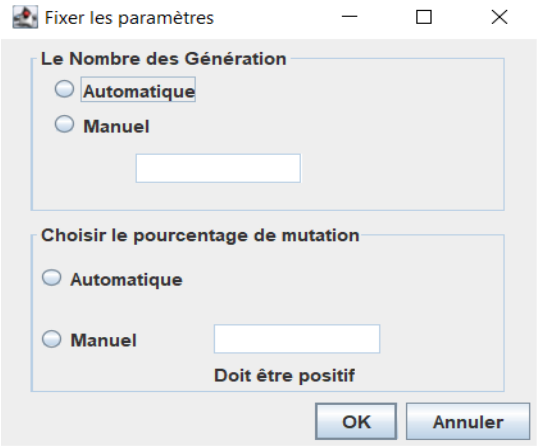


Figure 4.11 : Boite de choix du nombre de génération et nombre d'itération pour chaque mutation.

Le choix du nombre de génération automatique, fixe la valeur à (10*Nombre de classes) génération.

Le choix du nombre d'itération pour chaque mutation automatique est fixé à la valeur 1% du nombre d'itérations.

Après avoir fixé tous les paramètres de l'algorithme génétique, ce dernier vas générer une population initiale de taille égale au nombre total des classes et affecter à chaque classe son identificateur. (Figure 4.13).

```
[1][2][3][4][5]
[1 2][3 4][5]
[1 2 3][4 5]
[1 2 3 4][5]
[1 2 3 4 5]
```

Figure 4.12 : Exemple de population initiale pour 5 classes

Après l'exécution complète de l'algorithme, une boite de dialogue apparait et affiche le nombre total des services comme résultat (figure 4.14). Puis un fichier texte va s'ouvrir automatiquement contenant le 11meilleur chromosome comme résultat de l'exécution de l'algorithme (figure 4.15)

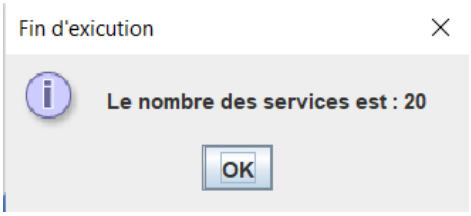
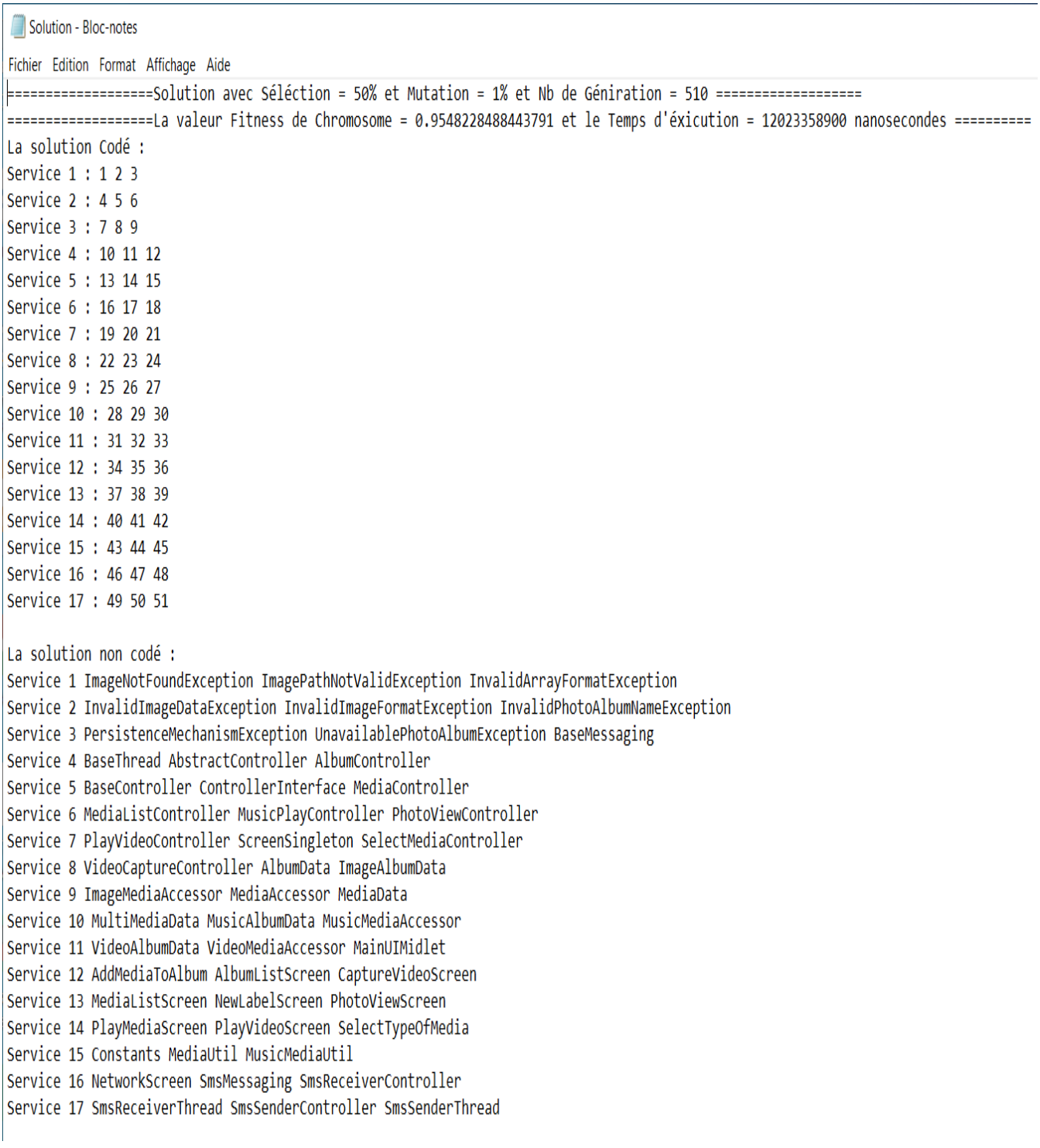


