RÉPUBLIQUE ALGERIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ « YAHIA FARÈS » DE MÉDÈA

FACULTE DES SCIENCES

Département Mathématiques et Informatique

Mémoire de fin d'étude

Présenté pour l'obtention du diplôme

de Master en Informatique

Option : Ingénierie des Systèmes et Technologie du Web

Thème :

DÉCOUVERTE MULTICRITÈRES BASEE SERVICE DES COMMUNAUTES DANS LES RÉSEAUX SOCIAUX COMPLEXES

Proposé et dirigé par

M^r BOUDJEBBOUR Karim

Préparé et présenté

M^r Boudjema Ali

M^r BENAISSA Azzeddine

Devant le jury composé de

M^r. RAHIM Messaoud - Président

M^r. RAFAA Tahar - Examinateur

Promotion 2017

DEDICACES

Je tiens à dédier ce modeste travail à la lumière de ma vie : mes chers parents, pour leur plus grand amour, soutien, encouragement de la patience et de l'aide continue pendant mes années d'études.

Ce travail est également dédié pour toute la famille

Et à mes chères amis.

Je désire dédier ce travail aussi à tous mes amis dans le campus, et à tous ceux qui ont contribué à la réalisation de ce travail.

A mon Binôme Mahdi

Je dédie ce travail.

A tous ceux qui ont participé de près ou de loin à la réalisation de ce Projet je dédie le fruit de ce mémoire.

ALI

Je dédie ce travail à ceux qui me portent
dans leurs cœurs et m'accompagnent dans leurs pensées.
A ceux dont personne ne peut remplacer les sacrifices
qu'ils ont consentis pour mon éducation et mon bien
être Mes très chers parents
A mes très chers frères : Kamel et Sidahmed et sa
femme Assia et leur deux bébés et à ma très
chère sœur Karima, et toute ma famille sans exception
A mon très cher partenaire Sido qui a fait
un excellent travail et a donné une très bonne
ambiance et aux membres de notre promotion
A tous mes amis et toute personne qui de près ou de
loin nous a aidé et a participé à réaliser notre projet
et qui nous ont encouragé tout au long de notre
formation

Azzedine



Remerciement

MERCI !

Au terme de ce travail, nous tenons à remercier en premier lieu « Dieu » qui nous a donné la force pour achever ce travail. Nous remercions vivement notre promoteur

Mr .BOUDJEBBOUR .K

Et tous les enseignants de l'université de Médéa

Nous remercions du fond du cœur nos chers parents de tout ce qu'ils ont fait pour nous.

Nous tenons également à remercier tous ceux qui nous ont aidés à réaliser ce travail de près ou de loin.

Nos remerciements s'adressent également à tous les membres de jury, qui font l'honneur du juger notre travail

Table de Matières

Résumé ….....**i**

Liste des figures….....**iii**

Liste des algorithmes …… **vi**

Liste des tableaux ……..**vi**

Liste des acronymes……..**vii**

Introduction générale …… **1**

CHAPITRE I : Etat de l'art

1. INTRODUCTION **3**

2. SERVICES WEB **3**

2.1. DEFINITION **3**

2.2. HISTORIQUE ET ORIGINE DES CONCEPTS **4**

2.3. INTERETS DES SERVICES WEB (OBJECTIFS) **5**

3. CARACTERISTIQUES DES SERVICES WEB **6**

3.1. ASPECTS DES SERVICES WEB **6**

3.1.1. USAGE DU XML **6**

3.1.2. COUPLAGE LACHE **6**

3.1.3. GROS GRAINS **6**

3.1.4. CAPACITE A ETRE SYNCHRONE OU ASYNCHRONE **6**

3.1.5. PRISE EN CHARGE DES APPELS DE PROCEDURE A DISTANCE (RPC) **7**

3.1.6. PREND EN CHARGE L'ECHANGE DE DOCUMENTS **7**

3.2. ARCHITECTURE ET FONCTIONNEMENT **7**

3.3. PROTOCOLE SERVICE WEB STACK **9**

3.3.1. TRANSPORT SERVICE **9**

3.3.2. MESSAGERIE XML **9**

3.3.3. DESCRIPTION DU SERVICE **9**

3.3.4. SERVICE DISCOVERY **9**

3.4. LES TECHNOLOGIES STANDARDS AUTOUR DES SERVICES WEB **10**

3.4.1. REST **10**

3.4.2. SOAP **11**

3.4.3. LE LANGAGE WSDL **12**

3.4.4. UDDI **14**

4. RESEAUX SOCIAUX 16
4.1. DEFINITION 16
4.2. REPRESENTATION DES RESEAUX SOCIAUX 16
4.3. QUELQUES PRELIMINAIRES SUR LES GRAPHES 18
4.4. QUELQUES INDICATEURS DES RESEAUX 20
5. COMMUNAUTE 22
5.1. DEFINITION 22
5.2. TYPES DE COMMUNAUTES 23
5.2.1. COMMUNAUTE STATIQUE 23
5.2.2. COMMUNAUTE DYNAMIQUE 23
5.3. GENERALITE SUR LA DETECTION DE COMMUNAUTE 23
5.3.1. PARTITIONNEMENT DE GRAPHES 24
5.4. DETECTION DE COMMUNAUTES 24
6. LES ALGORITHMES DE DETECTION DE COMMUNAUTE 27
6.1. LES ALGORITHMES DE CLASSIFICATION HIERARCHIQUE 27
6.2. LES INCONVENIENTS DES APPROCHES EN INTERACTION UTILISATEURS 31
6.3. APPROCHE DE DECOUVERTE DE COMMUNAUTE BASEE SERVICE 32
7. CONCLUSION 34

CHAPITRE II : Analyse et Conception

1. INTRODUCTION 35
2. IDEE GENERALE 35
3. METHODOLOGIE 37
3.1. PROCESSUS D'INITIALISATION DU GRAPHE 39
3.1.1. DESCRIPTION DU CONTEXTE UTILISATEUR ET SERVICE 41
3.2. PROCESSUS DE CALCUL DE CONVENANCE 43
3.2.1. CALCUL DE LA CONVENANCE GLOBALE 44
3.2.2. CONVENANCE DES CHAMPS DE TYPE NUMERIQUE 45
3.2.3. CONVENANCE DES CHAMPS A VALEUR ENUMERATIVE 46
3.2.4. CONVENANCE DES CHAMPS A MULTI-VALEURS 47
3.2.5. CALCUL DE CONVENANCE BASEE CONTACT 47
3.2.6 CONVENANCE DES CHAMPS DE TYPE CHAINE DE CARACTERES 48
3.3. PROCESSUS DE GENERATION DES ACCES 50
3.3.1. PROCESSUS DE GENERATION DU GRAPHE DES ACCES FILTRE 50
3.4. PROCESSUS DE GENERATION DES COMMUNAUTES 51
4. MODELISATION 52
4.1. DEFINITION UML 52
4.2. ASPECT FONCTIONNEL 53

4.4. ASPECT STATIQUE 67
4.5. MODELE RELATIONNEL DE DONNEES 69
5. CONCLUSION 70

CHAPITRE III : Implémentation et Test

1. INTRODUCTION 71
2. LES OUTILS UTILISES 71
2.1. LES LANGAGES DE PROGRAMMATIONS UTILISES 71
2.1.1. JAVA 71
2.1.2. SQL 71
2.2. LES ENVIRONNEMENTS DE DEVELOPPEMENT 72
2.2.1. NETBEANS 72
2.2.2. MYSQL 73
2.3. PAJEK-XXL 73
2.4. PATTERN MVC 74
2.4.1. LE MODELE 74
2.4.2. LA VUE 75
2.4.3. LE CONTROLEUR 75
2.5. TECHNOLOGIE RMI 75
3. MISE EN MARCHE DE L'APPLICATION 76
3.4. GENERATION DES COMMUNAUTES 90
4. EXEMPLE D'USAGE 92
5. ETUDE COMPARATIVE 95
6. CONCLUSION 99

Résumé /ملخص

Résumé :

La détection des communautés occupe une grande importance dans divers domaines. Elle a attiré beaucoup d'attention ces dernières années. Détecter les communautés peut être un moyen d'identifier les sous-structures qui pourraient correspondre à des fonctions importantes. Une des caractéristiques les plus pertinentes des graphes représentant les systèmes réels est la structure de la communauté. Plusieurs méthodes de détection de communauté ont été proposées pour identifier les structures communautaires efficaces et d'analyser les propriétés du réseau. En effet, l'identification de la structure des communautés dans les réseaux complexes aide à extraire des informations utiles à analyser et à comprendre. Dans les réseaux sociaux les utilisateurs sont représentés par les nœuds et les liens représentent l'interaction entre ces utilisateurs. Le nombre d'utilisateurs participants à ces réseaux est important à raison de centaines de millions dans le monde. Des efforts de recherche ont été consacrés à développer des méthodes et des algorithmes qui peuvent efficacement mettre en évidence cette structure cachée d'un réseau, ce qui donne une vaste littérature sur la détection de communautés. Cependant, la plupart de ces méthodes ne convenaient pas pour les réseaux complexes car ils ont des limitations importantes.

Le travail consiste à trouver une méthode de découverte multicritères de communautés dans les réseaux complexes basée service, c'est une nouvelle approche prédictive de détection de communauté dans les réseaux complexes qui se base sur les services similaires aux besoins des utilisateurs en prenant en considération plusieurs contraintes à définir.

- **Mots-clés :** Réseau complexe, réseaux sociaux, convenance , services web, clustering (segmentation), Découverte de communautés, profilation, graphe biparti.

ملخص:

الكشف عن المجتمعات يأخذ أهمية كبيرة في مختلف المجالات. فقد جذب الكثير من الاهتمام في السنوات الأخيرة حيث يمكن أن يكون الكشف عن المجتمعات وسيلة لتحديد الهياكل الفرعية التي يمكن أن تتوافق مع المهام الهامة.

واحدة من السمات الأكثر أهمية في الرسوم البيانية التي تمثل الأنظمة الحقيقية هي بنية المجتمع. وقد اقترحت عدة طرق للكشف عنه لتحديد الهياكل المجتمعية الفعالة وتحليل خصائص الشبكة. في الواقع، تحديد بنية المجتمع في شبكات معقدة يساعد على استخراج معلومات مفيدة للتحليل والفهم.

في الشبكات الاجتماعية يمثل المستخدمين عن طريق العقد ويمثل التفاعل بين هؤلاء المستخدمين عن طريق الروابط بين العقد. عدد المستخدمين المشاركين في هذه الشبكات يصل إلى مئات الملايين في جميع أنحاء العالم. وقد كرست الجهود البحثية لتطوير الأساليب والخوارزميات التي يمكنها الكشف عن هاته الهياكل المخفية. ومع ذلك، فإن معظم هذه الطرق لم تكن مناسبة للشبكات المعقدة لأن لديهم حدود هامة.

عملنا هذا يتمثل في العثور على طريقة جديدة للكشف عن المجتمعات بعيدة معايير في الشبكة المعقدة استنادا على خدمات الواب. إنها طريقة جديدة للكشف عن المجتمعات قائمة على التنبؤ في الشبكات المعقدة تعتمد على الخدمات المماثلة لاحتياجات المستخدم، مع الأخذ بعين الاعتبار عدة قيود للتحديد.

الكلمات المفتاحية: شبكات معقدة، الشبكات الاجتماعية، التشابه، خدمات الإنترنت، التجميع (التجزئة)، اكتشاف المجتمعات، شكل بياني ثنائي

Liste des figures

Figure I.1 : Architecture de l'UDDI ….....8

Figure I.2 : Couches du protocole service web stack ……10

Figure I.3 : Fonctionnement du protocole SOAP ……11

Figure I.4 : Structure d'un message SOAP…….. 12

Figure I.5 : Schéma démontrant la structure d'un document WSDL……13

Figure I.6 : Schéma représentatif de l'annuaire UDDI ……15

Figure I.7 : Exemple sociogramme……...17

Figure I.8 : Exemple de graphe biparti…….19

Figure I.10. : Représentation graphique du Zachary Karate club…….26

Figure I.11. : Représentation graphique d'Internet……...27

Figure I.12. Les processus constituant l'approche de découverte de communautés basée service……...33

Figure II.1. Exemple d'un graphe biparti étiqueté……36

Figure II.2. Les processus constituant l'approche de découverte de communautés basée service ……...38

Figure II.3. Spécification des profils utilisateur et service……40

Figure II.4. Cas d'utilisation de génération des utilisateurs et des services……..53

Figure II.5. Cas d'utilisation génération du graphe de convenance …….54

Figure II.6. Cas d'utilisation génération des accès……...54

Figure II.7. Cas d'utilisation génération du graphe d'accès filtré…….55

Figure II.8. Cas d'utilisation pour la génération des communautés……..55

Figure II.9. Diagramme de séquence de lancement de l'application……56

Figure II.10. Diagramme de séquence contexte utilisateur/service……..57

Figure II.11. Diagramme de séquence calcul de convenance ……...58

Figure II.12. Diagramme de séquence pour la génération des accès……59

Figure II.13. Diagramme de séquence pour la génération du graphe d'accès filtrés……60
Figure II.14. Diagramme de séquence export fichier .net……61
Figure II.15. Diagramme de séquence pour le filtre intérêt utilisateur……61
Figure II.16. Diagramme de séquence relative au sexe ……...62
Figure II.17. Diagramme de séquence pour le filtre « Age »……62
Figure II.18. Diagramme de classe pour le filtre localisation……...63
Figure II.19. Diagramme de séquence pour le filtre « Connection »……64
Figure II.20. Diagramme de séquence des filtres : Religion, niveau d'étude, langue……..65
Figure II.21. Diagramme de séquence pour la génération de communautés……66
Figure II.22. Diagramme de classes……..67
Figure III.1. Interface du logiciel Pajek-XXL……..74
Figure III.2. Schéma du MVC……..75
Figure III.3. Schéma expliquant l'architecture de la technologie RMI……76
Figure III.4. Fenêtre « Profile Context (Service) »……..77
Figure III.5. Fenêtre « Profile Context (User) »……...78
Figure III.6. Fenêtre « Access Creation »……79
Figure III.7 Onglet filtre intérêt……....82
Figure III.8. Onglet filtre localisation_1 (Service)……...83
Figure III.9. Onglet filtre localisation_2 (Service)……...84
Figure III.10. Onglet filtre localisation_1 (Utilisateur)……84
Figure III.11. Onglet filtre localisation_2 (Utilisateur)……85
Figure III.12. Onglet filtre localisation_3 (Utilisateur)……85
Figure III.13. Onglet filtre Age et sexe (Age-option 1)……86
Figure III.14. Onglet filtre Age et Sexe (Age-option 2)……...86
Figure III.15. Onglet filtre Age et Sexe (Sexe)……87
Figure III.16. Onglet filtre Connection_1……87
Figure III.17. Onglet filtre Connection_2……88
Figure III.18. Onglet Autre Filtres……88
Figure III.19. Lancer la génération de communautés……...90

Figure III.20. Message de nombre d'accès créés……..92

Figure III.21. Message nombre d'accès trouvés……...93

Figure III.22. Visualisation du graphe de l'exemple ……94

Figure III.23. Visualisation du graphe des communautés…….94

Figure III.24 Zoom sur le graphe précédant …….95

Figure III.25. Variation du temps d'exécution en fonctions du nombre d'utilisateurs et de services …….96

Figure III.26. Variation du temps d'exécution en fonctions du nombre d'utilisateurs et de services (Algorithme Louvain)…….98

Liste des algorithmes

Algorithme II.1. Algorithme Calcul de convenance 81

Algorithme II.2. Algorithme Génération des accès ……82

Algorithme II.3. Algorithme Génération du graphe d'accès filtrés…….90

Algorithme II.4. Algorithme Génération des communautés .....…...92

Liste des tableaux

Tableau III.1. Tableau Comparatif de variation de temps d'exécution en fonction du nombre d'utilisateurs et de services ……97

Tableau III.2. Tableau Comparatif de variation de temps d'exécution en fonction du nombre d'utilisateurs et de services (Algorithme Louvain)…….98

Liste des formules

Formule de la convenance globale……...44

Formule de la convenance atomique ……...44

Liste des acronymes

XML : Extensible Markup Language

HTTP: HyperText Transfer Protocol

EDI: Electronic Data Interchange

CORBA: Common Object Request Broker Architecture

COM: Component Object Model

IBM: International Business Machines

SOAP: Simple Object Access Protocol

WSDL: Web Services Description Language

RPC: Remote Procedure Call

DCOM: Distributed Component Object Model

RMI: Remote Method Invocation

CGI: Common Gateway Interface

EJB: Entreprise JavaBeans

UDDI: Universal Description Discovery and Intégration

SMTP: Simple Mail Transfer Protocol

FTP: File Transfer Protocol

BEEP: Blocks Extensible Exchange Protocol

REST: Representational State Transfer

MIME: Multipurpose Internet Mail Extensions

SOA: Service Oriented Architecture

URL: Uniform Resource Locator

EBC: Edge Betweenness Centrality

Introduction générale

Introduction générale

Au cours des deux dernières décennies, nous sommes devenus les témoins d'un développement rapide de l'informatique distribué et des réseaux informatiques. Les utilisateurs qui étaient initialement restreints à accéder aux données textuelles statiques qui étaient disponibles sur Internet profitent maintenant du contenu multimédia qui est même produit par d'autres utilisateurs en temps réel.

La prolifération croissante et l'abordabilité des dispositifs Internet, ainsi que la facilité de publication, de recherche et d'accès à l'information sur le web encourage les utilisateurs individuels à communiquer leur contenu avec la société web. Cela a donné naissance à l'idée d'interaction sociale sur Internet qui en conjonction avec l'avènement de la technologie Web 2.0 a conduit à l'apparition des premiers grands médias sociaux.

Ces applications sont aujourd'hui sources d'une grande partie des informations qui sont échangées via Internet tous les jours. Ils produisent des ensembles de données de taille massive qui ont tendance à évoluer au fil du temps.

Cet impact majeur des médias sociaux sur l'industrie des technologies de l'information a émergé une variété de problèmes concernant les réseaux complexes. Dans le contexte de la théorie des réseaux, les réseaux sociaux complexes sont définis comme des graphes de caractéristiques topologiques non triviales se produisant dans des graphes réels. Des réseaux dans différents domaines d'application présentent une structure interne, dans laquelle où les nœuds forment des groupes de composants étroitement connectés qui sont reliés plus isolé au reste du réseau. Ces composantes sont surtout connues sous le nom de communautés, de grappes ou de groupes.

La découverte de la structure communautaire d'un réseau est un problème fondamental dans des réseaux complexes, déjà appliquée avec succès dans un large éventail de disciplines scientifiques, y compris la physique, la biologie, la sociologie, les sciences sociales, les mathématiques et l'informatique.

Les communautés sont des structures mésoscopiques du réseau, connues pour être présentes dans la plupart des graphes de terrain, et en particulier dans les réseaux de type petit monde. On peut les définir comme des ensembles de nœuds fortement liés entre eux, et plus faiblement liés

avec le reste du réseau. Ces structures ont du sens dans les réseaux : par exemple, dans un réseau social, ces communautés pourront correspondre à des groupes sociaux (famille, groupe d'amis, etc.). Elles pourront correspondre à des champs lexicaux dans des réseaux de synonymie, ou encore à des blogs traitant de sujets similaires dans une analyse de pages web, pour donner quelques exemples.

La détection de communautés est un domaine qui forme un grand enjeu de cette décennie du web, son principal objectif est d'identifier automatiquement des groupes pertinents.