Figure 4.13 : Le nombre des services


Solution - Bloc-notes

Fichier Edition Format Affichage Aide

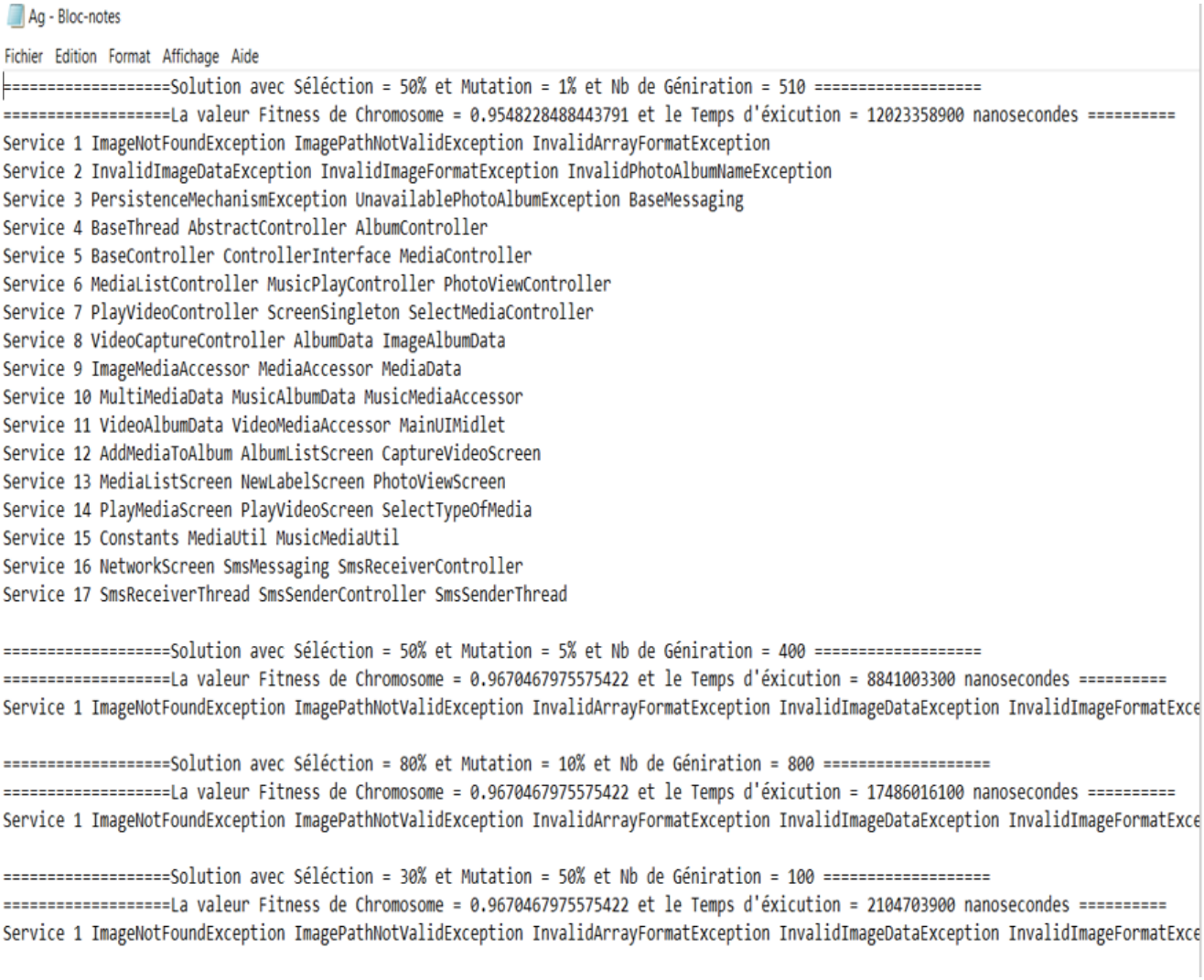
==========Solution avec Séléction = 50% et Mutation = 1% et Nb de Génération = 510 ==========
==========La valeur Fitness de Chromosome = 0.9548228488443791 et le Temps d'éxicution = 12023358900 nanosecondes ==========
La solution Codé :
Service 1 : 1 2 3
Service 2 : 4 5 6
Service 3 : 7 8 9
Service 4 : 10 11 12
Service 5 : 13 14 15
Service 6 : 16 17 18
Service 7 : 19 20 21
Service 8 : 22 23 24
Service 9 : 25 26 27
Service 10 : 28 29 30
Service 11 : 31 32 33
Service 12 : 34 35 36
Service 13 : 37 38 39
Service 14 : 40 41 42
Service 15 : 43 44 45
Service 16 : 46 47 48
Service 17 : 49 50 51

La solution non codé :
Service 1 ImageNotFoundException ImagePathNotValidException InvalidArrayFormatException
Service 2 InvalidImageDataException InvalidImageFormatException InvalidPhotoAlbumNameException
Service 3 PersistenceMechanismException UnavailablePhotoAlbumException BaseMessaging
Service 4 BaseThread AbstractController AlbumController
Service 5 BaseController ControllerInterface MediaController
Service 6 MediaListController MusicPlayController PhotoViewController
Service 7 PlayVideoController ScreenSingleton SelectMediaController
Service 8 VideoCaptureController AlbumData ImageAlbumData
Service 9 ImageMediaAccessor MediaAccessor MediaData
Service 10 MultiMediaData MusicAlbumData MusicMediaAccessor
Service 11 VideoAlbumData VideoMediaAccessor MainUIMidlet
Service 12 AddMediaToAlbum AlbumListScreen CaptureVideoScreen
Service 13 MediaListScreen NewLabelScreen PhotoViewScreen
Service 14 PlayMediaScreen PlayVideoScreen SelectTypeOfMedia
Service 15 Constants MediaUtil MusicMediaUtil
Service 16 NetworkScreen SmsMessaging SmsReceiverController
Service 17 SmsReceiverThread SmsSenderController SmsSenderThread

Figure 4.14 : L'affichage du résultat

4.2.2 Affichage des résultats

Après l'exécution de l'algorithme génétique plusieurs fois sur le même projet java, et avec différents paramètres, si l'utilisateur veut voir les résultats précédents il peut cliquer sur l'option Afficher les résultats. Un fichier texte apparait et contient tous les résultats déjà calculés. (Figure 4.15).