Parmi les applications dont la détection de communauté peut s'appliquer est la compréhension de la structure des réseaux, la visualisation, l'amélioration des moteurs de recherches, système P2P, la détection de fraude et de groupes de terroristes, il est aussi utilisé dans le domaine du commerce électronique.

Dans le cadre de ce mémoire, nous exposerons une nouvelle approche de détection de communauté qui est la découverte multicritères basée services. Nous présentons ci-après le contenu de chacun des chapitres :

Le premier chapitre est consacré à l'étude de l'existant, il introduit les services web en décrivant les technologies essentielles le constituant, aussi il fait un tour d'horizon sur le domaine des réseaux sociaux mais aussi la théorie des graphes, et il est conclu par la présentation des structures communautaires et les nombreux travaux du domaine de détection de communautés.

Le second chapitre concerne la modélisation et la conception de notre approche, il présente de façon précise les quatre piliers sur lesquels est bâtie l'approche de découverte de communautés, en passant par la modélisation par plusieurs aspects en utilisant le langage UML.

Le dernier chapitre est accordé à la réalisation des processus décrit dans le chapitre précédant, les différents outils et technologies utilisées seront présentés, ensuite on passera au résultat obtenu lors de l'usage de l'application, et on terminera par une étude comparative avec une autre méthode celle de l'algorithme de Louvain.

I

Etat de l’art

1. INTRODUCTION

Ces dernières années, l'accroissement du nombre d'utilisateurs dans le world wide web n'a pas cessé d'augmenter. Des réseaux sociaux tels que Facebook, Twitter ou Google+ attirent des centaines de millions d'utilisateurs. Ce développement a transformé le World Wide Web en une énorme industrie avec plus de 30000 services web disponibles auprès de quelques 8000 fournisseurs[1].

Les données gigantesques associées aux utilisateurs et leurs interactions avec d'autres constituent la base de la découverte de nombreuses relations inconnues telles que la détection des communautés. Dans ce premier chapitre, nous commencerons par introduire les services web (une définition, les technologies utilisées, les caractéristiques et les objectifs), nous passerons ensuite vers les réseaux sociaux où nous aborderons les principes primordiaux en passant par les notions de la théorie des graphes, et nous clôturons par la détection de communauté objet de cette thèse.

2. SERVICES WEB

2.1. DEFINITION

Un service web ou en anglais « web services » est une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Pour ce faire, les services Web s'appuient sur un ensemble de protocoles Internet très répandus (XML, HTTP), afin de communiquer. Cette communication est basée sur le principe de demandes et réponses effectuées avec des messages XML. [1]

On retrouve plusieurs définitions des services web :

Citation : Comment ça marche

Les services web représentent un mécanisme de communication entre des applications distantes à travers le réseau internet indépendant de tout langage de programmation et de toute plate-forme d'exécution.

1 http://webservices.seekda.com, as of 07/01/2013

Citation : Wikipédia

Un service web est un protocole d'interface informatique de la famille des technologies web permettant la communication et l'échange de données entre les applications et les systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans aucune intervention humaine, de manière synchrone ou asynchrone.

2.2. HISTORIQUE ET ORIGINE DES CONCEPTS

Les services web sont nés de l'effort de plusieurs organisations qui ont partagé un intérêt commun en développant et en maintenant "un marché électronique". Celles-ci souhaitaient pouvoir communiquer plus simplement et sans avoir à se concerter sur chacune de leur transaction pour pouvoir interpréter leurs différentes données. C'est ainsi que naquit en 1975 l'EDI (Échange de Données Informatisées). [1]

L'EDI peut être défini comme l'échange, d'ordinateur à ordinateur, de données concernant des transactions en utilisant des réseaux et des formats normalisés. Cet échange permet l'entente entre les différentes applications sur la modélisation des échanges et sur les protocoles de communication.

Vers la fin des années 80, l'évidence fut que l'âge des systèmes informatiques isolés touchait à son terme tandis que différents ordinateurs, de tailles, de capacités et de formes variées, apparaissaient au sein d'une même organisation.

Pour répondre à cette nouvelle situation, de nouvelles technologies apparurent telles que CORBA (Common Object Request Broker Architecture) ou la version qu'en fit Microsoft, le COM (Component Object Model).

Les années 90 furent témoin non seulement de la recrudescence des ordinateurs personnels, mais aussi du décollage phénoménal d'Internet avec une demande grandissante de standards susceptibles de travailler sur n'importe quelle plateforme.

A la fin des années 90, l'e-speak d'Hewlett Packard fit son apparition en même temps que XML, tandis que Microsoft et IBM avait déjà pensé à une alternative grâce à EDI.

Ils tentèrent de coder des transactions EDI en XML pour permettre de faciliter les relations inter-entreprises grâce au web.

A peu près à la même époque, un regroupement d'organisations recherchant une façon de structurer et d'échanger des documents XML créa un protocole appelé Simple Object Access Protocol (SOAP) qui permet la transmission de messages entre applications distantes.

On peut affirmer que c'est avec SOAP que le concept de services web est définitivement apparu renforcé par la création de WSDL (Web Services Description Language) qui donne la description au format XML des services web.

Aujourd'hui, les services web provoquent un intérêt certain auprès des architectes et des décideurs. Par contre, la normalisation complète d'une architecture distribuée construite sur les services web n'est pour l'instant pas encore tout à fait établie, mais même si ils n'ont pas acquis la maturité nécessaire à leur industrialisation, ils se présentent plus que jamais comme la solution appropriée aux problématiques d'échange de données et d'intégration d'applications.

2.3. INTERETS DES SERVICES WEB (OBJECTIFS)

Un service web est un mécanisme qui tend à donner plus d'interactions pour permettre à deux entités hétérogènes (entreprises, clients, applications, etc. ...) de dialoguer à travers le réseau informatique. Les logiciels écrits dans divers langages de programmation sur diverses plateformes et avec diverses architectures peuvent employer des services web pour échanger des données à travers des réseaux informatiques. [2]

Les services web ont été mis en place afin de répondre à un certain nombre de besoins :

- Remplacer les protocoles (RPC – Remote Preocedure Call -, DCOM – Distributed Component Object Model -, RMI –Remote Method Invocation-) par une approche entièrement ouverte et interopérable basée sur la généralisation des serveurs web avec scripts CGI (Common Gateway Interface)
- Faire interagir des composants hétérogènes, distants et indépendants avec un protocole standard (SOAP)
- Simplifier la communication entre ces composants
- Ne pas créer de nouvelles technologies, mais se baser sur celles qui existent déjà (XML, HTTP)

- Mise en place d'un mécanisme de transfert de données compatible avec de nombreux environnements de développement (indépendant du langage de programmation, OS... etc)
- Offrir des applications accessibles à distance

Les services web sont grandement utilisés par les entreprises, ce qui leur permet d'exposer un certain nombre de services et d'échanger les informations entre elles.

3. CARACTERISTIQUES DES SERVICES WEB

La technologie des services web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur l'internet et offre beaucoup d'opportunités au développeur web. [5]

3.1. ASPECTS DES SERVICES WEB

Les services web ont les caractéristiques comportementales particulières suivantes :

3.1.1. Usage du XML

Un service web utilise XML pour la représentation des données et de transport de données des couches ainsi car étant un protocole interopérable.

3.1.2. Couplage lâche

Un consommateur d'un service web n'est pas lié à ce service directement, son interface peut changer au fil du temps, sans compromettre la capacité du client d'interagir avec le service.

L'adoption d'une architecture faiblement couplée tend à rendre les systèmes logiciels plus faciles à gérer et permet une intégration plus simple entre les différents systèmes. [5]

3.1.3. Gros grains

Les entreprises et les interfaces qu'elles exposent devraient être à gros grains. La technologie des services web fournit un moyen naturel de la définition des services de gros causes qui accèdent à la bonne quantité de logique métier. [5]

3.1.4. Capacité à être synchrone ou asynchrone

La synchronicité se réfère à la liaison du client à l'exécution du service. Dans les appels synchrones, les blocs client attendent le service de terminer son opération avant de continuer.

Les opérations asynchrones permettent à un client d'appeler un service, puis d'exécuter d'autres fonctions.[5]

3.1.5. Prise en charge des appels de procédure à distance (RPC)

Un service web prend en charge le Remote Procedure Call (RPC), en fournissant des services de son propre, équivalentes à celles d'un composant traditionnel, ou en traduisant les invocations entrantes en une invocation d'un Entreprise Java Beans (EJB) ou un composant .NET.

3.1.6. Prend en charge l'échange de documents

L'un des principaux avantages de XML est sa manière générique de représenter non seulement des données, mais aussi des documents complexes. Ces documents peuvent être aussi simples que représentant une adresse actuelle ou ils peuvent être aussi complexes que représentant un livre entier ou un appel d'offre. Les services web prennent en charge l'échange transparent de documents pour faciliter l'intégration de l'entreprise.

3.2. ARCHITECTURE ET FONCTIONNEMENT

Pour comprendre le fonctionnement d'une architecture de service web, il faut savoir que les services web sont des "applications modulaires basées sur Internet qui exécutent des tâches précises et qui respectent un format spécifique"[2]; cela veut dire, au niveau conceptuel, que le terme « service web » mettrait en avant les qualités d'une fonctionnalité commerciale présentée par une entité hétérogène quelconque sur l'internet afin de fournir un moyen d'utiliser ce service à distance, et en ce qui concerne l'aspect opérationnel, les services web ne sont que des applications modulaires qui peuvent être présentées, publiées, situées et invoquées dans un réseau et ce automatiquement.

3.2.1. Rôles de service Web

Le rôle principal d'un service web consiste à examiner les rôles individuels de chaque acteur du service web, aussi appeler architecture de référence, elle est constituée de trois grands rôles :

[2] Définition de Mark Colan, Web Service and XML Chief Advocate chez IBM

3.2.2. Fournisseur de services

Ceci est le fournisseur du service web. Le prestataire des services met en œuvre le service et le rend disponible sur l'internet.

3.2.3. Demandeur Service (Client)

Ceci est un consommateur du service web. Le demandeur utilise un service web existant en ouvrant une connexion réseau et l'envoi d'une requête XML.

3.2.4. Annuaire UDDI :

Ceci est un répertoire logique centralisé des services. Le registre fournit un lieu central où les développeurs peuvent publier de nouveaux services ou de trouver ceux qui existent déjà. Il sert donc comme une chambre de compensation centralisée pour les entreprises et leurs services.

Cette architecture est illustrée dans la figure suivante :

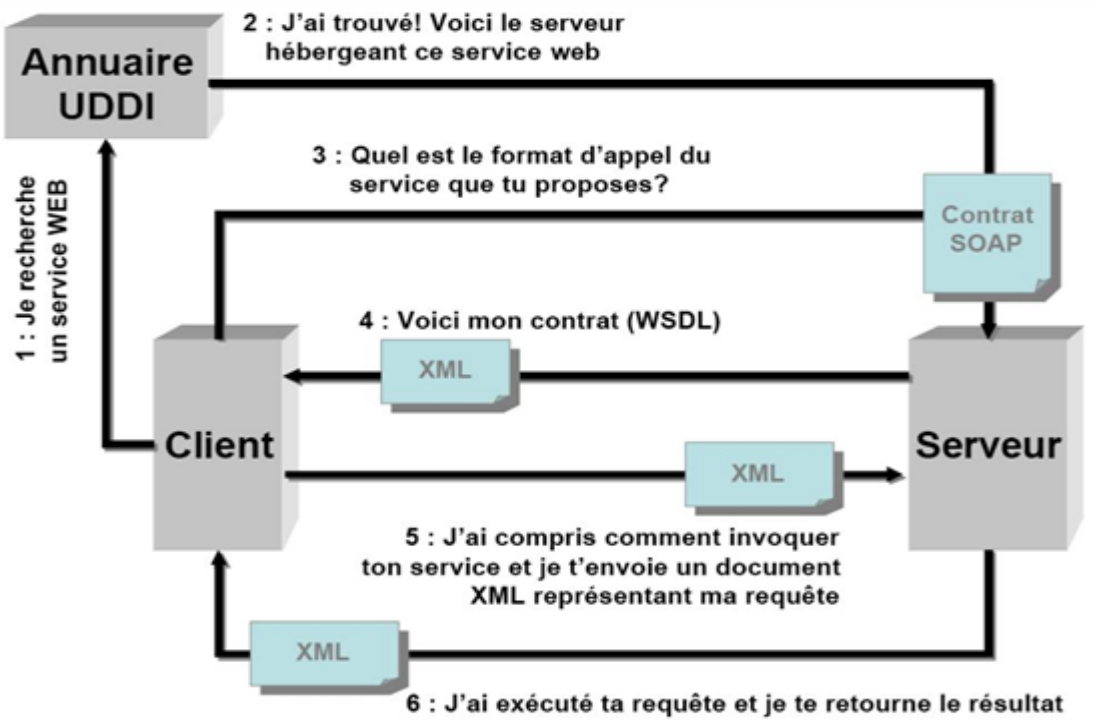


Figure I.1. Architecture de l'UDDI

1. Le client envoie une requête à l'annuaire de service pour trouver le service web dont il a besoin.
2. L'annuaire cherche pour le client, trouve le service web approprié et renvoie une réponse au client en lui indiquant quel serveur détient ce qu'il recherche.

3. Le client envoie une deuxième requête au serveur pour obtenir le contrat de normalisation de ses données
4. Le serveur envoie sa réponse sous la forme établie par WSDL en langage XML.
5. Le client peut maintenant rédiger sa requête pour traiter les données dont il a besoin.
6. Le serveur fait les calculs nécessaires suite à la requête du client, et renvoie sa réponse sous la même forme normalisée.

3.3. PROTOCOLE SERVICE WEB STACK

Une deuxième option pour visualiser l'architecture des services web est d'examiner la pile de protocoles de services web émergents. La pile est encore en évolution, mais dispose actuellement de quatre couches principales. [3]

3.3.1. Transport Service

Cette couche est responsable du transport de messages entre les applications. Actuellement, cette couche comprend HTTP (Hyper Text Transport Protocol), le protocole de transfert de courrier simple (SMTP), protocole de transfert de fichiers (FTP), et des protocoles plus récents tels que les Blocks Extensible Exchange Protocol (BEEP).

3.3.2. Messagerie XML

Cette partie est responsable du codage des messages dans un format XML commun de sorte que les messages puissent être compris à chaque extrémité. Actuellement, cette couche comprend XML-RPC et SOAP (Simple Object Access Protocol).

3.3.3. Description du service

Cette branche est responsable de la description de l'interface publique à un service web spécifique. A l'époque actuelle, la description du service est assurée via le Web Service Description Language (WSDL).

3.3.4. Service Discovery

Ce service est chargée de centraliser les services dans un registre commun et de fournir des fonctionnalités de publication / recherche faciles. À l'heure actuelle, la découverte de service est gérée via Universal Description Discovery and Intégration (UDDI).

Comme les services web évoluent, des couches supplémentaires peuvent être ajoutées et des technologies supplémentaires peuvent être ajoutées à chaque couche.

Le schéma (Figure I.2) suivant présente les couches essentielles du protocole service web stack mentionné ci-dessous :

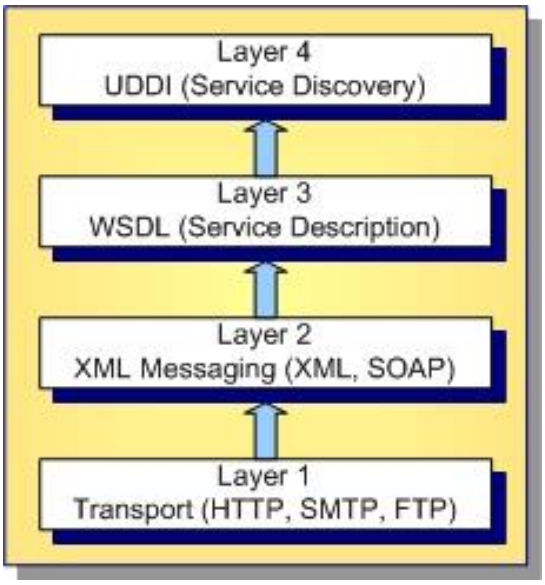


Figure I.2. Couches du protocole service web stack [4]

3.4. LES TECHNOLOGIES STANDARDS AUTOUR DES SERVICES WEB

Les services Web reprennent la plupart des idées et des principes du Web (HTTP, XML) et les appliquent à des interactions entre machines. Comme pour le World Wide Web, les services web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante.
Les principales technologies utilisées par les services Web sont REST, SOAP, WSDL et UDDI. [5]

3.4.1. REST

REST (Representational State Transfer) est une architecture de services web. Élaborée en l'an 2000 par Roy Fiedling (l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres travaux fondamentaux). Ce style architectural s'applique tout

autant à la réalisation d'applications pour un utilisateur humain qu'à la réalisation d'architectures orientées services destinées à la communication entre machines.
Les systèmes qui suivent les principes de l'architecture REST sont souvent appelés RESTful.

3.4.2. SOAP

SOAP est une note du Consortium W3C dont Microsoft fait partie. C'est un protocole de dialogue par appels de procédures à distance entre objets logiciels. Basé sur XML, SOAP a pour principal objectif d'assurer la communication entre machines. Il prend en charge divers protocoles de transport, tels que HTTP et SMTP, ainsi que différents formats comme MIME (Multipurpose Internet Mail Extensions). Ces derniers sont très répandus sur de multiples plates-formes ; ce qui donne à SOAP une grande portabilité et interopérabilité, c'est ce qui le rend indépendant des plates-formes et langages de programmation. [23]
La figure I.3 Illustre le fonctionnement du protocole SOAP :

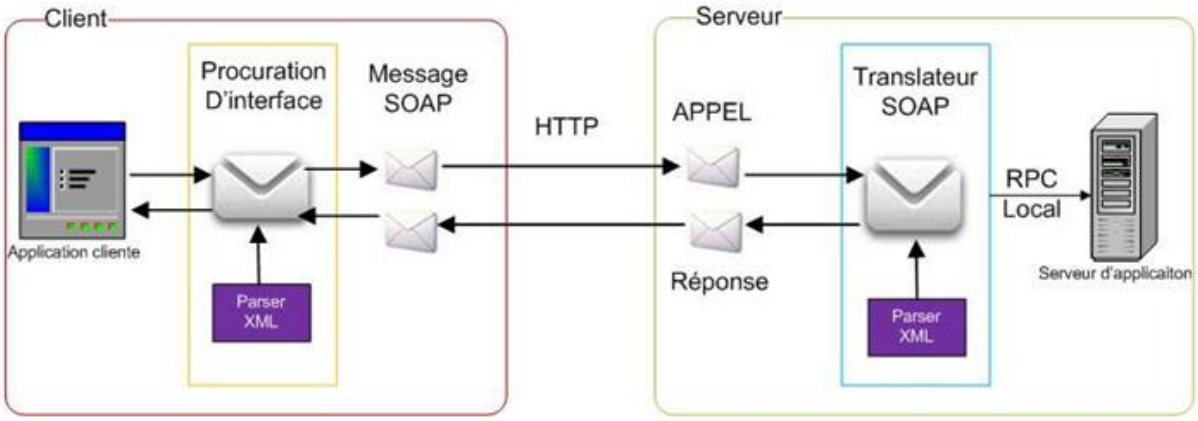


Figure I.3. Fonctionnement du protocole SOAP [5]

Les deux plus fortes fonctionnalités de SOAP sont sa simplicité et le fait qu'il est devenue un standard. SOAP définit un format pour l'envoi des messages. Les messages SOAP sont structurés en un document XML et comportent deux parties obligatoires : l'enveloppe SOAP et le corps SOAP, et une partie optionnelle l'en-tête SOAP :

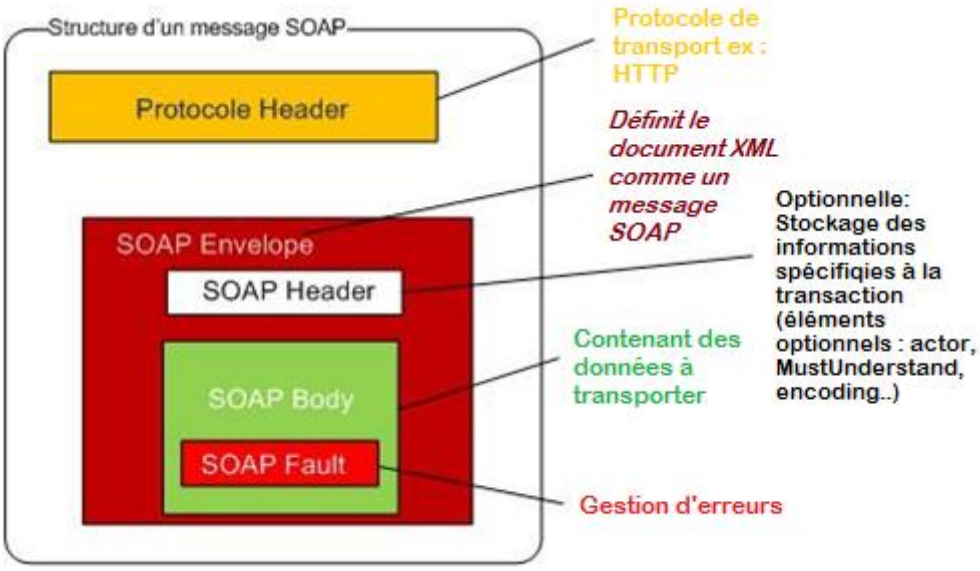


Figure I.4. Structure d'un message SOAP [5]

- **Enveloppe SOAP** (envelope) est l'élément de base du message SOAP. L'enveloppe contient la spécification des espaces de désignation (namespace) et du codage de données.
- **Entête SOAP** (header) est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent. C'est ici qu'il est indiqué si le message est mandataire ou optionnel. L'entête est utile surtout quand le message doit être traité par plusieurs intermédiaires.
- **Corps SOAP** (body) c'est un container pour les informations mandataires à l'intention du récepteur du message, il contient les méthodes et les paramètres qui seront exécutées par le destinataire final.
- **Erreur SOAP** (fault) est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs. [5]

3.4.3. LE LANGAGE WSDL

WSDL (Web Services Description Language) est un langage de description standard. Il indique comment utiliser le service web et comment interagir avec lui. WSDL est basé sur XML et permet de définir de façon précise les détails concernant l'endroit où le service est disponible et le protocole de communications utilisé pour converser avec le service. Cela signifie que le fichier WSDL définit tout ce qui est nécessaire pour écrire un programme fonctionnant avec un service Web.

Le schéma suivant éclairci la structure du langage WSDL, en décrivant les relations entre les sections constituant un document WSDL.

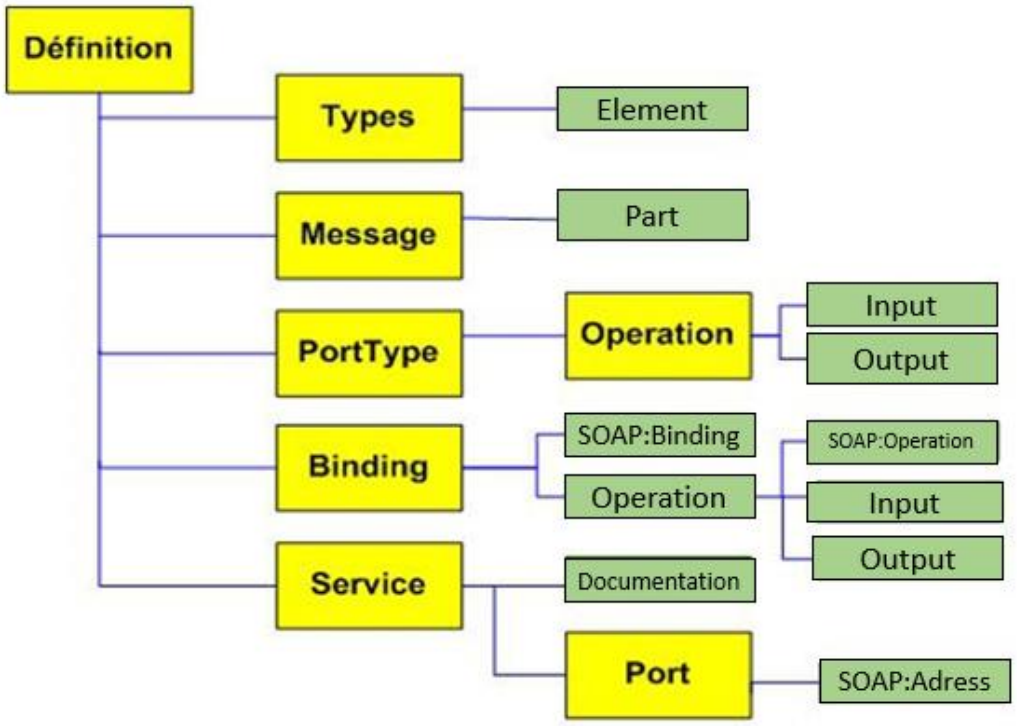


Figure I.5. Schéma démontrant la structure d'un document WSDL [6]

Un fichier WSDL contient donc sept éléments :

- **Types** : fournit la définition de types de données utilisées pour décrire les messages échangés
- **Messages** : représente une définition abstraire (noms et types) des données en cours de transmission.
- **PortTypes** : décrit un ensemble d'opérations. Chaque opération a au maximum un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.
- **Binding** : spécifie une liaison entre un <portType> et un protocole concret (SOAP, HTTP...).
- **Service** : indique les adresses de port de chaque liaison
- **Port** : représente un point d'accès de services défini par une adresse réseau et une liaison.
- **Opération** : c'est la description d'une action exposée dans le port

En résumé, WSDL est un contrat entre un client et un serveur qui fait état :

- Des spécifications d'interfaces qui décrivent toutes les méthodes publiques
- Des spécifications relatives aux types de données de messages mis en œuvre dans les questions-réponses
- Des informations liées au protocole de transport utilisé
- Des informations d'adresse permettant de localiser le service décrit

3.4.4. UDDI

Né d'une collaboration entre IBM et ARIBA, Universal Description Discovery and Integration contient les registres Internet globaux regroupant les services et descriptions business depuis septembre 2000.

UDDI est une interface fournissant l'infrastructure de base pour la publication et la découverte des services web. C'est une norme d'annuaire de services web qui se concentre sur le processus de découverte de l'architecture orientée services (SOA) et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre partenaires dans le cadre des échanges commerciaux.

Une entrée du répertoire UDDI est constituée d'un fichier XML qui décrit une entreprise et les services qu'elle offre. Chaque entrée du répertoire UDDI est constituée de trois parties.

- **Les pages blanches :** contiennent des informations de base sur l'entreprise, les coordonnées de base, y compris le nom de l'entreprise, l'adresse, le numéro de téléphone de contact …etc. Elles comportent aussi des identificateurs uniques pour les ID d'impôt des sociétés. Cette information permet à d'autres de découvrir votre service Web en fonction de votre identification d'entreprise.
- **Les pages jaunes** : contiennent plus de détails sur l'entreprise. Ils comprennent des descriptions du type de capacités électroniques que l'entreprise peut offrir à quiconque veut faire des affaires avec elle.

  Les pages jaunes utilisent des schémas de catégorisation industrielle couramment acceptés, des codes sectoriels, des codes produits, des codes d'identification d'entreprise et autres afin de faciliter aux entreprises la recherche et l'exploitation dans les listes et de trouver exactement ce qu'elles veulent.

- **Les pages vertes** : contiennent des informations techniques sur un service Web. Une page verte permet à quelqu'un de se lier à un service Web après qu'il a été trouvé. Elles comprennent :
  - ✓ Les différentes interfaces
  - ✓ Les adresses URL
  - ✓ Les informations de découverte et données similaires requises pour trouver et exécuter le service Web.

Les entreprises publient les descriptions de leurs services Web en UDDI, sous la forme de fichiers WSDL. Ainsi, les clients peuvent plus facilement rechercher les services Web dont ils ont besoin en interrogeant le registre UDDI.

Lorsqu'un client trouve une description de service Web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI. Ensuite, à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service Web, le client peut invoquer le service web et lui demander d'exécuter certaines de ses fonctionnalités.

Le scénario classique d'utilisation de l'UDDI est illustré ci-dessous. L'entreprise B a publié le service Web S, et l'entreprise A est client de ce service :

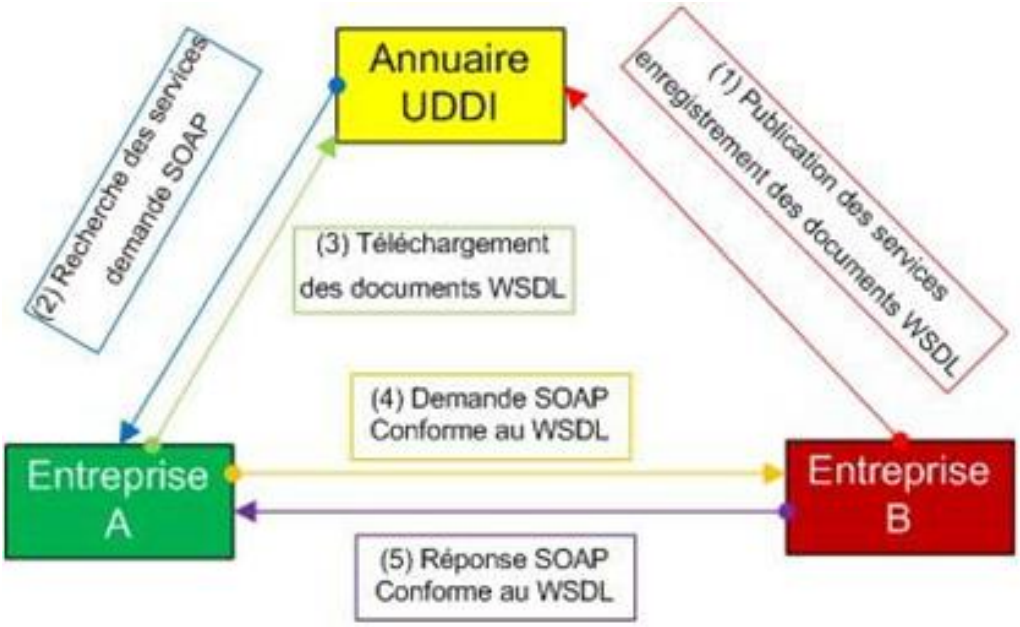


Figure I.6. Schéma représentatif de l'annuaire UDDI [5]

4. RESEAUX SOCIAUX

4.1. DEFINITION

Un réseau social est un ensemble d'identités sociales telles que des individus ou encore des organisations reliées entre elles par des liens créés lors des interactions sociales. Il se représente par une structure ou sous une forme dynamique d'un groupement social. Un réseau social représente une structure sociale dynamique se modélisant par des sommets et des arrêtes. Les sommets désignent généralement des gens (et/ou des organisations) qui sont reliées entre elles par des interactions sociales (des arrêtes). [24]

Les réseaux sociaux sont des médias sociaux devenus incontournables qui « désignent un ensemble de services permettant de développer des conversations et des interactions sociales sur internet ou en situation de mobilité »[3].

4.2. REPRESENTATION DES RESEAUX SOCIAUX

La notion de réseau existe dans plusieurs domaines de recherche en informatique et même dans d'autres disciplines. La modélisation des réseaux par des graphes facilite l'étude et la compréhension de leur structure en faisant appel aux notions de la théorie des graphes.

Les graphes ont une longue histoire qui remonte au problème des sept ponts de Königsberg et à sa résolution par Leonhard Euler, en 1759.

Le problème est souvent présenté comme étant : à partir d'un plan comportant divers lieux et des ponts les reliant, existe-t-il un chemin passant une fois et une seule fois par tous les ponts et nous ramenant au point de départ ? L'approche utilisée pour la résolution est un des plus anciens exemples d'application de la théorie des graphes.

Il faudra attendre le vingtième siècle pour que la théorie des graphes devienne un domaine disjoint de la combinatoire, au sein des mathématiques. Mais des représentations de diverses structures sous formes de graphes apparaîtront entre temps, comme dans les travaux de Gustav Kirchoff en électrodynamique ou de Gilbert Lewis en chimie…etc

Les premières utilisations de graphes pour étudier des problématiques de sciences sociales sont l'œuvre du psychiatre Jacob L. Moreno et de la psychologue Helen Jennings dans les années

[3] Cavazza Frédéric, « Une définition des médias sociaux », 29 juin 2009)

1930. Ils nommèrent cela sociométrie. Ils nomment « *Sociogramme* » cette méthode de visualisation. Cela est illustré dans la figure suivante (figure I.7)

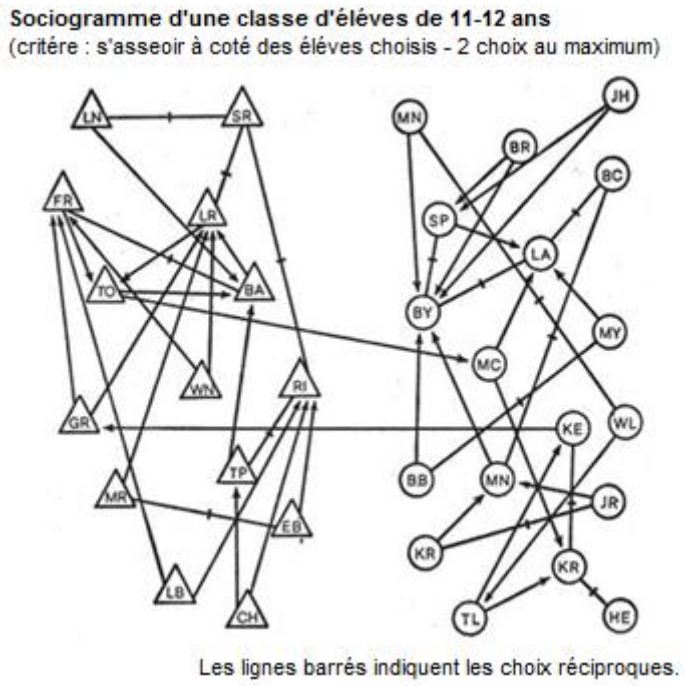


Figure I.7. Exemple sociogramme [7]

De ce fait Jacob Moreno a été un des pionniers à utiliser la représentation « nœud-lien » pour communiquer sur ses travaux. Dans cette représentation, un nœud représente un acteur et un lien représente une relation du réseau. Il s'agit de la représentation la plus courante des réseaux.

L'autre représentation courante est la représentation matricielle qui est une représentation beaucoup moins utilisée que les diagrammes nœuds-liens. Bien que moins intuitives, les matrices ont d'intéressantes propriétés, elles permettent notamment de passer à l'échelle et de traiter de très grands graphes.

La représentation matricielle de graphe se base principalement sur la matrice d'adjacence.

Cette dernière représente chaque sommet d'un réseau à la fois comme une ligne et comme une colonne. Si deux sommets sont connectés, la case correspondant à l'intersection de la ligne et de la colonne est marquée. Traditionnellement, on utilise une valeur numérique (0 marquant l'absence de connexion, 1 marquant la présence), soit par une marque graphique.

4.3. QUELQUES PRELIMINAIRES SUR LES GRAPHES

L'analyse des réseaux sociaux regroupe un ensemble de théories et d'outils s'appliquant aux graphes, un graphe étant un objet mathématique composé de nœuds (les individus) et d'arrêtes (les relations entre les individus). Elle considère que le comportement des individus d'une société résulte des relations d'interdépendance existant entre eux et s'intéresse à la forme que prend l'ensemble de ces relations, des conditions ayant amené à cette structure et aux propriétés que le groupe peut en tirer.

Au niveau du langage, on parle de graphe lorsqu'il s'agit de l'objet mathématique, tandis qu'il s'agit d'un réseau dès qu'un contexte lui est attribué, par un exemple pour un réseau social.

Nous présentons un graphe G tel que G = (V, E) avec :

- V l'ensemble des sommets
- E l'ensemble des arrêtes
- n=|V| est le nombre de sommets
- m=|E| est le nombre d'arrêtes

• Un **sommet (nœud)** est l'unité de base d'un réseau, il en représente une ressource (dans un réseau social on parle d'acteur).

• Une **arrête (un arc)** est une connexion entre deux sommets. On parle également de lien.

• Une **arrête est orientée (arc)** si elle ne s'utilise que dans une seule direction. Inversement, on parle **d'arrête non orientée** pour une arête qui s'utilise dans les deux sens.

• Une **arrête est pondérée** lorsqu'on lui attribue un poids.

• Une **arête est étiquetée** lorsqu'on lui attribue un label.

• Un **graphe orienté** désigne un graphe avec des arrêtes orientées (arcs).

• Un **graphe pondéré** désigne un graphe avec des arrêtes pondérées.

• Un **graphe étiqueté** désigne un graphe avec des arrêtes étiquetées.

• Un **graphe multipartite** désigne un graphe avec des sommets de types différents.

• **Le degré d'un sommet** d'un graphe non orienté est le nombre de liens d'arêtes reliant ce sommet, avec les boucles comptées deux fois, par contre dans un graphe orienté on parle de **degré entrant** d'un sommet $\boldsymbol{d^-(s)}$, c'est-à-dire le nombre d'arcs dirigés vers le sommet **s** , et du **degré sortant** de ce sommet $\boldsymbol{d^+(s)}$, c'est-à-dire le nombre d'arcs sortant de **s**. On a le degré du sommet est la somme du degré sortant et du degré entrant.

• Un **chemin** est une séquence d'arrêtes qui relie deux sommets.

• Un **chemin orienté** est une séquence d'arrêtes qui relie deux sommets en respectant l'orientation du parcours à chaque arrête.

• Une **géodésique** est l'un des plus courts chemins entre deux sommets donnés.

• Le **diamètre** d'un graphe est le plus long chemin géodésique de ce graphe.

• Un **graphe est complet** lorsqu'il existe une arrête entre toute paire de sommets.

• Un graphe est dit **fortement connexe** lorsqu'il existe un chemin entre toute paire de sommets.

• une **n-clique** est un sous-graphe G′ tel que la distance entre les nœuds de G′ soit inférieure à n dans G. L'ensemble G′ peut ne pas être connecté ou avoir un diamètre supérieur à n.

• Un **graphe biparti** $G = (V, E)$ est un graphe possédant deux classes de sommet disjoints V_r et V_b avec $V = V_r \cup V_b$ et $V_r \cap V_b = \emptyset$ telles qu'il n'existe aucune arrête reliant deux sommets de la même classe. Autrement dit, on a :

$E \cap (V_r \times V_r) = \emptyset$ et $E \cap (V_b \times V_b) = \emptyset$.

Exemple :

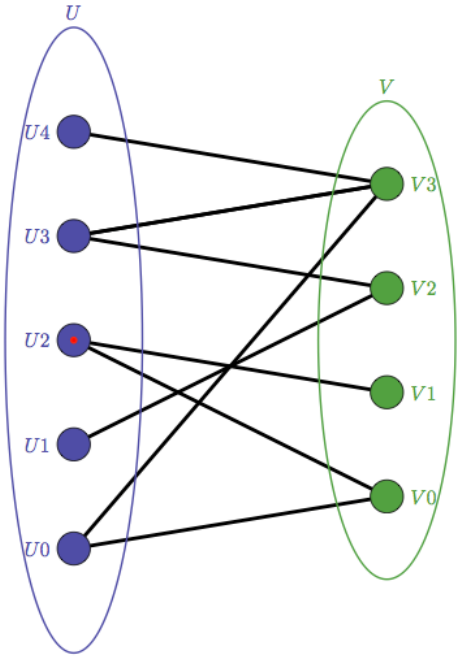


Figure I.8. Exemple de graphe biparti

Ce type de graphe est utilisé pour modéliser des réseaux dans lesquels on identifie deux types distincts de nœuds.