Ag - Bloc-notes

Fichier Edition Format Affichage Aide

==========Solution avec Séléction = 50% et Mutation = 1% et Nb de Génération = 510 ==========
==========La valeur Fitness de Chromosome = 0.9548228488443791 et le Temps d'éxicution = 12023358900 nanosecondes ==========
Service 1 ImageNotFoundException ImagePathNotValidException InvalidArrayFormatException
Service 2 InvalidImageDataException InvalidImageFormatException InvalidPhotoAlbumNameException
Service 3 PersistenceMechanismException UnavailablePhotoAlbumException BaseMessaging
Service 4 BaseThread AbstractController AlbumController
Service 5 BaseController ControllerInterface MediaController
Service 6 MediaListController MusicPlayController PhotoViewController
Service 7 PlayVideoController ScreenSingleton SelectMediaController
Service 8 VideoCaptureController AlbumData ImageAlbumData
Service 9 ImageMediaAccessor MediaAccessor MediaData
Service 10 MultiMediaData MusicAlbumData MusicMediaAccessor
Service 11 VideoAlbumData VideoMediaAccessor MainUIMidlet
Service 12 AddMediaToAlbum AlbumListScreen CaptureVideoScreen
Service 13 MediaListScreen NewLabelScreen PhotoViewScreen
Service 14 PlayMediaScreen PlayVideoScreen SelectTypeOfMedia
Service 15 Constants MediaUtil MusicMediaUtil
Service 16 NetworkScreen SmsMessaging SmsReceiverController
Service 17 SmsReceiverThread SmsSenderController SmsSenderThread

==========Solution avec Séléction = 50% et Mutation = 5% et Nb de Génération = 400 ==========
==========La valeur Fitness de Chromosome = 0.9670467975575422 et le Temps d'éxicution = 8841003300 nanosecondes ==========
Service 1 ImageNotFoundException ImagePathNotValidException InvalidArrayFormatException InvalidImageDataException InvalidImageFormatExce

==========Solution avec Séléction = 80% et Mutation = 10% et Nb de Génération = 800 ==========
==========La valeur Fitness de Chromosome = 0.9670467975575422 et le Temps d'éxicution = 17486016100 nanosecondes ==========
Service 1 ImageNotFoundException ImagePathNotValidException InvalidArrayFormatException InvalidImageDataException InvalidImageFormatExce

==========Solution avec Séléction = 30% et Mutation = 50% et Nb de Génération = 100 ==========
==========La valeur Fitness de Chromosome = 0.9670467975575422 et le Temps d'éxicution = 2104703900 nanosecondes ==========
Service 1 ImageNotFoundException ImagePathNotValidException InvalidArrayFormatException InvalidImageDataException InvalidImageFormatExce

Figure 4.15 : L'affichage des résultats sauvegardés.

4.2.3 Exporter les solutions

Cette fonctionnalité permet aux utilisateurs d'exporter leurs résultats après l'exécution de l'algorithme sous forme de fichier texte afin de garder les résultats pour une exploitation ultérieur, ce qui permet de les comparer et de détecter les paramètres qui donnent les meilleurs résultats. (Figure 4.17).

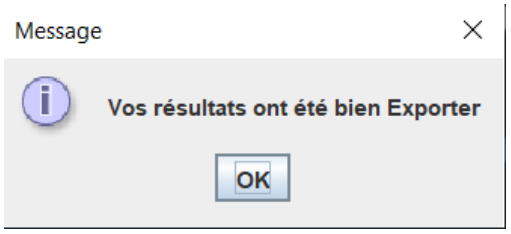


Figure 4.16 : L'exportation des résultats.

5. Expérimentation

Pour valider nos résultats nous avons suivi les étapes suivantes :

- Chercher dans les travaux similaires qui ont essayé de faire migrer des applications orientées objet vers la SOA. Nous avons prix 2 exemples de tailles différentes afin de voir le comportement des algorithmes dans les différents cas. Puis nous avons comparé nos résultats avec les résultats des travaux similaires
- Une deuxième chose à faire pour améliorer le paramétrage de l'AG, nous avons testé l'algorithme avec différentes combinaisons de paramètre afin de trouver le meilleur paramétrage.