•**Réseau de terrain** : Un réseau de terrain est un réseau constitué à partir de données collectées, et qui correspond à une réalité de terrain. Par opposition, les réseaux générés ne correspondent à aucune donnée du monde réel.

•Réseau petit monde (Small-world network)

Selon les auteurs, la notion de réseau petit monde peut être légèrement différent. Le consensus général est de les caractériser par les propriétés suivantes :

- Faible distance entre les nœuds : le nombre moyen de pas nécessaire pour joindre deux nœuds du réseau choisis aléatoirement par la plus courte chaine reste très faible, quelle que soit la taille du réseau. Cette distance moyenne est de l'ordre de ln(n), n étant le nombre de nœuds du réseau.
- Loi de puissance de la distribution des degrés des nœuds du réseau. Cette propriété n'est cependant pas vérifiée pour certains réseaux. Facebook, par exemple, présente une distribution en loi de puissance tronquée, étant donné l'absence de hubs de très fort degré.
- Fort clustering : cette propriété signifie que les nœuds ont tendance à créer des structures locales denses. Cette propriété peut venir notamment de la transitivité : s'il existe un lien (a, b) et un lien (b, c), alors la probabilité qu'un lien (a, c) existe est renforcée.
- Structure en communauté : cette propriété est fortement liée à la précédente. Dans un réseau petit monde, on observe des structures mésoscopique, c'est-à-dire des ensembles de nœuds fortement liés entre eux et plus faiblement liés au reste du réseau.

Les graphes orientés sont adaptés pour représenter des relations non symétriques comme les réseaux de confiance par exemple.

Les graphes pondérés sont adaptés aux réseaux sociaux qui contiennent différents niveaux d'intensités dans les relations.

Les graphes étiquetés permettent de représenter différents types de relations. Les graphes multipartites sont adaptés pour des réseaux sociaux incluant différent types de ressources manipulées par les acteurs et qui sont le support d'interactions.

4.4. QUELQUES INDICATEURS DES RESEAUX

La Densité indique la quantité de liens au sein d'un réseau et permet de définir la cohésion d'un réseau social. [8]

Une analyse centrée sur l'individu consiste à mesurer la densité des liens autour d'un nœud donné. Une telle analyse montre notamment l'influence du nœud analysé sur la densité du sous graphe auquel il appartient avec ses voisins. Une analyse socio-centrée considère la densité sur l'ensemble du graphe et mesure la contrainte du réseau sur ses membres. Le calcul de la densité

est relatif au nombre maximal de lignes que peut contenir un graphe. Or, ce nombre maximal est lui-même fonction de la taille du graphe, ainsi toute comparaison de densité entre graphes ne fournit aucun résultat significatif.

La centralité d'un réseau social a été largement discutée. La problématique est de définir ce qui rend un nœud plus central qu'un autre, on parle alors de centralité locale.
Plusieurs approches ont été considérées. En 1979, Freeman a repris l'ensemble de ces approches et on a conclu trois principales centralités. [9]
La première approche appelée *centralité de degré* [10], Freeman considère comme centraux les nœuds qui possèdent les degrés les plus élevés du graphe. En effet, ces nœuds suscitent un grand intérêt, sont très visibles et ont un potentiel élevé à faire circuler l'information par leur forte connectivité aux autres éléments du réseau.
La centralité d'intermédiarité [9]; Freeman se concentre sur la capacité d'un nœud à servir d'intermédiaire dans un graphe. Un sommet situé sur un chemin géodésique possède une position stratégique dans la cohésion d'un réseau et dans la circulation de l'information, d'autant plus si ce chemin est unique. Plus un nœud est intermédiaire, plus le réseau est dépendant de lui et plus il a de pouvoir.
Enfin, *la centralité de proximité* [9]; Freeman mesure la centralité d'un nœud en se basant sur la taille des chemins qui le lient aux autres nœuds. Cette mesure représente la capacité d'un nœud à se connecter rapidement avec les autres nœuds du réseau

La modularité est une mesure pour la qualité d'un partitionnement des nœuds d'un graphe, ou réseau, en communautés. Elle est principalement utilisée en analyse des réseaux sociaux. Elle a été introduite par M. E. J. Newman. C'est aussi une fonction d'optimisation pour certaines tâches de détection de communautés dans les graphes.

Le principe est qu'un bon partitionnement d'un graphe implique un nombre d'arêtes intra-communautaires important et un nombre d'arêtes intercommunautaires faible.

La modularité est décrite comme la proportion des arêtes incidentes sur une classe donnée moins la valeur qu'aurait été cette même proportion si les arêtes étaient disposées au hasard entre les nœuds du graphe. La modularité prend ses valeurs entre -1 et 1 inclus. Un graphe a une structure de communautés significative quand une partition obtient un score de modularité supérieur à 0,3. [11]

La modularité est plus élevée lorsque chaque module apparaît isolé du reste du réseau.

Cela se produit lorsque les nœuds interagissent souvent avec des nœuds dans le même module et qu'il y a peu d'interactions entre clusters.

Il existe plusieurs définitions de la modularité utilisées dans les réseaux bipartites.

On cite « la modularité de Guimerà [12] » et « la modularité de Barber [13] » qui ont été récemment revues dans le contexte des réseaux écologiques.

La modularité de Guimerà utilise des projections pondérées pour identifier des communautés distinctes au sein de chaque type de nœud. En revanche, la modularité de Barber identifie des communautés communes composées des deux types de nœuds.

5. COMMUNAUTE

5.1. DEFINITION

Une communauté est un groupe de personnes possédant et jouissant de façon indivise d'un patrimoine commun, une collectivité historique ou culturelle, un groupe ethnoculturel ou encore une société humaine organisée de manière autonome sur un territoire délimité.
Jusqu'en 1776, on parlait de communauté pour désigner un groupe socio-professionnel exerçant le même art (au sens d'artisanat) et le même commerce.
Aujourd'hui, Il existe aussi des communautés digitales qui apparaissent sur les réseaux sociaux, rassemblant des personnes qui parfois ne se connaissent même pas uniquement sur la base de leurs diverses affinités.
Face à cette accumulation de définitions, il est nécessaire de chercher les similitudes afin d'établir ce qui fait le cœur de la notion de communauté.

Il apparaît alors ce qui fait la base d'une communauté ce n'est pas un pacte ou une convention signée entre ses membres mais plutôt la simple existence d'un point commun, qu'il soit matériel ou immatériel (un patrimoine, un territoire, une culture, une origine, un savoir-faire, une cause, un goût ou un hobby…etc.)

La définition d'une communauté regroupant les divers traits évoqués serait donc la suivante : **une communauté est un groupement d'individus partageant une ou plusieurs affinités et prêts à les revendiquer comme partie intégrante de leur identité.** [14]

5.2. TYPES DE COMMUNAUTES

Il existe deux grands types de communautés, les communautés statiques et les communautés dynamiques :

5.2.1. COMMUNAUTE STATIQUE

Une communauté statique est un réseau constitué d'un ensemble de nœuds et d'un ensemble de liens reliant ces nœuds, sans prendre en considération la notion de temps, ou d'ordre, pour les caractériser. [15]

5.2.2. COMMUNAUTE DYNAMIQUE

L'évolution des communautés amène qu'il n'existe pas de définition bien précise de ce qu'est une communauté dynamique. On peut les définir comme une succession de communautés statiques [16]. Le point le plus important à propos de ces communautés est que ce sont des groupes d'utilisateurs construits à partir de la communication existant entre les utilisateurs.

Cette communication, ainsi que les intérêts des utilisateurs sont par nature des éléments changeants. C'est pourquoi au fil du temps, les communautés sont nécessairement appelées à évoluer. Pour une communauté donnée, cette tendance est reflétée par deux indicateurs :

- Les utilisateurs entrants, ceux qui rejoignent la communauté.
- Les utilisateurs sortants, ceux qui la quittent.

5.3.GENERALITE SUR LA DETECTION DE COMMUNAUTE

Depuis la publication de l'article fondateur de Girvan et Newman (en 2002), la détection de communautés ne cesse d'attirer l'attention de nombreux chercheurs du domaine. La détection de communautés consiste en un procédé commun s'appliquant à tout type de réseau, permettant le partitionnement de nœuds en communautés. L'ensemble de ces communautés doit répondre à certains critères de qualité, en particulier la modularité réseau. De nouvelles approches sont de plus en plus proposées afin de minimiser les coûts de détection en termes de temps et d'espace, tout en maximisant la qualité des partitions trouvées.
Selon les cas, de différents compromis (sur le temps, l'espace, la définition adoptée, etc.) sont permis ; ce qui se traduit par la multiplication de méthodes de détection proposées.

5.3.1. PARTITIONNEMENT DE GRAPHES

A l'origine, une version plus simple du problème de détection de communauté avait été explorée avec le partitionnement de graphe, on peut noter que parfois, partitionnement de graphe (graph clustering en anglais) et détection de communautés sont souvent utilisés indifféremment. [17]

Historiquement, le terme partitionnement de graphe se rapporte plutôt aux méthodes informées (nombre et taille des clusters connus), tandis que la détection de communautés se rapporte aux méthodes non-informées.

Le partitionnement de graphe consiste à trouver pour un graphe donné un partitionnement optimal en k partitions, k étant donné.

Pour évaluer la qualité d'un partitionnement, on prend simplement le nombre de liens situés entre les partitions, c'est à dire les liens reliant un nœud appartenant à une partition P_1 à un second nœud appartenant à une partition P_2, avec P_1 différent de P_2.

Sans autres contraintes, une solution triviale consisterait à créer une partition contenant n-(k-1) nœuds et chacune des autres partitions contiendrait un seul nœud, parmi ceux ayant le moins de liens. Cette solution satisfait efficacement le problème de la minimisation des liens inter-communautés mais ne présente évidemment que peu d'intérêt.

Dans un premier temps, une solution proposée consistait à chercher des partitions de taille donnée souvent de même taille. La méthode la plus connue est celle de Kernighan-Lin.

Cette contrainte étant trop stricte pour être efficace dans des cas réels, elle a ensuite été relâchée de manière à chercher un nombre donné de communautés, mais sans avoir à en préciser la taille exacte.

5.4. DETECTION DE COMMUNAUTES

Le partitionnement de graphe a d'importantes limites, qui ont conduit à s'intéresser à un problème lié mais plus complexe, celui de la détection de communautés. Lorsqu'on étudie des réseaux de terrain, en particulier lorsque ceux-ci sont de grande taille et/ou représentent des données complexes (réseaux sociaux, réseaux biologiques, réseaux informatiques, etc.), le nombre de groupes que l'on cherche à obtenir ne peut être connu à l'avance. Dans ces réseaux, le nombre de communautés existant est en fait, en lui-même, un résultat important.

C'est dans l'article de Girvan et Newman que le problème a pour la première fois été posé dans les termes que l'on connait maintenant. Les auteurs partent du principe que dans les réseaux petit monde réel, il existe une structure en communauté. Ils se demandent alors comment détecter automatiquement cette structure.[1]

Le nouveau problème ainsi posé peut se définir de la manière suivante : pour un réseau donné, comment le décomposer en un nombre inconnu de groupes de nœuds de manière à ce que ces groupes de nœuds correspondent à l'agencement topologique du réseau. Le problème de cette définition tient à ce que, contrairement au partitionnement de graphe, il n'y a pas de critère concret permettant d'évaluer la qualité du découpage, tel que, par exemple, le nombre de liens entre les communautés. En effet, il est très vite apparu qu'un critère aussi simple ne conduisait pas à des résultats concluants.

5.4.1. Les communautés dans les réseaux du monde réel

Dans cette section, nous présenterons quelques exemples de vrais réseaux avec une structure communautaire.

Les réseaux sociaux sont des exemples paradigmatiques de graphes avec communautés.

Le mot communauté lui-même désigne un contexte social. Les gens ont naturellement tendance à former des groupes dans leur milieu de travail, leur famille, leurs amis…

Le premier exemple (Figure I.9.1) est le réseau des membres du club de karaté de Zachary (1977), ce graphique est bien connu, régulièrement utilisé comme point de repère pour tester les algorithmes de détection de communautés.

Il se compose de 34 sommets, les membres d'un club de karaté aux États-Unis, qui ont été observés pendant une période de trois ans. Les arêtes relient les individus qui ont été observés à interagir en dehors des activités du club.

À un moment donné, un conflit entre le président du club et l'instructeur a conduit à la séparation du club en deux groupes soutenant l'instructeur et le président, respectivement (indiqué par des carrés et des cercles). La question est de savoir si de l'original structure du réseau, est-il possible d'inférer la composition des deux groupes. En effet, en regardant la Figure I.9.1, on peut distinguer deux agrégations, l'une autour des sommets 33(l'instructeur) et l'autre autour de 34 (34 est le président) .On peut aussi identifier plusieurs sommets situées entre les deux structures principales, telles que 3, 9, 10 ; de tels sommets sont souvent mal classés par les méthodes de détection de communautés.

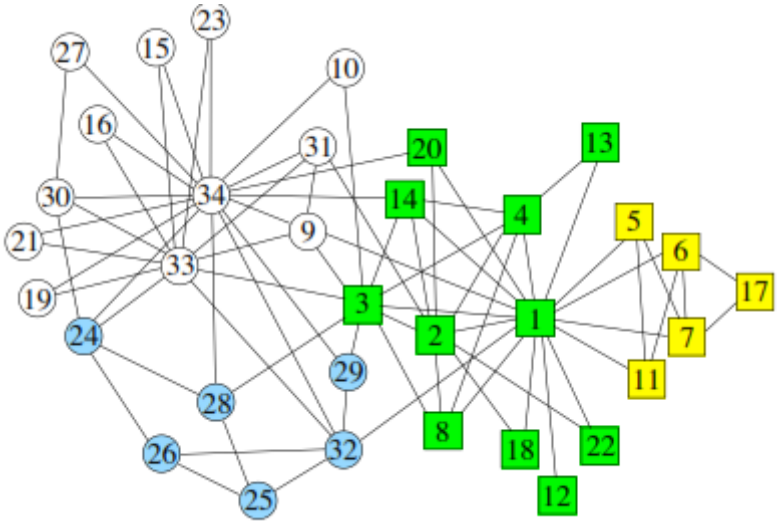


Figure I.9. Représentation graphique du Zachary Karate club [18]

Un autre exemple (Figure I.9.2) tiré de la technologie est le world wide web qui peut être considéré comme un graphique en représentant des pages web en tant que sommets et les hyperliens qui font passer les utilisateurs d'une page à l'autre comme des arêtes.

Les hyperliens sont dirigés : si on peut passer de la page A à la page B en cliquant sur un hyperlien de A, on ne trouve généralement pas sur B un lien hypertexte qui rend à A.

En fait, très peu d'hyperliens (moins de 10%) sont réciproques.

Les communautés du graphique web sont des groupes de pages ayant des convenance s typiques. Détecter les communautés dans le graphique web peut aider à identifier les grappes artistiques créées par les ensembles de liens pour améliorer la valeur du PageRank des sites web et leur accorder un classement plus élevé sur les moteurs de recherche. De cette façon on pourrait décourager cette pratique déloyale. On suppose généralement que l'existence d'un hyperlien entre deux pages implique qu'elles sont liées au contenu et que cette convenance est indépendante de la direction de l'hyperlien.

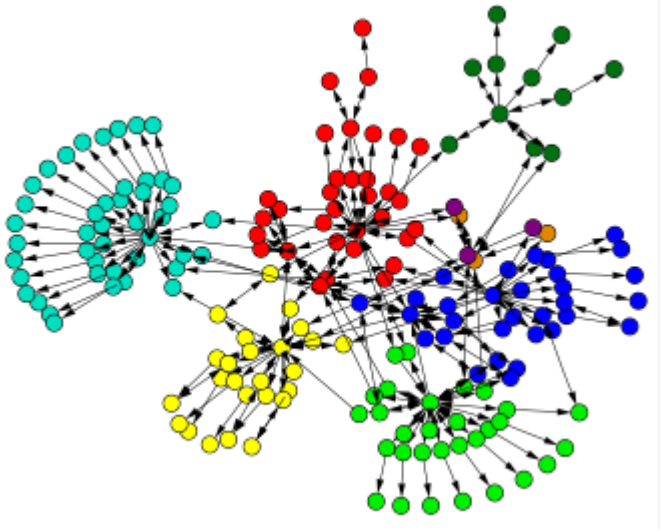


Figure I.10. Représentation graphique d'Internet

6. LES ALGORITHMES DE DETECTION DE COMMUNAUTE

En raison de la multitude d'approches proposées dans le domaine de détection de communautés, nous allons dans ce qui suit ne présenter que les approches ayant reçu le plus d'intérêt dans la communauté scientifique.

6.1. LES ALGORITHMES DE CLASSIFICATION HIERARCHIQUE

Les algorithmes de classification hiérarchique cherchent à regrouper les nœuds d'un réseau dans différentes communautés de telle sorte que les nœuds d'une même communauté se ressemblent le plus possible alors que les nœuds de communautés différentes sont les plus différentes possible. Dans ce contexte, nous distinguons deux approches distinctes à savoir les algorithmes agglomératifs et les algorithmes divisifs.

6.1.1. LES ALGORITHMES AGGLOMERATIFS

L'approche agglomérative part d'une structure dans laquelle chaque nœud du graphe représente une communauté. Nous avons initialement ***n*** communautés (où ***n*** est le nombre de nœuds).

On commence par calculer les distances entre les communautés et fusionner les deux communautés les plus proches pour former une nouvelle communauté. À chaque étape, on recalcule toutes les distances entre les communautés et on fusionne deux communautés.
Lorsqu'il n'y a qu'une seule communauté représentant le graphe entier, il n'existe plus de distance à calculer.
Les différentes étapes de ce processus peuvent être représentées par une forme arborescente appelée dendrogramme.
Les feuilles sont les communautés avec un seul nœud et la racine représente le graphe entier.

- **L'algorithme de Bondel et al. (Louvain*)***

La méthode de Louvain [Blondel et al., 2008] implante une méthode d'optimisation gloutonne locale de la modularité (la modularité étant une valeur d'échelle entre -1 et 1 qui mesure la densité des arêtes à l'intérieur des communautés à des arêtes en dehors des communautés). A l'état initial chaque nœud est affecté à une communauté différente des autres. L'algorithme applique ensuite une itération de succession de deux phases :
La 1ére phase consiste à l'affectation des nœuds : pour chaque nœud x on évalue le gain de la modularité si on le déplace dans la communauté de ses voisins directs. On déplace x dans la communauté du voisin qui maximise le gain de la modularité. Si aucun gain n'est trouvé le nœud reste dans sa communauté.
La 2éme phase consiste à la compression : on compresse le graphe obtenu en remplaçant chaque communauté par un seul nœud. Deux nœuds x et y dans le nouveau graphe sont liés par un lien s'il existe un lien entre un nœud de la communauté représentée par x et un nœud de la communauté représentée par y. [19]
Le poids de lien entre deux communautés est égal à la somme des poids des liens reliant des nœuds de deux communautés.
L'algorithme s'arrête s'il n'y a plus de possibilité de réaffectation de nœuds ou si un maximum de modularité est atteint.