5.1 Les applications testées

Nous avons choisi deux applications orientées objet comme exemple de test que nous avons récupéré d'un travail similaire dans [39].

- **Java Calculator Suite :** est une calculatrice à code source ouvert implémentée en Java. Elle effectue des opérations mathématiques de base, possède une interface graphique et prend en charge les booléens, les grands nombres, les nombres machine et environ 25 opérations différentes. Cette application comprend 17 classes.
- **MobileMedia :** est une application Java à code source ouvert utilisée pour gérer les médias (photo, musique et vidéo) sur les appareils mobiles. Cette application contienne 51 classes.

Nous avons récupéré les résultats des applications depuis [39].

L'Application	Les résultats attendus	
	Le numéro de service	Les classes qui composent
Java Calculator Suite	Service 1	CalcMachineNumber
	Service 2	OperatorControlCenter
	Service 3	Calculator CalculatorException CalculatorTester Jcalc Jcalc_applet
	Service 4	E Jjcalc_math Jcalc_trig Variable_interface
	Service 5	VariableTable OperatorChecker
	Service 6	PI
	Service 7	Entries GuiCommandLine ResultsList

Tableau 4.2 : Les résultats attendus de l'application JavaCalculatorSuite.

Pour l'application MobileMedia ils ont indiqué le nombre des services résultat seulement qui est 13.

5.2 Le choix des paramètres de l'AG

Le tableau ci dessue montre les résultats des tests avec différent paramétrage de l'AG appliqués aux applications citées.

L'application	Avec un			
	Sélection	Nombre de génération	Mutation	Nbservice
Java Calculator Suite	Automatique (50%)	Automatique (10*nb de classes)	Automatique (1% du nombre d'itérations)	9
	30%	300	5%	8
	70%	1000	10%	7
	80%	1100	15%	3
	100%	2000	50%	2
MobileMedia	Automatique (50%)	Automatique (10*nb de classes)	Automatique (1% du nombre d'itérations)	19
	30%	300	5%	17
	70%	1000	10%	11
	80%	1100	15%	17
	100%	2000	50%	17

Tableau 4.3 : Les résultats d'application de l'AG pour chaque application avec différent paramètres.

Voici ci-dessous une comparaison entre les résultats de l'application Java Calculator Suite par l'AG avec un paramétrage (70%, 1000, 10%) et une valeur de fitness = 0.53 avec le résultat de [39].

L'Application	Les résultats	
	Résultat SIGA	Attendu resultat [39]
Java Calculator Suite	Entries	CalcMachineNumber
	OperatorControlCenter	OperatorControlCenter
	CalcMachineNumber Calculator CalculatorException CalculatorTester jcalc_trig VariableTable GuiCommandLine jcalc jcalc_applet jcalc_math operatorChecker	Calculator CalculatorException CalculatorTester Jcalc Jcalc_applet
	E	E Jjcalc_math Jcalc_trig Variable_interface
	variable_interface	VariableTable OperatorChecker
	PI	PI
	ResultsList	Entries GuiCommandLine ResultsList

Tableau 4.4: Comparaison des résultats de l'AG pour l'application Java Calculator Suite (Avec paramétrage 70%, 1000, 10%).

Remarque :

Nous pouvons voir que les résultats avec les paramètres choisies sont assez proches des résultats du [39], mais nous ne pouvons dire quelles sont les meilleurs résultats puisque nous n'avons pas la version de l'application JavaCalculatorSuite sous la forme de service, en plus la fonction fitness n'est pas la même

Afin de mieux valider notre travail, nous avons décidé de le comparer avec le travail de [21] puisque la fonction fitness est très proche malgré que les algorithmes utilisés soient différent (algorithme hiérarchique, soustractive et isodata).

Voici ci-dessous une autre comparaison entre notre AG SIGA et [21] appliqués sur l'application Java Calculator Suite (17) et une autre application qui s'appelle calcul_entier (12 classes)

- **Calcul_*entier :*** est une application simple qui contient 13 classes java, elle est composée de trois modules spécialisés dans le calcul sur deux entier (la somme, le produit et la soustraction).