- **Walktrap**

Au début de l'algorithme, on a une partition $Pi = \{\{vi\}, vi \in V, i = 1...n\}$ du graphe G qui représente n communautés. Chaque communauté contient un seul nœud. La première étape consiste à calculer toutes les distances entre les n communautés. Ensuite, on répète les étapes suivantes jusqu'à l'obtention d'une seule communauté :

- Étape 1 : sélectionner les deux communautés C1 et C2 de la partition Pk qui ont une distance minimale entre elles.
- Étape 2 : fusionner les deux communautés en une seule communauté $C3 = C1 \cap C2$ et créer un nouveau partitionnement Pk+1 = (Pk\ {C1, C2}) ∪ {C3}.
- Étape 3 : calculer les distances entre les communautés et aller à l'étape 1.

Après n − 1 itérations, l'algorithme a comme partition Pn = {V} représentant le graphe entier. Chaque fusion entre deux communautés engendre une nouvelle partition. Cet algorithme se représente à travers un dendrogramme dont P1 représente les feuilles qui se trouvent en bas et Pn la plus haute racine. [19]

6.1.2. LES ALGORITHMES DIVISIFS

Les algorithmes divisifs consistent à scinder un réseau en plusieurs communautés en éliminant itérativement les liens entre les nœuds. Ils commencent par une seule communauté (le réseau entier) en haut du dendrogramme, jusqu'à avoir n communautés à un seul nœud représentant les feuilles du dendrogramme. Dans chaque itération, tout réseau connexe est considéré comme une communauté. Les méthodes existantes se distinguent par le choix des liens à éliminer et par les poids accordés aux liens. Nous retenons l'algorithme Edge Betweenness comme exemple d'algorithme divisif.

- **Edge Betweenness**

Les méthodes de détection de communautés s'intéressent généralement aux nœuds du réseau. Newman s'est penché sur les liens plutôt que sur les nœuds [20]. En effet, l'identification des liens se trouvant entre les communautés, ainsi que leur élimination permet d'identifier les différentes communautés dans un réseau. Afin de trouver les liens inter-communautés, l'algorithme *Edge Betweenness* accorde à chaque lien une mesure de centralité d'intermédiarité qui se base sur le calcul du plus court chemin.

Le calcul de cette mesure repose sur un algorithme de calcul d'intermédiarité des liens qui est composé de deux principales phases.

La première phase consiste à affecter pour chaque nœud un poids, tandis que la seconde calcule le degré d'intermédiarité des liens.

L'approche de l'Edge Betweeness se base sur le principe qu'un lien se trouvant entre les communautés fait partie d'un très grand nombre de chemins géodésiques entre les différents nœuds. Il a de ce fait une grande valeur de centralité d'intermédiarité. Cependant, quelques liens se trouvant entre les communautés et qui sont plus longs que les chemins géodésiques, ont

des ***EBC*** (centralité d'intermédiarité d'un lien) faibles. D'où l'idée d'éliminer, dans chaque itération, le lien le plus intermédiaire (i.e., ayant la plus grande valeur ***EBC***), puis recalculer à nouveau les ***EBC*** de tous les liens restants.

L'algorithme est constitué de quatre étapes principales :

1- Calculer les ***EBC*** pour tous les liens du graphe.

2- Éliminer le lien le plus intermédiaire.

3- Recalculer les ***EBC*** de tous les liens restants.

4- Répéter les étapes 2 et 3 jusqu'à l'élimination de tous les liens.

Ce processus se répète jusqu'à l'obtention des feuilles du dendrogramme qui sont constituées par ***n*** communautés. Chacune de ces communautés contient donc un seul nœud.

6.1.3. LES ALGORITHMES D'OPTIMISATION D'UNE FONCTION OBJECTIVE

Il existe un certain nombre d'algorithmes se basant sur des heuristiques pour définir la structure de communauté des réseaux. Ce type d'algorithme consiste à définir une fonction objective dont la valeur varie selon les communautés dégagées. La fonction est maximale pour la meilleure structure de communauté. Un exemple de ce type d'algorithme est Fast Greedy de Newman.

- **Fast Greedy**

Les sommets sont regroupés itérativement en partant d'une partition où chaque sommet représente une communauté jusqu'à obtenir une seule communauté regroupant tous les sommets. L'algorithme fusionne à chaque étape la paire de communautés qui permet d'avoir la plus grande augmentation de la modularité. La modularité est une fonction de mesure de la qualité d'une partition basée sur la proportion d'arêtes internes à chaque communauté. [21]

6.1.4. LES ALGORITHMES A BASE DE MODELE

Les algorithmes à base de modèle sont des algorithmes de classification non supervisée utilisant des méthodes à base de prototypes exprimés dans un formalisme de modèles. Pour chaque type de données, un modèle d'apprentissage adapté à la nature des données traitées est proposé.

Pour le traitement des graphes, nous détaillons l'algorithme Label Propagation comme algorithme à base de modèle.

- **Label Propagation**

Le premier algorithme qui implanta l'idée de propagation de labels est proposé par Raghavan et al. en 2007. C'est un algorithme itératif où à chaque itération un nœud passe son label à ses voisins directs, et reçoit ceux de ses voisins. Chaque nœud détermine le label majoritaire qu'il

adopte pour l'itération suivante. Ce processus itératif mène à un consensus sur un label précis pour chaque groupe de nœuds. La propagation de labels peut se faire en mode synchrone ou asynchrone. L'avantage du premier mode est qu'il peut facilement passer à l'échelle vu le parallélisme du calcul. Par contre, il peut y avoir un problème de convergence lié à un échange infini de label entre deux nœuds. Ce problème a été évité dans le mode asynchrone. [21]

6.1.5. ALGORITHMES UTILISANT LES PROPRIETES SPECTRALES DES RESEAUX

Les algorithmes spectraux utilisent les représentations matricielles des réseaux pour détecter les communautés. La décomposition en vecteurs propres et valeurs propres est exploitée de diverses façons afin d'aboutir à une partition adaptée.

L'algorithme de Newman (Leading Eigenvector) est inspiré par les méthodes traditionnelles de partitionnement spectral. Ces algorithmes visent à trouver la coupe du graphe minimisant le « nombre » d'arêtes coupées entre les différentes parties. Minimiser ce coût de coupe revient à calculer le vecteur propre correspondant à la plus petite valeur propre non nulle de la matrice Laplacienne du graphe. Le graphe est alors séparé en deux parties en fonction du signe de leur composante selon ce vecteur propre. Cette méthode est particulièrement adaptée lorsque les partitions sont de taille comparable. Ceci n'est ni approprié ni réaliste pour des problèmes de détection de communautés. Afin d'adapter cette technique à la détection de communautés, Newman a proposé de substituer la maximisation de la modularité à la minimisation du coût de coupe. Il a montré que le problème peut s'exprimer à partir de la matrice de modularité.

Son rôle est identique à celui joué par la matrice Laplacienne pour la minimisation du coût de coupe. La partition obtenue permet de trouver simplement deux communautés. L'algorithme utilise ensuite une approche hiérarchique divisive en n'acceptant pas les divisions qui ne permettent pas un gain de la modularité. [22]

6.2. LES INCONVENIENTS DES APPROCHES EN INTERACTION UTILISATEURS

6.2.1. L'ignorance du type d'interaction

Deux utilisateurs partageant le même intérêt seront représentés par deux nœuds liés par une arrête, cette représentation étant claire ne fournit aucune information sur l'intérêt partagé.

Un autre défaut est la non prise en considération du nombre d'intérêts partagés étant donné que la liaison entre deux utilisateurs ne sera représentée que par une seule arrête.

De ce fait, les nœuds composant les communautés générées auront une forte interaction, mais les objets et les sujets de ses interactions seront inconnus.

6.2.2. Approches centrés sur les utilisateurs

La plupart des approches de détection de communauté se base sur la représentation d'utilisateurs seulement.

Il n'existe pas de base de données pour les utilisateurs qui connaissent un accroissement dans leurs nombres et leurs informations suite à l'explosion des réseaux sociaux, ce qui rend difficile de connaitre au préalable la communauté de tous les utilisateurs à analyser.
Cette méthode est donc inappropriée aux réseaux complexes actuels.

6.2.3. Application très longue dans les réseaux complexes

Les algorithmes disponibles deviennent incapables de traiter un réseau complexe où le nombre d'utilisateurs augmente considérablement.

6.3. APPROCHE DE DECOUVERTE DE COMMUNAUTE BASEE SERVICE

Les inconvénients cités ci-dessus ont poussé à trouver une nouvelle approche de détection de communauté basée sur l'interaction entre les utilisateurs et les services web tel que les communautés générées seront composées des utilisateurs ayant accédé aux mêmes services.

Cette nouvelle approche est basée sur 2 processus :

- **Processus d'initialisation :** ce premier processus consiste à récupérer les utilisateurs et les services et les interactions entre eux.

  A la fin de ce processus, un graphe biparti sera généré constitué de deux groupes de nœuds (utilisateurs et services) et un lien établi signifie qui il y a une interaction entre ses extrémités (utilisateur/service)

- **Processus de découverte de communauté :** cette procédure consiste à générer des communautés en regroupant les utilisateurs ayant atteint les mêmes services.

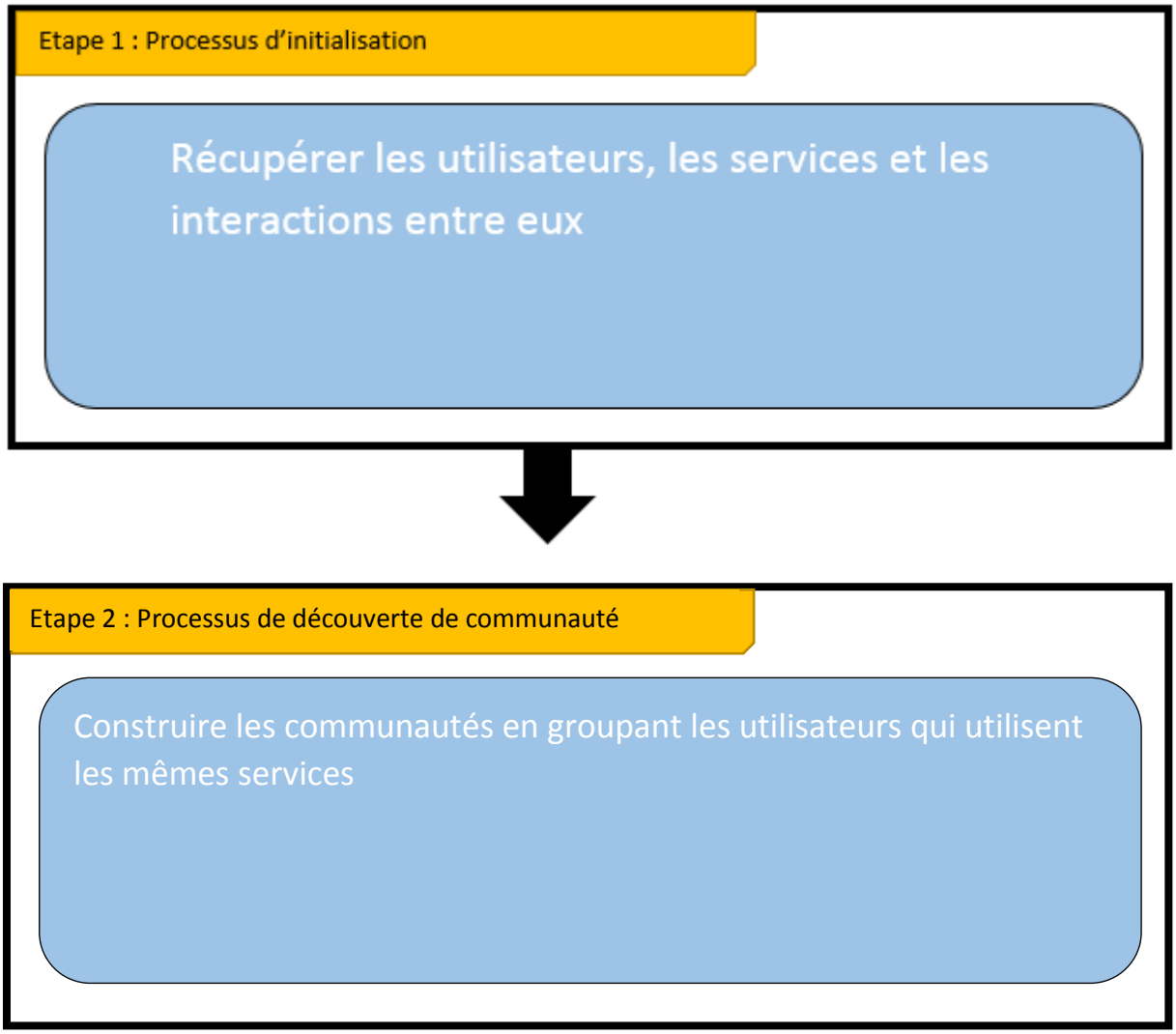


Figure I.11. Les processus constituant l'approche de découverte de communautés basée service

Cependant, cette approche reste toujours limitée surtout dans les réseaux complexes et ne donne pas une signification sur les communautés détectées, d'autant plus que le monde réelle comporte des dizaines de milliard d'utilisateurs sur plusieurs domaines et sujets d'interaction.

7. CONCLUSION

Dans ce chapitre, nous avons proposé un bref aperçu sur les concepts essentiels de notre étude. Le premier volet de ce chapitre a été consacré à l'introduction de certains concepts de base indispensables sur les services web, les réseaux sociaux et les bases de la théorie de graphes afin de décrire avec aisance les notions utilisées dans les différents travaux rencontrés et l'approche que nous proposons. Nous nous sommes particulièrement penchés dans le deuxième volet de ce chapitre sur les différentes définitions relatives à la notion de communautés et les nombreux algorithmes utilisés dans la détection de communautés.

Nous avons détaillé dans le dernier volet de ce chapitre, les inconvénients de l'approche de détection de communauté basée sur l'interaction des utilisateurs et nous avons fait un bref tour d'horizon sur la nouvelle approche basée sur les services web.

Dans le chapitre suivant nous présenterons une amélioration de l'approche de détection de communauté basée service.

II

Analyse et Conception

1. INTRODUCTION

L'objectif de la détection de communautés dans les réseaux sociaux ou plus précisément dans les graphes est de créer une partition des sommets en tenant compte des relations qui existent entre les sommets dans le graphe de telle sorte que les communautés soient composées de sommets fortement connectés et qu'elles soient peu reliées entre elles.

Dans le chapitre précédant, nous avons exposé les principales méthodes de détection de communautés proposées dans la littérature, mais nous avons constaté que toutes ces méthodes sont principalement concentrées sur les relations utilisateur-utilisateur seulement et ce qui a conduit à l'ignorance de toute autre interférence et ainsi des informations relatives aux services web qui pourraient améliorer le processus de découverte de communautés.

Dans le chapitre qui suit, nous suggérons une solution à la problématique précédente en présentant une amélioration de l'approche de découverte de communautés qui se base sur les interactions utilisateur-service.

2. IDEE GENERALE

Les graphes bipartites sont la représentation des interactions entre deux classes distinctes de nœuds de sorte que les nœuds ne peuvent qu'interagir avec les nœuds de l'autre classe [Chapitre I section 4.3], ces graphes peuvent être utilisés par exemple pour représenter la manière dont certains acteurs sont liés à certains événements dans les réseaux sociaux [26]; Pour représenter les réseaux commerciaux industriels [27]; Et en écologie pour représenter les interactions entre les espèces végétales et les espèces pollinisatrices [28]…etc. L'identification de la structure au sein des réseaux est utile pour expliquer leur formation, leur fonction et leur comportement et représente un défi important dans un ensemble diversifié de disciplines.

Les algorithmes de détection communautaire [Chapitre I section 6] sont conçus pour identifier des clusters ou des modules de nœuds dans un réseau qui sont plus susceptibles d'interagir entre eux qu'avec le reste du réseau. La modularité est une évaluation de la façon dont les nœuds sont divisés en sous-ensembles séparés formant des modules. Cela est fait en évaluant les mesures dans laquelle les interactions dans le réseau se produisent dans les modules plutôt qu'entre les modules.

Ceci dit, pour notre approche, on utilise un graphe biparti utilisateur-service défini comme étant un graphe non-orienté G= (V, E) où V et E représentent des ensembles finis de nœuds et d'arrêtes respectivement. L'ensemble V est une union de deux sous-ensembles disjoints V = V1 ∪ V2 où V1 représente l'ensemble des utilisateurs et V2 l'ensemble des services.

Pour les arrêtes, elles sont étiquetées par des informations concernant l'interaction entre ces extrémités, cela sera présenté dans le reste du chapitre.

La figure II.1 est un exemple du graphe biparti cité ci-dessus :

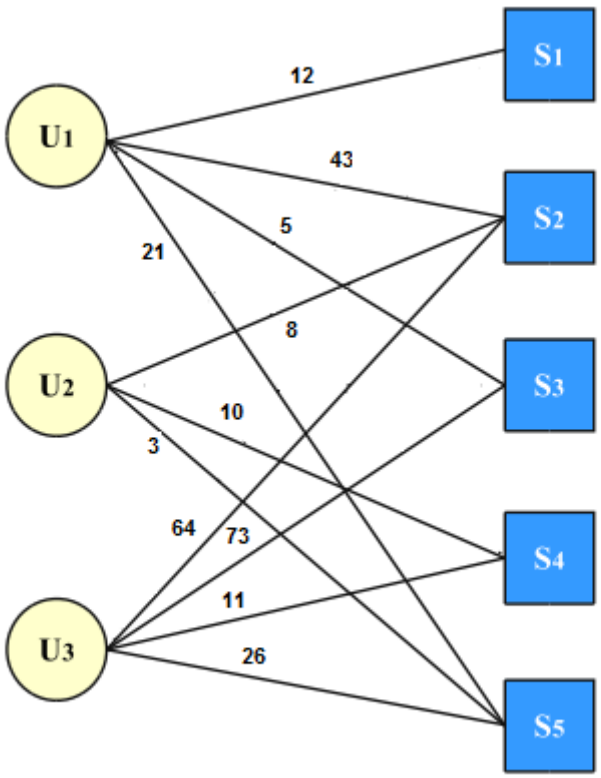


Figure II.1. Exemple de graphe biparti étiqueté

On utilisera le terme profil au lieu de nœud pour une meilleure présentation de l'idée. Ainsi un profil utilisateur sera lié à un profil service s'il y a une interaction entre ces deux profils.

3. METHODOLOGIE

La méthode utilisée repose sur les interactions des utilisateurs avec les services web.

Notre méthode est composée des 4 processus successeurs :

- **Processus de création :** A cause de l'indisponibilité d'une base de donnée réelle (benchmark), on était obligé de passer par ce premier processus, celui de la création des utilisateurs et les services
- **Processus de calcul de convenance :** Cette partie est consacrée à l'établissement des liens entre chaque deux nœud des deux groupes tel que le lien est étiqueté par la valeur de convenance calculée entre un internaute et un service
- **Processus de génération des accès :** en utilisant les résultats obtenus dans le processus précédant, des liens entre les deux groupes de nœuds seront établis, tel qu'un lien entre un utilisateur et un service est implanté si l'utilisateur a accédé à ce service.
- **Processus de découverte de communauté :** cette dernière procédure, qui est la plus importante, consiste à générer des communautés grâce aux deux processus précédant en regroupant les utilisateurs ayant atteint les mêmes services par une certaine convenance et selon certains critères, tous spécifiés par le décideur.

L'organigramme suivant schématise les processus décrits ci-dessus :

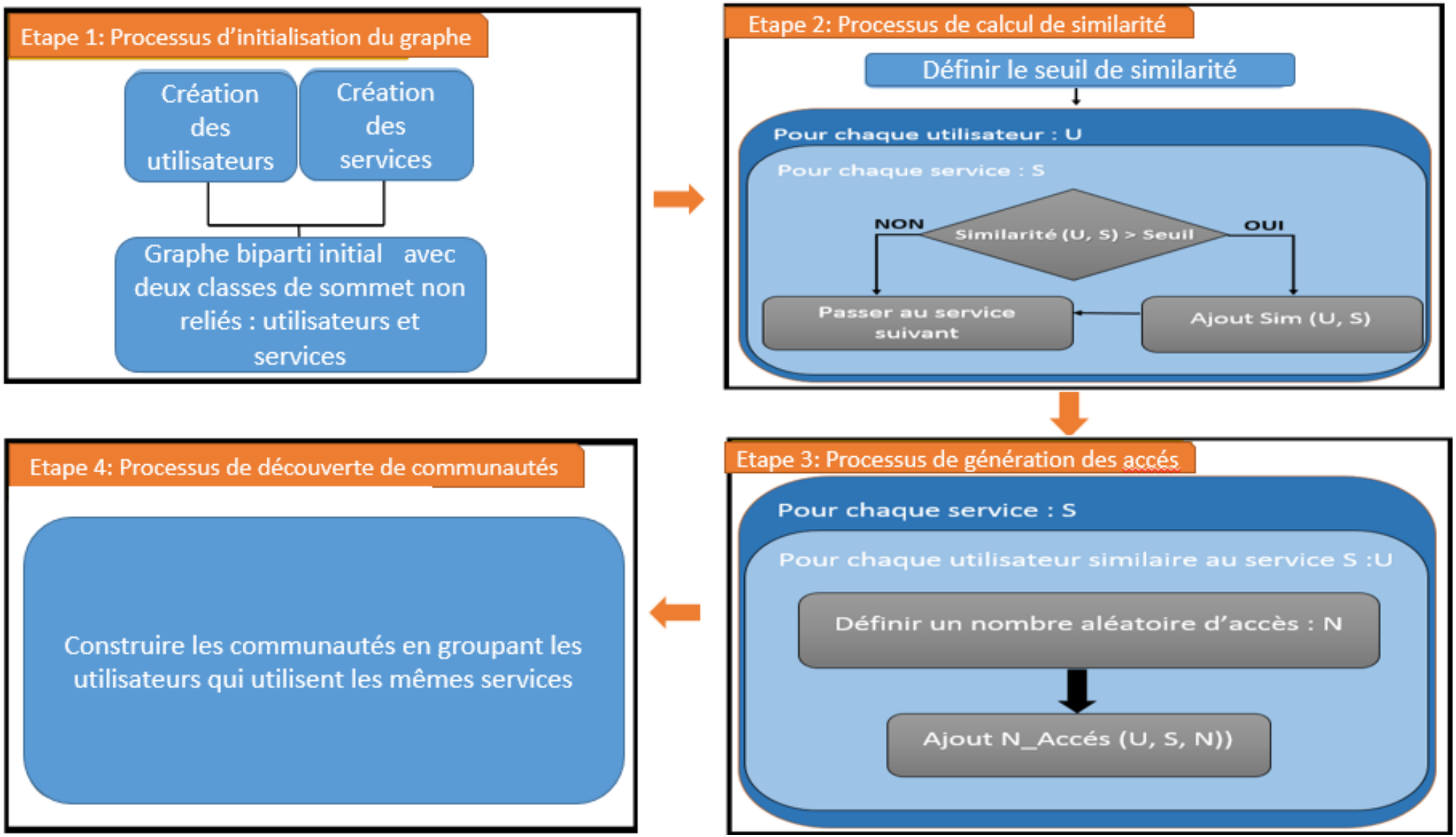


Figure II.2. Les processus constituant l'approche de découverte de communautés basée service

3.1. PROCESSUS D'INITIALISATION DU GRAPHE

Le développement des profils des utilisateurs est central pour des mécanismes tels que la recommandation ou la personnalisation d'informations qui correspondent aux besoins spécifiques de l'utilisateur. Dans un système d'information, un profil d'utilisateur est habituellement construit et enrichi de façon itérative à partir du comportement de son propriétaire.

Dans la plupart des approches de découverte de communautés, le profil utilisateur est aussi défini par des information trouvées dans les systèmes informatiques, ainsi par des intérêts pondérés dans un ou plusieurs domaines (par exemple, culture, sports) (Gauch et al., 2007), ces intérêts peuvent également varier en fonction des informations contextuelles (par exemple, heure, lieu) (Tchuente et al.2010).

A la différence des algorithmes classiques qui ne prennent en considération que les profils utilisateur, on s'intéressera dans cette thèse à la découverte de communautés en prenant compte des profils des utilisateurs et leurs interactions avec des services, tel que le profil de ces derniers sont caractérisés par un nombre d'attributs qui seront présentés dans la section suivante.

La figure II.3 schématise les profils en forme d'arbre :

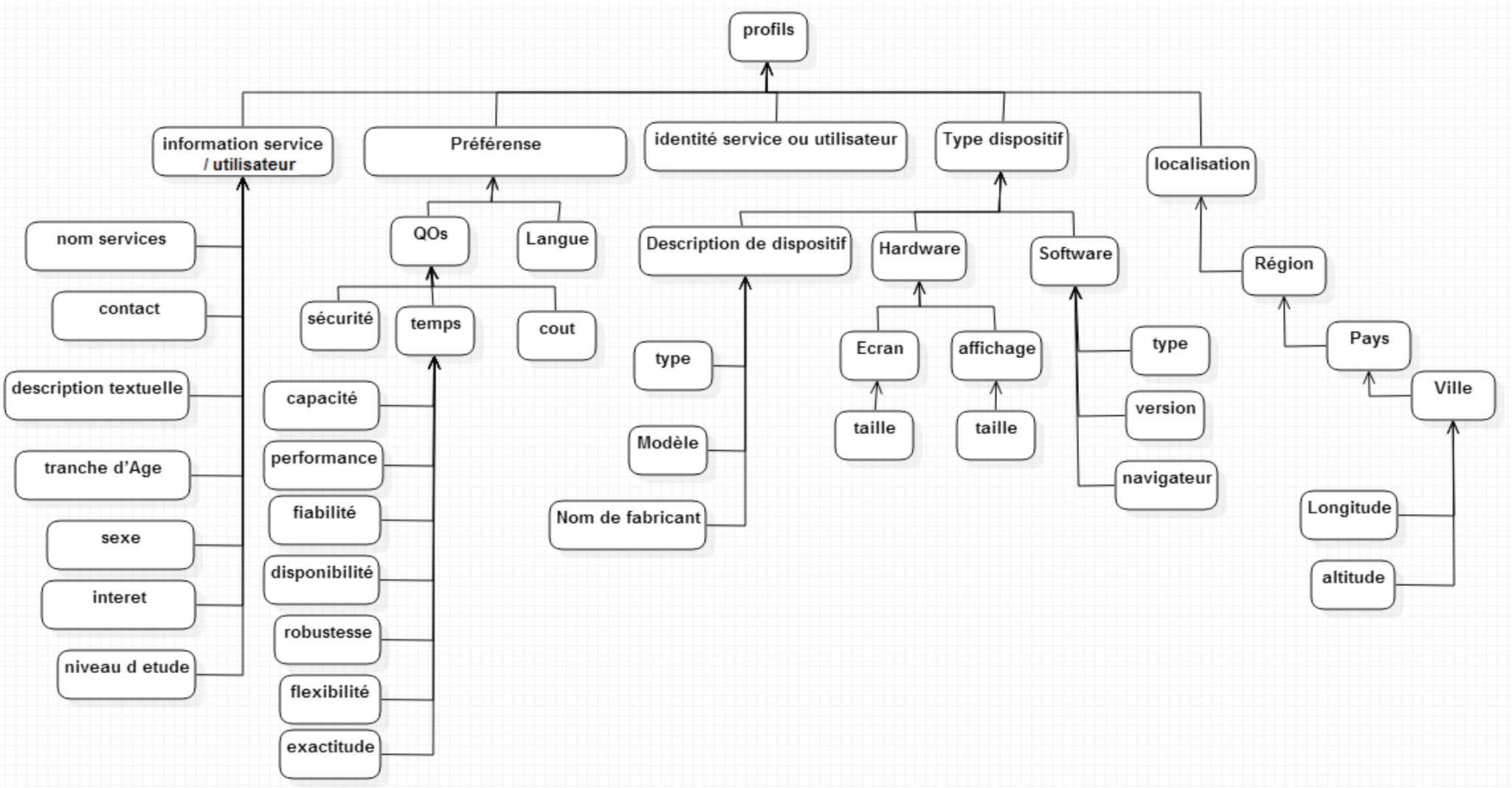


Figure II.3. Spécification des profils utilisateur et service

Comme elle est citée dans la section précédente, un utilisateur possède un profil qui représente son identité qui est composé de plusieurs informations tel que le nom, prénom, date de naissance, son adresse (son adresse géographique), préférence, centre d'intérêts… etc, mais le processus de découverte de communautés ne se base que sur un sous-ensemble de ces informations.

Cependant, un profil service est composé d'informations tel que la qualité de service, le coût d'utilisation de service, les restrictions géographiques couvertes par ce service …etc.

3.1.1. DESCRIPTION DU CONTEXTE UTILISATEUR ET SERVICE

Dans ce qui suit, les paramètres représentant les contextes schématisés précédemment seront détaillés :

3.1.1.1.Identité service ou utilisateur :

Cet attribut spécifie pour chaque profil service ou utilisateur un identifiant unique.

3.1.1.2. Information service ou utilisateurs

Ce composant spécifie les informations suivantes :

- **Nom service ou utilisateur :** Pour le service, il correspond au nom du service offert. Pour l'utilisateur, cet attribut réfère à son propre nom
- **Contact :** Il réfère pour le service la boite de réception (e-mail) de son fournisseur. Pour l'utilisateur, cet attribut réfère à sa boite de réception (e-mail)
- **Description textuelle :** Du point de vue service, il décrit les fonctionnalités qu'offre celui-ci. Pour l'utilisateur, cet attribut fournit une brève description de ce que doit réaliser le service désiré
- **Age :** pour le service, cet attribut décrit la tranche d'âge (mineur, jeune, vieux, tous les âges) de gens qui peuvent l'utiliser, pour l'utilisateur cet attribut est calculable depuis sa date de naissance.
- **Sexe :** pour le service, cet attribut désigne la catégorie sexuelle pour laquelle ce service est dédié, cependant pour l'utilisateur, il désigne son sexe.
- **Intérêts :** cet attribut détermine sur quel intérêt, le service se concentre.
  Pour l'utilisateur, il représente une collection de ses intérêts.
- **Niveau d'étude :** Il symbolise le niveau scolaire exigé pour l'accès au service, et coté utilisateur, il représente son propre niveau scolaire (primaire, moyen, secondaire … etc)

- **Religion :** dénote pour le service, à quel communauté religieuse (ou pour tout le monde), sont destinés ses fonctionnalités, et coté utilisateur, il dénote la religion de ce dernier.

3.1.1.3.Préférence

Ce paramètre englobe deux aspects :

3.1.1.3.1. Les paramètres de qualités de service (QoS)

Pour l'utilisateur, ces paramètres représentent la qualité de service web recherchée par contre, pour le service, ils représentent la qualité de service web fournie. On distingue :
Les attributs de **QoS** liés au temps d'exécution

3.1.1.3.1.1.Le temps

- Capacité (Availability) : c'est le nombre maximum de requêtes simultanées que le système prend en charge avec des performances garanties

- **Performance (Response Time) :** c'est la vitesse d'exécution d'une requête ou le temps pris pour envoyer une requête et recevoir une réponse
- **Fiabilité (Throughput) :** c'est la capacité d'un service à exécuter ses fonctions requises dans des conditions indiquées pendant une période donnée
- **Disponibilité (Validity) :** c'est la probabilité que le système soit actif
- **Robustesse /Flexibilité (Successability) :** elle représente le taux pour qu'un service peut s'exécuter correctement dans le cas de données d'entrées inadmissibles, incomplètes ou contradictions

- **Exactitude (Reliability) :** il représente le taux d'erreur produit par le service

3.1.1.3.1.2. Sécurité

Représente le niveau de sécurité sur une échelle de 1 à 10, 10 étant la sécurité maximale. Pour le service elle désigne son niveau de sécurité et pour l'utilisateur le degré de sécurité du service web désiré.

3.1.1.4.La langue

Pour l'utilisateur ce paramètre réfère aux langues maitrisées par ce dernier et pour le service, il représente les langues avec lesquelles le service peut être fourni.

3.1.1.5.Type de dispositif

Ce paramètre procure un vocabulaire commun aux fournisseurs de services pour décrire les différents types de dispositifs pouvant supporter les données transmises par leurs services web et aux utilisateurs afin de décrire les contraintes matérielles et logiciel du terminal avec lequel ils accèdent aux différents services web.

Ce paramètre englobe trois aspectes :

- La description du dispositif

Type (Laptop, tablette, smartphone …etc)

Modèle (Nom commerciale du fabriquant)

-Le hardware

Résolution (pouces)

-Le software : Ce paramètre définit le type et la version du navigateur utilisés :

Version

Navigateur

3.1.1.6. Localisation

Ce composant permet à l'utilisateur de décrire sa localisation (latitude et longitude) et coté service, l'adresse dont il est fournie.

Il est spécifié par une « Région » qui est composé de plusieurs « Pays » ou par coordonnées « Longitude et Latitude ».

3.2. PROCESSUS DE CALCUL DE CONVENANCE

La mesure de convenance est le processus de calcul de correspondance entre un profil utilisateur et un profil service pour construire le graphe de convenance .
Le processus consiste à comparer les champs (attributs) des deux contextes cités auparavant.

Etant donné que les paramètres sont de deux types différents, par exemple le calcul de convenance pour les champs numérique n'est pas applicable sur les champs de chaine de caractères, ce qui nous a conduits à utiliser de différentes mesures de convenance pour les attributs de différents types.

Aussi un poids est affecté à chaque attribut afin de valoriser son intervention dans le calcul de convenance et ceci d'après la nature de l'étude.

3.2.1. CALCUL DE LA CONVENANCE GLOBALE

Afin de calculer la convenance globale, nous avons utilisé la formule suivante :

$$GSim(x,y) = 1/N * \Sigma Sima(xi, yi) * coeffi \quad (1)$$

Chaque profil est modélisé par un tableau d'enregistrement contenant la position de l'attribut dans l'arbre et sa valeur. Alors beaucoup de descriptifs ont des valeurs manquantes que ce soit pour un profil utilisateur ou service, il faut une mesure de convenance qui prend en charge ce fait. Donc, la mesure de convenance quantitative (SIMA) [25] répond parfaitement à ce problème.

Avec :

- x, y représente respectivement un utilisateur et un service.
- coeffi désigne le coefficient affecté à l'attribut i
- $GSim(x,y) \in [0,1]$
- $N = \Sigma$ coeffi
- La convenance atomique (SIMA) est une formule qui retourne une valeur dans l'intervalle [0,1] , tel que :

$$Sima(x_i, y_i) = \begin{cases} \text{SIMA : R*R} \rightarrow [0,1] \text{ ou [CC]*[CC]} \rightarrow [0,1] \\ \text{Min } (x_i, y_i) \text{ / Max } (x_i, y_i) \\ \text{Sinon} \\ \text{String_Similarity } (x_i, y_i, \text{string_metric}) \end{cases} \quad (2)$$

- CC : Chaine de caractères
- Similarity (x_i, y_i) désigne l'existence de convenance de l'attribut i, utilisée pour le calcul des champs numérique

- String_similarity (x_i, y_i, string_metric) : cette fonction est dédiée pour le calcul de convenance des champs non numérique (les chaines de caractère), elle prend en paramètres deux chaine de caractère et l'une des métriques suivante :

  - ✓ Levenshtein Distance
  - ✓ Jaro-Winkler Distance
  - ✓ Sorensen-Dice Coefficient

Ces métriques seront représentées prochainement (section 3.2.6)

3.2.2. CONVENANCE DES CHAMPS DE TYPE NUMERIQUE

Les champs numériques qui sont intégrés dans le calcul de la convenance sont : les attributs concernant le QOS (temps et sécurité) et l'âge.

3.2.2.1. Calcul de convenance basé âge :

La plus part des applications sont dédiées à des groupe d'utilisateurs et ces dernières en ligne forment des communautés différentes selon ces critères. Les groupes les plus forts sont ceux qui sont constitués de plus de jeunes, les enfants et les personnes ayant plus de 30 ans sont moindres. Ce critère joue un rôle très important pour une communauté qui existe réellement, celle-ci peut avoir des utilisateurs de différent âge. Beaucoup de services web ciblent des tranches d'âge spécifiques ce qui nous entraine à diviser les utilisateurs en trois catégories :

- Mineur < 18 ans
- 18 <= Jeune < 40 ans
- Vieux >= 40 ans

Plus une autre catégorie 'All' qui rassemble toute les catégories précédentes pour les services web qui ne précisent pas d'âge spécifique pour les utilisateurs pour accéder à leur fonctionnalités.

La fonction de convenance atomique pour l'attribut âge est égale à 1 si la tranche d'âge du service est 'All' ou bien si l'âge d'utilisateur est inclus dans le tronçon d'âge du service, sinon 0.

3.2.2.2. Calcul de convenance basé QoS

La qualité du service (QoS) fait référence à la capacité d'un réseau à atteindre une bande passante maximale et à traiter d'autres éléments de performance du réseau tels que la latence, le taux d'erreur, le temps de disponibilité.

Également, Elle implique le contrôle et la gestion des ressources réseau.

Dans notre cas les paramètres de qualité de service utilisés sont présentés dans la section 3.2.1.1.3.

En ce qui concerne la sécurité, la valeur de convenance est égale à la valeur minimale divisé par la valeur maximale de l'attribut des profils utilisateur et service.

Les paramètres de qualité de service relative au temps sont des valeurs réelles qui sont inférieur à 100 pour les attributs «la performance, la fiabilité, la disponibilité, la robustesse et l'exactitude » et inférieur à 10000 pour l'attribut « la capacité » qui est de type 'entier'.

Afin de calculer la convenance des paramètres précédemment cités, on s'est basé sur le calcul du minimum sur le maximum des deux valeurs.

3.2.3. CONVENANCE DES CHAMPS A VALEUR ENUMERATIVE

Un champ à valeur énumérative prend une valeur parmi une collection définie, par exemple le champ sexe prend une valeur dans la collection : [Masculin, Féminin].

3.2.3.1. Calcul de convenance basée sexe

Selon la désignation de l'attribut sexe, un service est dédié soit aux hommes, aux femmes ou aux deux sexes.

Ainsi, la convenance est égale à 1 si le sexe de l'utilisateur est le même que celui du service ou si le service est dédié aux deux sexes, et 0 dans le cas échéant.

3.2.3.2. Calcul de convenance basée niveau scolaire

La plupart des services web sont dédié pour tous les utilisateurs d'internet, cependant le niveau scolaire diffère d'un utilisateur à un autre, ce qui rend ce critère important dans le processus de découverte de communauté, par exemple, les communautés dédié aux étudiants, aux élèves …etc.

Le paramètre niveau scolaire est une liste qui contient des chaines de caractères qui représente les différents niveaux d'études organisées en ordre croissant par niveau, plus un autre attribut qui désigne les services web qui sont accessible à tous les utilisateurs du web.