L'app	Paramétrage			Valeur de Fitness	Service	Temps d'ex en nano
Calcul_entier	Sélection	Génération	Mutation			
	Automatique	Automatique	Automatique	0.39	4	202080600
	55%	500	Automatique	0.33	6	786839700
	60%	1000	Automatique	0.24	4	1552676700
	60%	500	2%	0.36	5	752866910
	70%	500	2%	0.31	3	752856900
	Automatique	500	2%	0.48	2	777680300
	80%	1000	Automatique	0.27	7	1508337300
	80%	2000	Automatique	0.09	9	3000585420
	80%	5000	3%	0.48	2	7198099200
	90%	Automatique	3%	0.16	6	201960400
	90%	Automatique	10%	0.23	4	198115000
	95%	Automatique	7%	0.48	2	197816700
	70%	Automatique	5%	0.65	3	193685500
	20%	1000	10%	043	4	1463837800
	10%	1000	Automatique	0.44	4	1181133100
	10%	2000	Automatique	0.48	4	3380665100
	10%	3000	Automatique	0.62	3	3478899100
	15%	3000	Automatique	0.41	9	3375670200
	15%	3000	Automatique	0.41	7	3404680300
	15%	2500	Automatique	0.58	5	2831672100
	20%	2500	Automatique	0.66	4	2720621100
	20%	1500	Automatique	0.48	4	1785311062
	20%	2000	Automatique	0.74	7	2303790000

Tableau 4.5 : exécution de Calcul_entier par SIGA avec différent paramètres

L'application	Avec un			Nb des services					Temp d'exécution			
	Sélection	Nombre de génération	Mutation	Avec l'AG		Par rapport [21]			De l'AG	Le travail de [21]		
				Service	F	L'algorithme hiérarchique.	L'algorithme Soustractive Clustering.	L'algorithme IsoData.		L'algorithme hiérarchique.	L'algorithme Soustractive Clustering.	L'algorithme IsoData.
Java Calculator Suite	Automatique (50%)	Automatique (10*nb)	Automatique (1%)	9	0.1	5	6	7	485879600	2940223	172669	3329636
	30%	300	5%	8	0.34				791086100			
	70%	1000	10%	7	0.42				2747441701			
	80%	1100	15%	3	0.33				2797295500			
	100%	2000	50%	5	0.73				5060996400			
Calcul_entier	Sélection	Nombre de génération	Mutation			L'algorithme hiérarchique.	L'algorithme Soustractive Clustering.	L'algorithme IsoData.		L'algorithme hiérarchique.	L'algorithme Soustractive Clustering.	L'algorithme IsoData.
	Automatique (50%)	Automatique (10*nb)	Automatique (1%)	4	0.21	2	3	3	174545100	1380749	31999	1642168
	30%	300	5%	3	0.47				380220099			
	70%	1000	10%	3	0.18				129133899			
	80%	1100	15%	4	0.93				135074000			
	100%	2000	50%	8	0.11				2483062600			

Tableau 4.6 : comparaison avec le travail de [21]

Nous avons remarqué que notre algorithme est moins rapide que les algorithmes de [21], cela est dû au calcul intensif de l'AG, mais il est plus efficace que l'algorithme hiérarchique et Soustractive Clustering. Par contre il est proche de l'algorithme isoData par rapport les résultats.

<table>
<tr><td rowspan="2">L'Application</td><td colspan="2">Les résultats</td></tr>
<tr><td>Résultat SIGA</td><td>Attendu résultat [21]</td></tr>
<tr><td rowspan="4">Calcul_entier</td><td>add
add_affiche
add_fonct</td><td>Add
Add_affiche
Add_fonct
Add_param</td></tr>
<tr><td>add_param
mul
mul_affiche</td><td>Mul
Mul_affiche
Mul_fnct
Mul_param
Mul</td></tr>
<tr><td>mul_fnct
mul_param
sous</td><td rowspan="2">Sous
Sous_affiche
Sous_fnct
Sous_param</td></tr>
<tr><td>sous_affiche
sous_fnct
sous_param</td></tr>
</table>

Tableau 4.7 : Comparaison des résultats de l'AG pour l'application Calcul_entier (Avec paramétrage 80%, 1100, 15%).

6. Conclusion

Dans ce chapitre nous avons présenté notre application d'identification des services avec l'AG, nous avons expliqué chaque partie de notre interface.

Ensuit nous avons testé notre algorithme avec deux applications et comparé les résultats de l'application récupéré depuis deux travaux différents qui sont assez proche avec les résultats de notre AG.

En fin nous pouvons déduire que les résultats de l'algorithme génétique ne sont pas stable, par exemple si en exécuter l'AG sur une application plusieurs fois avec les mêmes paramètres nous allons obtenir des résultats différents mais proche.

Conclusion générale et perspectives

Ces dernières années, l'architecture orientée services a connu une attention croissante dans le domaine de la réingénierie logicielle, vu ses avantages et sa facilité de maintenance, d'agilité et de flexibilité.

A travers ce projet, Nous avons exploré la possibilité d'utiliser les algorithmes génétiques pour la réingénierie des application OO vers une SOA et plus précisément l'automatisation de la phase d'identification des services. Le plus grand défi que nous avons rencontré est comment modéliser la population des solutions possible puis la définition d'une fonction fitness représentatif.

Afin d'explorer le maximum de possibilités, nous avons proposé un AG paramétrable, ce qui nous a permis de tester différentes valeurs en cherchant la combinaison des paramètres qui nous donnes la meilleure solution.

Comme on le sait l'AG donne une solution approximative, dans notre cas les solutions varient de façon aléatoirement, ce qui nous n'a pas permis de trouver une relation claire entre les paramètres et résultats.

Nos déductions ont été faites suite à une expérimentation conduite sur deux exemples trouvés dans des études similaires antérieures.

Comme perspective à ce travail, nous proposons de faire une nouvelle description des caractéristiques de l'algorithme génétique, spécialement une redéfinition de la fonction Fitness ou bien l'utilisation d'une autre méthode de codage, sélection etc… puis de faire une étude comparative entre nos résultats et ceux de la nouvelle description afin de conclure laquelle des descriptions et la plus optimale.

Liste des webographies

[W1] [https://www.commentcamarche.net/contents/1241-soa-architecture-orientee-service, 20/03/2021].

[W2] [http://www.journaldunet.com/developpeur/tutoriel/theo/051013-explication-soa.shtml, 20/03/2021].

[W3] [https://www.zdnet.fr/actualites/soa-comprendre-l-approche-orientee-service-39206712.htm, 28/03/2021].

Liste des bibliographiques

[1] Depoortere Franck, Méthodes de réingénierie des logiciels. Mémoire de Maitrise d'informatique Option transversale, Septembre 2002.

[2] Boby laforet, Rétro-ingénierie d'un logiciel existant du domaine de la santé en préparation pour sa modernisation, rapport de projet, Montréal, le 15 juillet 2015.

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

[4] Christophe heubès "Mise en œuvre d'une SOA : Les clés du succès" 2007.

[5] Nicolas Durand , Algorithmes Génétiques et autres méthodes d'optimisation appliqués à la gestion de trafic aérien ,2004 ,Thèse , l'institut national polytechnique de Toulouse.

[6] Souquet Amédée. Algorithmes génétiques. Mémoire de master, Université de Nice, 2004.

[7] Holland, J. H. "Adaptation in Natural and Artificial Systems". University of Michigan Press, Ann Arbor, 1975.

[8] Eric Taillard Partick Siarry Johann Dréo, Alain Petrowski. Métaheuristiques pour l'oprimisation difficile. Eyrolles, Juillet 2003.

[9] Jean-Marc Alliot, Nicolas Durand Algorithmes génétiques. Technical report.

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

[11] Tarek Chaari, Un algorithme génétique pour l'ordonnancement robuste: application au problème du flow shop hybride, Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis.

[12] Sophie Voisin. Application des algorithmes génétiques à l'estimation de mouvement par modélisation markovienne. Technical report, Centre National de la Recharche Scientifique (CNRS).