Ainsi, pour le calcul de convenance , on compare le niveau scolaire de l'utilisateur à celui du service (le service est dédié à un seul niveau scolaire), si ce dernier est inférieur au premier ou si le service n'exige pas de niveau scolaire alors la convenance est égale à 1, sinon 0 dans le cas opposé.

3.2.3.3. Calcul de convenance basée religion

De nombreux service web sont destinés à une catégorie religieuse précise et d'autres ne se préoccupe pas de la religion des utilisateurs.

Ceci dit, la convenance selon la religion est égale à 1 si l'utilisateur et le service sont de même religion, ou que le service n'exige pas de religion précise, et 0 s'ils sont de religions différentes.

3.2.4. CONVENANCE DES CHAMPS A MULTI-VALEURS :

Ces champs dont on va parler, prennent plusieurs valeurs depuis une collection définie.

3.2.4.1. Calcul de convenance basée 'Langue' et 'Intérêts'

Ces deux paramètres sont sous forme de liste car l'utilisateur peut maitriser plusieurs langues et le service peut être fourni sous différentes langues mais ne peut se concentrer que sur un seul intérêt contrairement à l'utilisateur qui a plusieurs intérêts ciblés.

Le calcul de convenance est basé sur l'intersection d'ensembles :

Si l'intersection entre deux ensembles (coté utilisateur, coté service) du même attribut n'est pas vide alors la convenance est égale à 1, sinon elle est égale à 0 si le résultat est un ensemble vide.

3.2.5. CALCUL DE CONVENANCE BASEE CONTACT

Le contact représente une adresse de courrier électronique qui est constitué des trois éléments suivants :

- une partie locale, identifiant généralement une personne ou un service
- le caractère séparateur @ (arobase),

- l'adresse du service, généralement un nom de domaine identifiant l'entreprise hébergeant la boîte électronique.

La convenance est égale à 1 si la partie Adresse du service et de l'utilisateur sont semblables, 0 si elles sont différentes.

3.2.6 CONVENANCE DES CHAMPS DE TYPE CHAINE DE CARACTERES

Le paramètre qui dépend de cette convenance est « Description textuelle » appartenant à la branche « Information service ou utilisateurs ».

Cette convenance est calculée avec l'une des distances suivantes :

3.2.6.1. Distance de Levenshtein

La distance de Levenshtein est une distance mathématique donnant une mesure de la convenance entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.

Elle a été proposée par Vladimir Levenshtein en 1965. Elle est également connue sous les noms de distance d'édition ou de déformation dynamique temporelle, notamment en reconnaissance de formes et particulièrement en reconnaissance vocale.

On appelle distance de Levenshtein entre deux mots M et P le coût minimal pour aller de M à P en effectuant les opérations élémentaires suivantes :

- substitution d'un caractère de M en un caractère de P ;
- ajout dans M d'un caractère de P ;
- suppression d'un caractère de M.

On associe ainsi à chacune de ces opérations un coût. Le coût est toujours égal à 1, sauf dans le cas d'une substitution de caractères identiques.

3.2.6.2. Distance de Jaro-Winkler

La distance de Jaro-Winkler mesure la convenance entre deux chaînes de caractères.

Il s'agit d'une variante proposée en 1999 par William E. Winkler, découlant de la distance de Jaro (1989, Matthew A. Jaro) qui est principalement utilisée dans la détection de doublons.

Le résultat est normalisé de façon à avoir une mesure entre 0 et 1, donc zéro représente l'absence de convenance et 1, l'égalité des chaines comparées.

Cette mesure est particulièrement adaptée au traitement de chaînes courtes comme des noms ou des mots de passe.

La distance de Jaro entre chaînes s1et s2 est définie par :

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)$$

Où :

- $|si|$ est la longueur de la chaîne de caractères ;
- m est le nombre de caractères correspondants (voir ci-dessous) ;
- t est le nombre de transpositions (voir ci-dessous).

Deux caractères identiques de *s1* et de *s2* sont considérés comme correspondants si leur éloignement (i.e. la différence entre leurs positions dans leurs chaînes respectives) ne dépasse pas :

$$\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1.$$

Le nombre de transpositions est obtenu en comparant l'i-ème caractère correspondant de *s1* avec l'i-ème caractère correspondant de *s2*.

Le nombre de fois où ces caractères sont différents, divisé par deux, donne le nombre de transpositions.

3.2.6.3. Distance de Sorensen-Dice

L'indice de Sørensen-Dice, connu aussi sous les noms d'indice de Sørensen, coefficient de Dice et d'autres noms encore est un indicateur statistique qui mesure la convenance de deux échantillons. Il a été développé indépendamment par les botanistes Thorvald Sørensen et Lee Raymond Dice dans des articles publiés en 1948 et 1945 respectivement.

Pour des ensembles finis quelconques X et Y, l'indice s'exprime par :

$$s = \frac{2|X \cap Y|}{|X| + |Y|}.$$

Ici, |X| est le nombre d'éléments de X. L'indice peut varier de 0 (quand X et Y sont disjoints) à 1 (quand X et Y sont égaux). Comme pour l'indice de Jaccard, les opérations ensemblistes peuvent être exprimées en termes d'opérations vectorielles sur des vecteurs binaires X and Y :

$$s_v = \frac{2|X \cdot Y|}{|X| + |Y|}.$$

Dans le domaine de la recherche d'informations, le coefficient peut être vu comme le double de l'information partagée, rapportée à la somme des cardinalités. Le coefficient peut aussi être utilisé comme une mesure de convenance entre chaînes de caractères. Étant donnés deux chaînes x et y, on peut calculer le coefficient comme suit :

$$s = \frac{2n_t}{n_x + n_y}$$

où *nt* est le nombre de digrammes (formés de deux caractères consécutifs) communs aux deux chaînes, *nx* est le nombre de digrammes dans x et *ny* le nombre de digrammes dans *y*.

3.3. PROCESSUS DE GENERATION DES ACCES

Après avoir terminé le calcul de convenance des attributs précédents, chaque valeur résultante sera multipliée par un certain coefficient défini par l'utilisateur afin de calculer la convenance globale en sommant les valeurs obtenues.

Le processus de génération des accès est basé sur le calcul de convenance . Le décideur est invité en premier lieu à définir une valeur minimale de convenance , ainsi que les valeurs des paramètres introduits dans la section précédente.

L'étape suivante consiste à parcourir tous les utilisateurs et tous les services et calculer la convenance entre chacun d'eux. Des triplets contenant le nœud utilisateur, le nœud service et la valeur de convenance entre eux seront formés.

Les triplets ayant la valeur de convenance inférieure à la convenance minimale définie seront rejetés et à la base des triplets restants (ayant la valeur de convenance supérieure ou égale à la convenance minimale), un nombre aléatoire d'accès sera généré entre les deux nœuds du triplet dans un intervalle entre 1 et un seuil maximal prédéfini par l'utilisateur, ainsi qu'une date d'accès aléatoire.

3.3.1. PROCESSUS DE GENERATION DU GRAPHE DES ACCES FILTRE

Notre approche repose sur l'élaboration de communautés basées sur des filtres relatives à l'utilisateur et d'autres relatives au service, par le biais d'élaboration d'un graphe bipartie en filtrant le graphe obtenue précédemment avec des filtres relatives aux utilisateurs, aux services et aux accès.

Les filtres concernant l'utilisateur sont : le centre d'intérêt, la localisation, l'âge et le sexe, et d'autres filtres comme la religion, le niveau d'étude et la langue comme présenté dans les sections précédentes ; et les filtres correspondant au service sont : le domaine (intérêt) qui est un filtre obligatoire et la localisation du service.

En ajoute aux filtres précédant, le filtre concernant les accès, ce filtre est décomposé en quatre sous-filtres : date, durée, nombre minimum des accès (par défaut 1) et convenance minimum (par défaut 0).

Le premier sous-filtre « date » consiste à récupérer tous les accès entre ou en dehors de deux dates saisi par l'utilisateur, mais aussi récupérer les accès à une date précise, ou après/avant cette date.

Le second sous-filtre « durée » permet de récupérer les accès qui ont duré un certain temps définie par l'utilisateur, ou dans un intervalle de temps, en passant au suivant filtre (nombre minimum des accès) qui permet de définir le nombre minimum des accès entre un utilisateur et un service voulu dans le graphe en coure de génération .

Le dernier filtre alloue à l'utilisateur de notre produit une possibilité de choisir la convenance minimale désirée dans le graphe courant.

A la fin de cette étape, le graphe obtenu sera créé sous un fichier avec le format .net, ce fichier est exploitable avec le logiciel Pajek-XXL qui sera présenté dans le chapitre suivant.

3.4. PROCESSUS DE GENERATION DES COMMUNAUTES

La phase de la génération de communautés est divisée en trois étapes, qui vont être introduites dans la prochaine section.

- **Etape 1** : L'objectif de cette phase est de pouvoir créer des communautés qui respectent les filtres exigés, en se basant sur les résultats obtenus précédemment (graphe d'accès filtrés). C'est à ce niveau qu'on réalise l'intérêt de la résolution du problème par un graphe biparti.

En effet, considérant les accès obtenus antérieurement, il suffit pour chaque service, de déterminer les utilisateurs ayant accédé à ce service. Ces utilisateurs formeront la communauté relative au service.

Cependant, la détection de communautés relatives à un seul service n'est pas suffisante. Le but est de découvrir des communautés relatives à un groupe de service.

- **Etape 2** : La décomposition du réseau à l'étape précédente se caractérise par des chevauchements entre communautés. Cela résulte du fait qu'un utilisateur peut accéder à plusieurs services web, et peut par conséquent appartenir à plusieurs communautés.

  Dans cette phase, on considère deux à deux les communautés obtenues à partir de l'étape précédente. Pour chaque pair de communautés considérées, on crée une nouvelle communauté qui correspond à la fusion de leurs services. Cette communauté sera constituée des utilisateurs extraits de leurs intersections. On répète cette étape jusqu'à ce qu'il n'y ait plus de chevauchements entre les communautés. [29]

- **Etape 3 (Visualisation) :** Pour le graphe de communautés généré, un fichier sous format .net sera créé et visualisé avec le logiciel Pajek-XXL.

4. MODELISATION

La modélisation est le pilier de toute activité qui conduit au déploiement d'un produit informatique. Les modèles sont construits pour spécifier la structure et le comportement attendu d'un système pour visualiser, contrôler son architecture et pour mieux comprendre son organisation. [30]

4.1. DEFINITION UML

Le langage de modélisation unifié, de l'anglais Unified Modeling Language (UML) est un langage de modélisation orienté objet, c'est-à-dire que toutes les entités modélisées sont des objets ou se rapportent à des objets : un objet possède une structure de données (avec ce qui s'appelle des « attributs ») et des comportements (avec ce qui s'appelle des « opérations »).

Ce langage est né de la fusion de plusieurs méthodes existant auparavant et est devenu désormais la référence en termes de modélisation objet adopté par l'Object Management Group (OMG), il est ainsi devenu le standard de l'industrie.

Il s'agit est une forme visuelle fondée sur des diagrammes pour la visualisation de la conception d'un système.

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet, mais aussi pour des systèmes non informatiques.

4.2. ASPECT FONCTIONNEL

L'aspect fonctionnel permet de savoir qui utilisera le logiciel et ses fonctionnalités, mais aussi le déroulement des actions et quelles informations seront utilisées pour cela.

4.2.1. Diagramme de cas d'utilisation (use case)

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit.

Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils ont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique [31].

La représentation graphique des cas d'utilisation fait appel à quatre symboles élémentaires :

1 -l'acteur (petit bonhomme).
2-le cas d'utilisation (ellipse), correspondant à une fonctionnalité du système
3- les limites du système (grand rectangle entourant les cas d'utilisation).
4-l'arc ou segment de droite liant l'acteur à un cas d'utilisation

5-l'arc ou segment de droite liant un cas d'utilisation à une autre [32].

Dans notre cas, il n'existe qu'un seul acteur qui est le décideur.

4.2.2. Génération des utilisateurs et des services

Pour accomplir le processus de création de communautés il faut d'abord créer les profils utilisateurs et services :

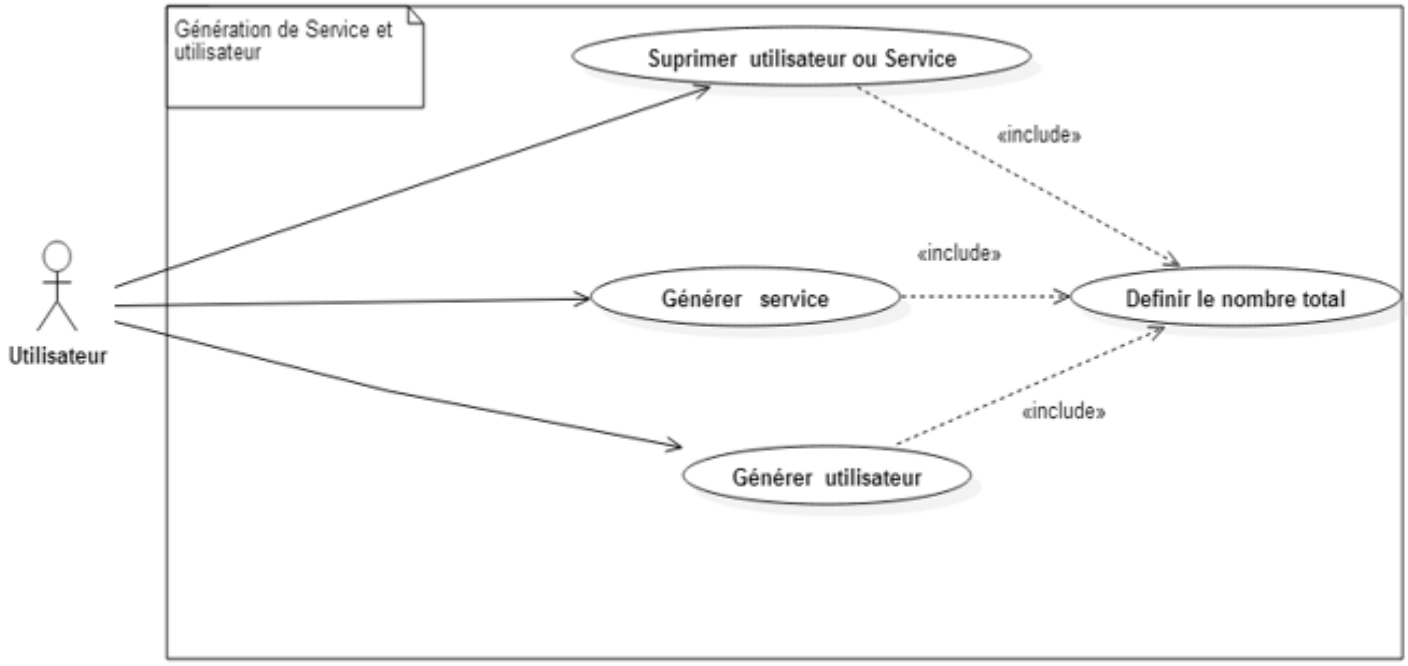


Figure II.4. Cas d'utilisation de génération des utilisateurs et des services

4.2.3. Génération de graphe de convenance

Après la génération des profils, on passe à la création du graphe de convenance s présenté dans ce qui suit :

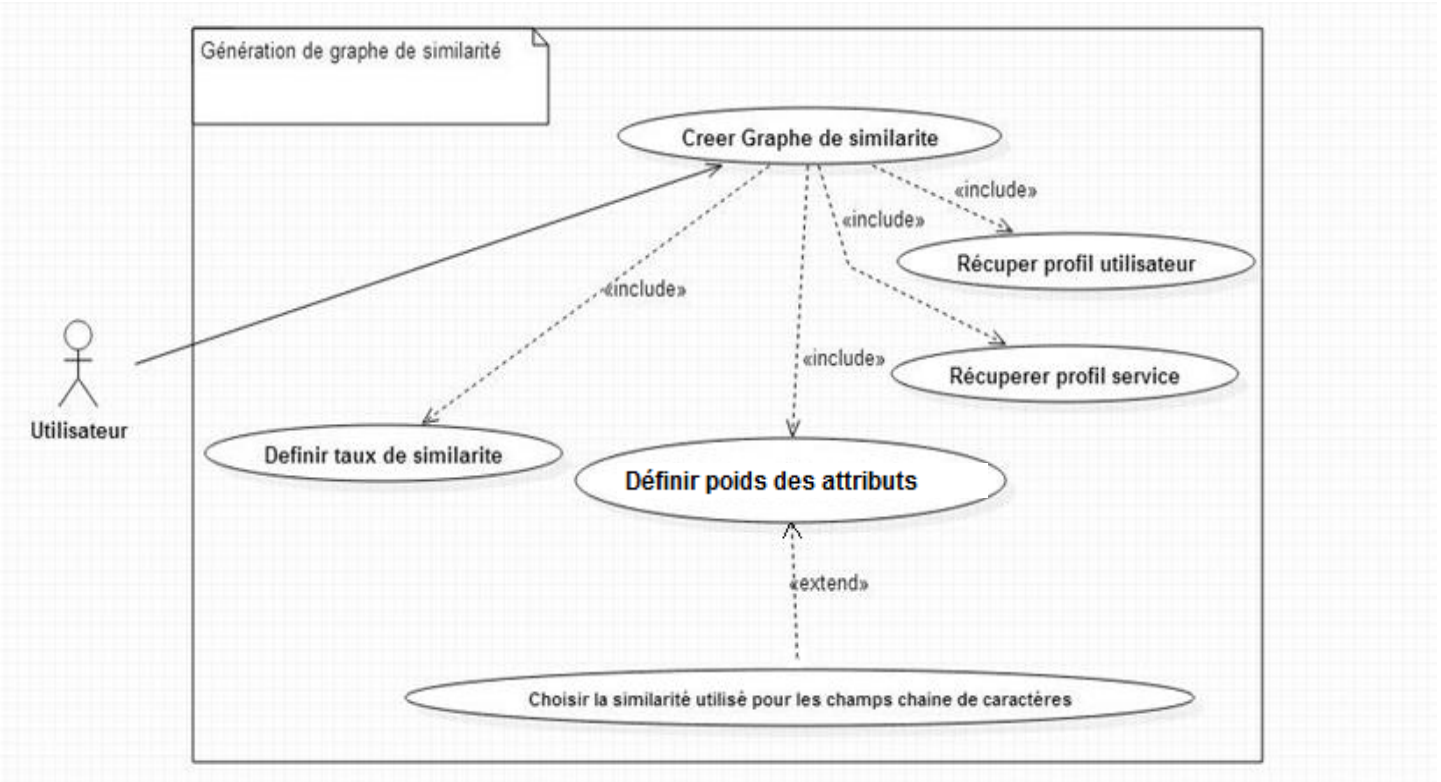


Figure II.5. Cas d'utilisation génération du graphe de convenance

4.2.4. Génération des accès

Le processus suivant utilise le graphe précédemment créé pour la génération du graphe des accès et en précisant d'autres paramètres :