[13] M.C.CowgillR, J.Harvey, L.T.Watson. A genetic algorithm approach to cluster analysis. Volume 37, Issue 7, April 1999

[14] S. Izza, L. Vincent, P. Burlat, H. Solignac et P. Lebrun, Intégration d'applications : *Etat de l'art et perspectives*, JIE 2'04, les 2 emes Journées d'Informatique pour l'Entreprise, université Saad Dahleb, Blida (Algérie) 2004.

[15] P. Sarang, R. Loganathan, F. Jennings, M. Juric, SOA Approach to Integration: XML, Web services, ESB, and BPEL in real-world SOA projects. Packt Publishing 2007. ISBN : 13 978-1-904811-17-6.

[16] Syntec Informatique, Les Architectures Orientées services : le moteur de l'innovation, livre blanc de Si (Syntec informatique), juin 2007.

[17] Rakesh Agarwal, Ajit Sarangui et Swati Das,« *Reengineering of database intensive application* »ACM SIGSOFT Software Engineering Notes,3 mai 2003.

[18] E.J. Chikofsky et J.H. Cross II, « Reverse engineering and design recovery: a taxonomy », Software, IEEE, vol.7, no.1,, janvier 1990.

[19] P. Bourque et R. Dupuis, « Guide to the Software Engineering Body of Knowledge 2004 Version SWEBOK », IEEE Computer Society Professional Practices Committee, 14 janvier 2004.

[20] Adeel Ahmad, Contribution à la Multi-modélisation des Applications Distribuées pour le Contrôle de l'Évolution des Logiciels Décembre 2011

[21] Chelaghmia rima,Nebili wafa. Mémoire de fin d'étude Présenté pour l'obtention du diplôme de master, 2015.

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

[23] W. Li, S.Henry, « Object oriented metrics that predict maintainability », Journal of Systems and Software, 1993.

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

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

[26] Charles Fleurent « Algorithmes génétiques ». Volume 14.2 - été-automne 2019.

[27] John H HOLLAND. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, *1975*.

[28] Guillaume « Algorithme génétique : Darwin au service de l'intelligence artificielle - Toile de Fond ».

[29] Kaur M, Kumar V. Beta chaotic map based image encryption using genetic algorithm. Int J Bifurcation Chaos.

[30] Kavitha AR, Chellamuthu C. Brain tumour segmentation from MRI image using genetic algorithm with fuzzy initialisation and seeded modified region growing (GFSMRG) method. The Imaging Science Journal.

[31] Shabankareh SG, Shabankareh SG. Improvement of edge-tracking methods using genetic algorithm and neural network, 2019 5th Iranian conference on signal processing and intelligent systems (ICSPIS) Iran: Shahrood; 2019.

[32] Lee Y, Hara T, Fujita H, Itoh S, Ishigaki T. Automated detection of pulmonary nodules in helical CT images based on an improved template-matching technique. in IEEE Transactions on Medical Imaging.

[33] Thomas Erl. SOA design patterns. Pearson Education, 2008.

[34] Clara Pizzuti, Nicola Procopio, "A K-means Based Genetic Algorithm for Data Clustering", National Research Council of Italy (CNR).

[35] Yong, Y. and G. Xin_cheng. A new minority kind of sample sampling method based on genetic algorithm and K-means cluster. in Computer Science & Education (ICCSE), 2012 7th International Conference on. 2012. IEEE

[36] Brian S Mitchell and Spiros Mancoridis. On the automatic modularization of software systems using the bunch tool. IEEE Transactions on Software Engineering, vol. 32, no. 3, 2006.

[37] Hemant Jain, Huimin Zhao, and Nageswara R Chinta. A spanning tree based approach to identifying web services. *International Journal of Web Services Research*, 2004.

[38] Mostefai Abdelkader, Mimoun Malki, and Sidi Mohamed Benslimane. A heuristic approach to locate candidate web service in legacy software. International Journal of omputer applications in Technology.

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

[40] Lyes Belhoul, "Résolution de problèmes d'optimisation combinatoire mono et multi-objectifs par énumération ordonnée", thèse 2014.