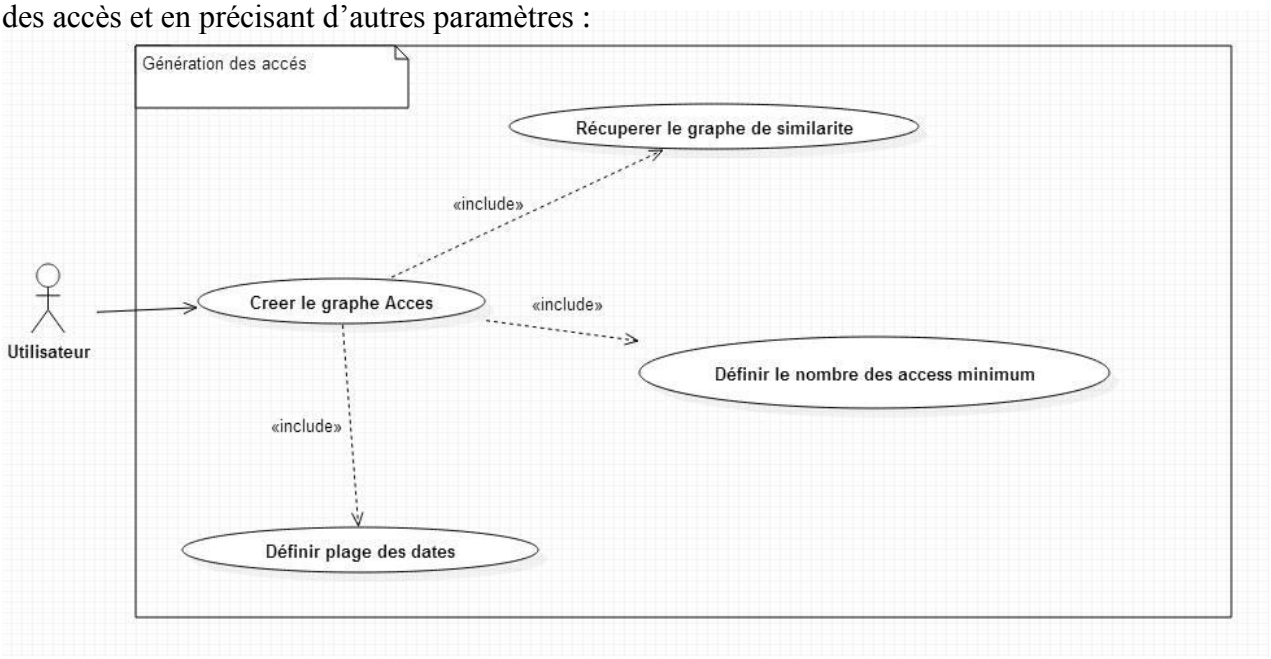


Figure II.6. Cas d'utilisation génération des accès

4.2.5. Génération

Après avoir cree le graphe des acces, on applique les differents filtres pour obtenir le graphe d'accès filtrés. Cette étape est schématisée par le diagramme suivant :

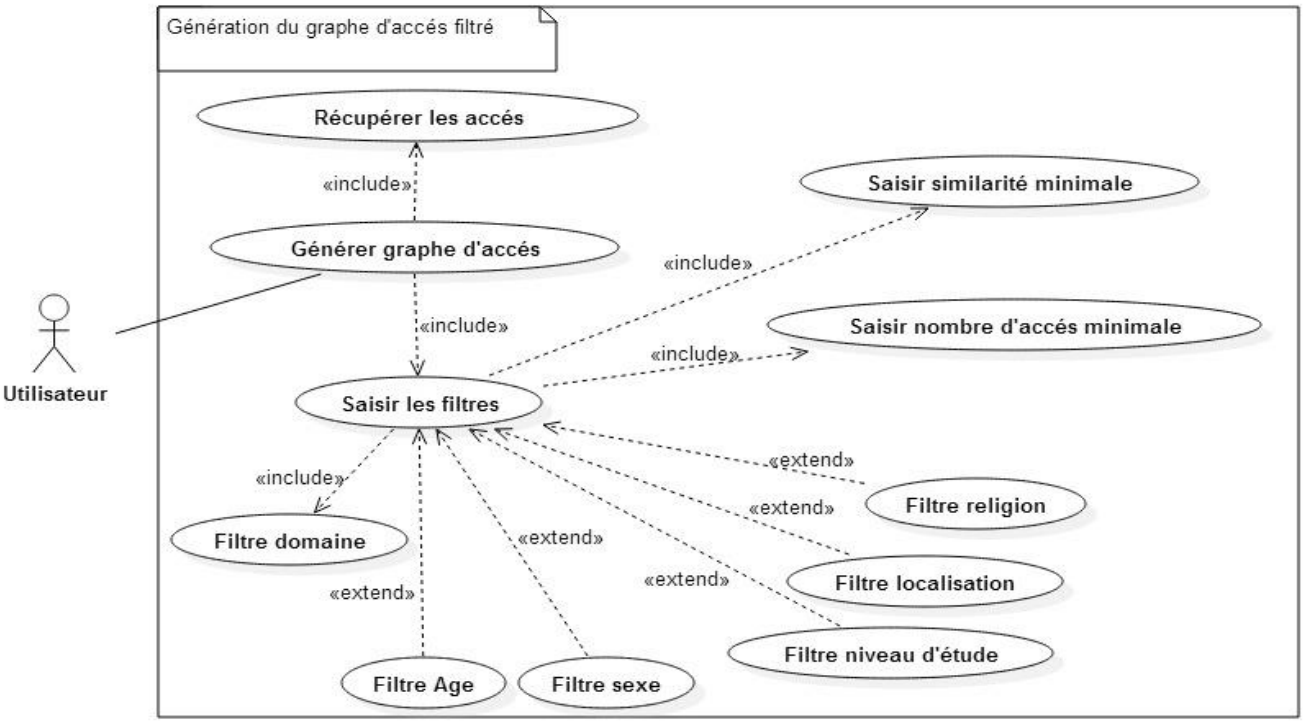


Figure II.7. Cas d'utilisation génération du graphe d'accès filtré

4.2.6. Génération du graphe de communautés

L'étape de création des communautés utilise le résultat du processus précédent, et créé les communautés selon le degré saisi par l'utilisateur :

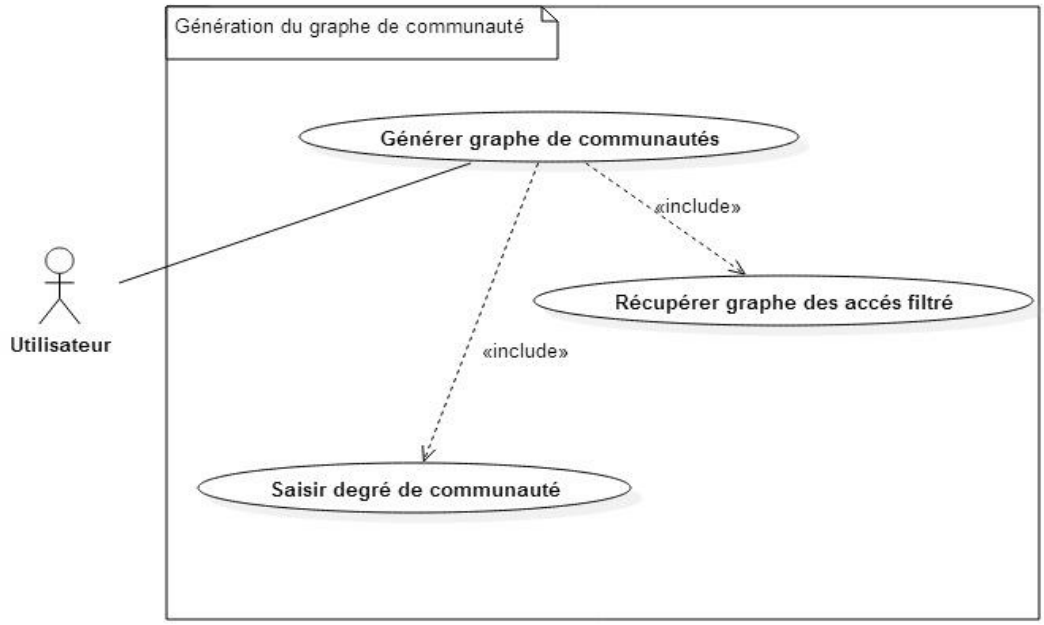


Figure II.8. Cas d'utilisation pour la génération des communautés

4.3. ASPECT DYNAMIQUE

Les aspects dynamiques représentent la collaboration entre objets pour la réalisation de tout ou une partie d'un cas d'utilisation.

4.3.1. Diagramme de séquence (Sequence Diagram) :

Le diagramme de séquence représente la succession des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre.

Les principales informations contenues dans ce diagramme sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique,

Un diagramme de séquence étant constitué de :

- ligne de vie : rectangle, auquel est accrochée une ligne verticale pointillée, contenant une étiquette
- Message : définit une communication particulière entre des lignes de vie [33]

4.3.2. Lancement de l'application

Le diagramme suivant schématise la première fenêtre lancée lors de l'utilisation du logiciel, l'utilisateur est invité alors à choisir l'un des contextes utilisateur ou service pour passer à l'étape suivante :

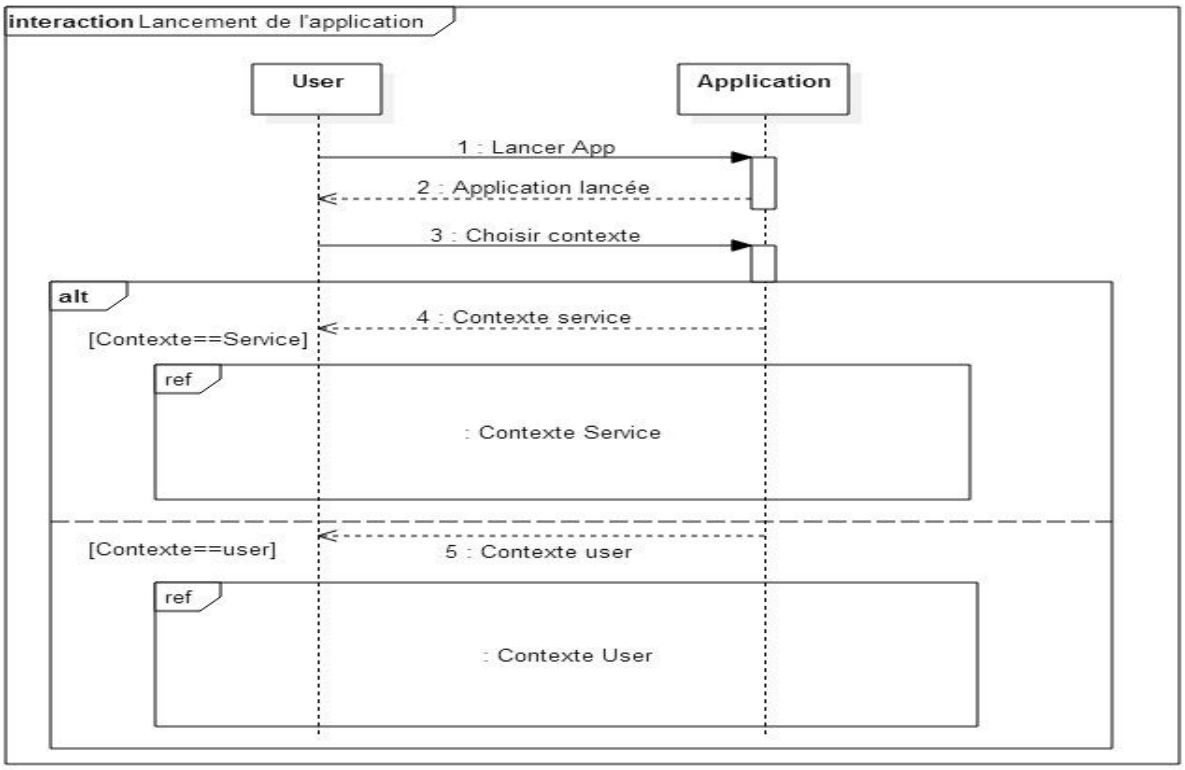


Figure II.9. Diagramme de séquence de lancement de l'application

4.3.3. Contexte Utilisateur/Service

Après avoir choisi un contexte, l'utilisateur peut maintenant exécuter de nombreuses taches telles que la suppression ou la création de profils comme illustré dans la figure suivante :

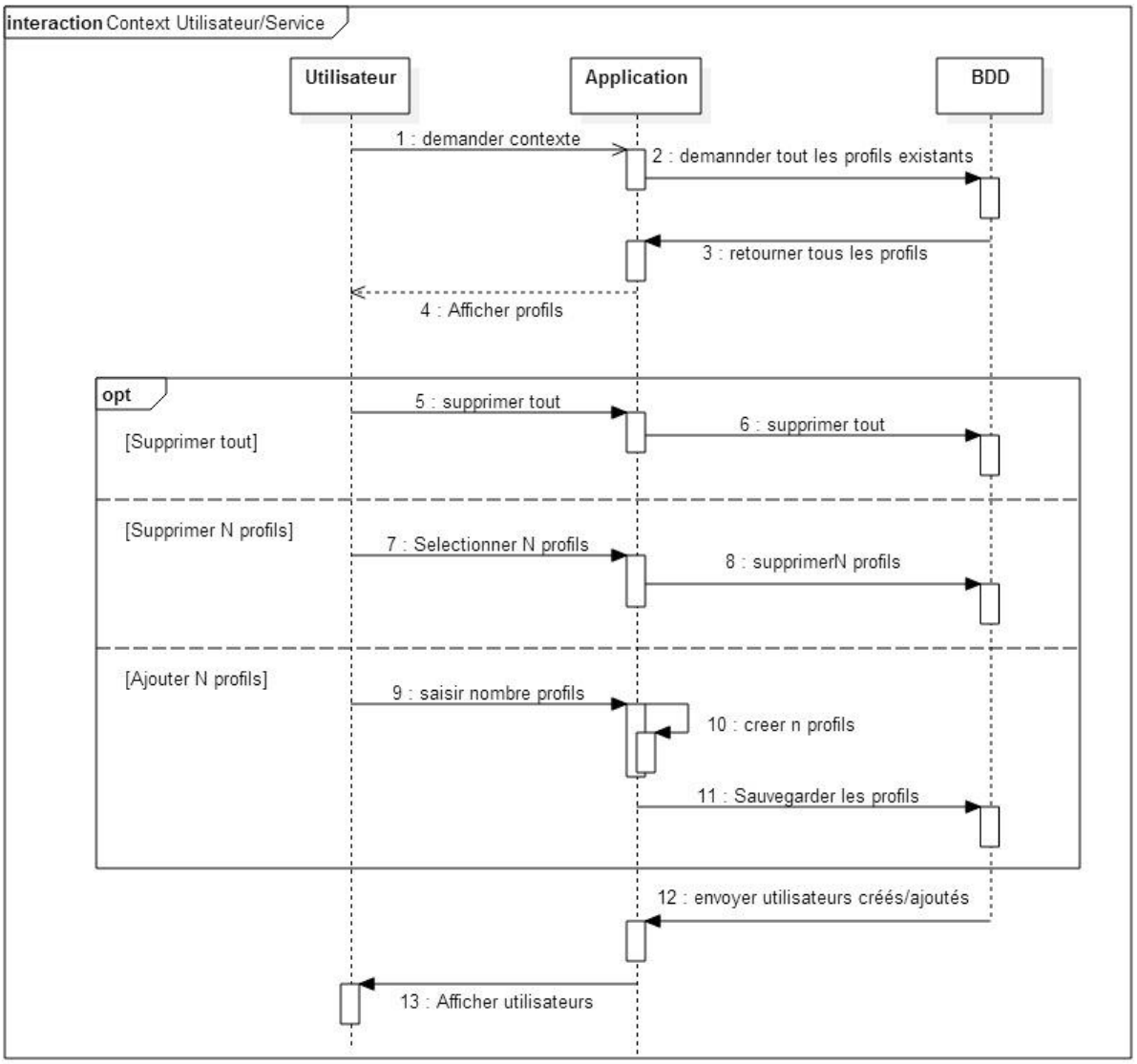


Figure II.10. Diagramme de séquence contexte utilisateur/service

4.3.4. Calcul de la convenance

Le diagramme ci-dessous illustre le processus de calcul de convenance :

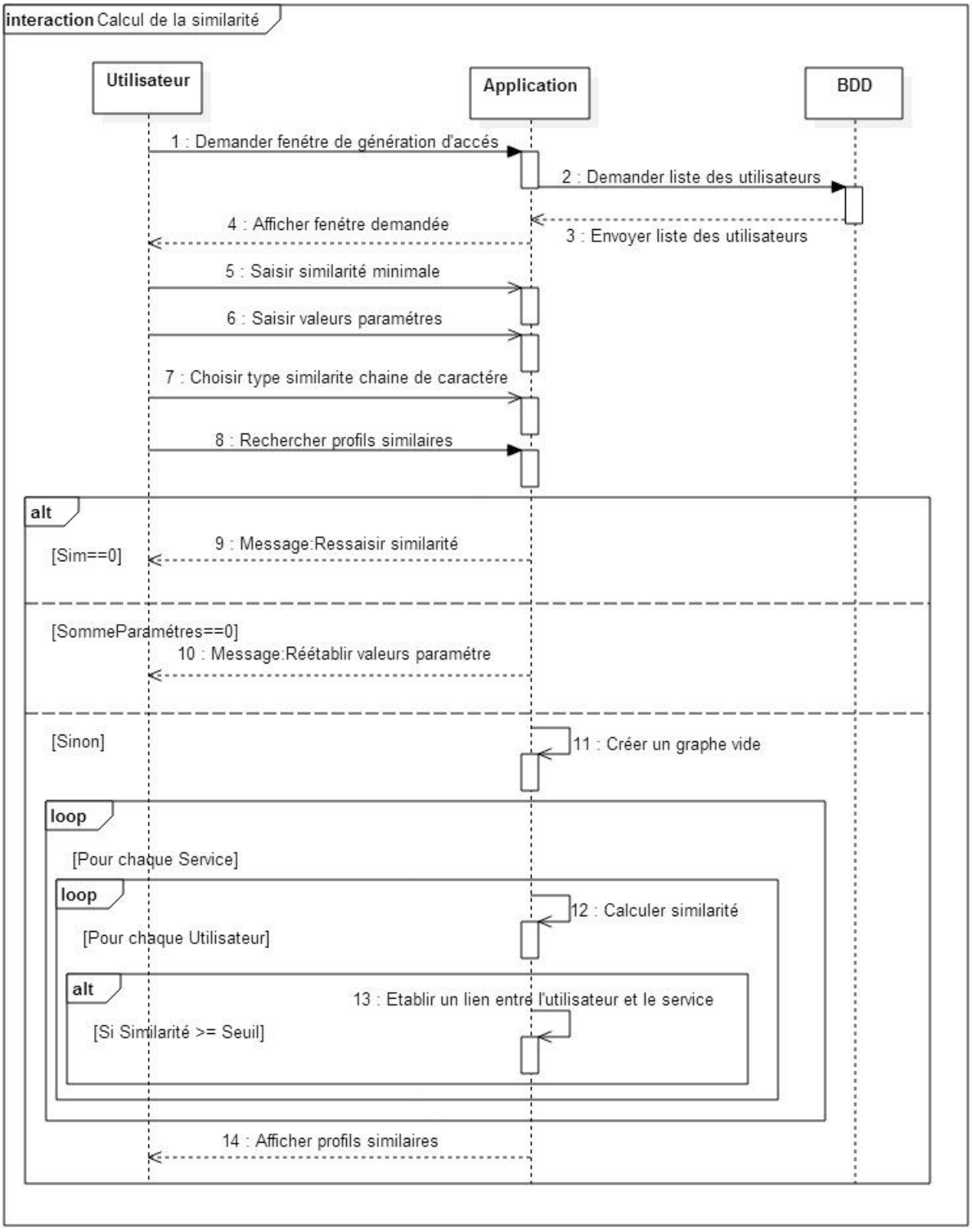


Figure II.11. Diagramme de séquence calcul de convenance

4.3.5. Génération des accès

Le diagramme suivant donne une vue général sur la génération des accès entre les profils utilisateurs et service :

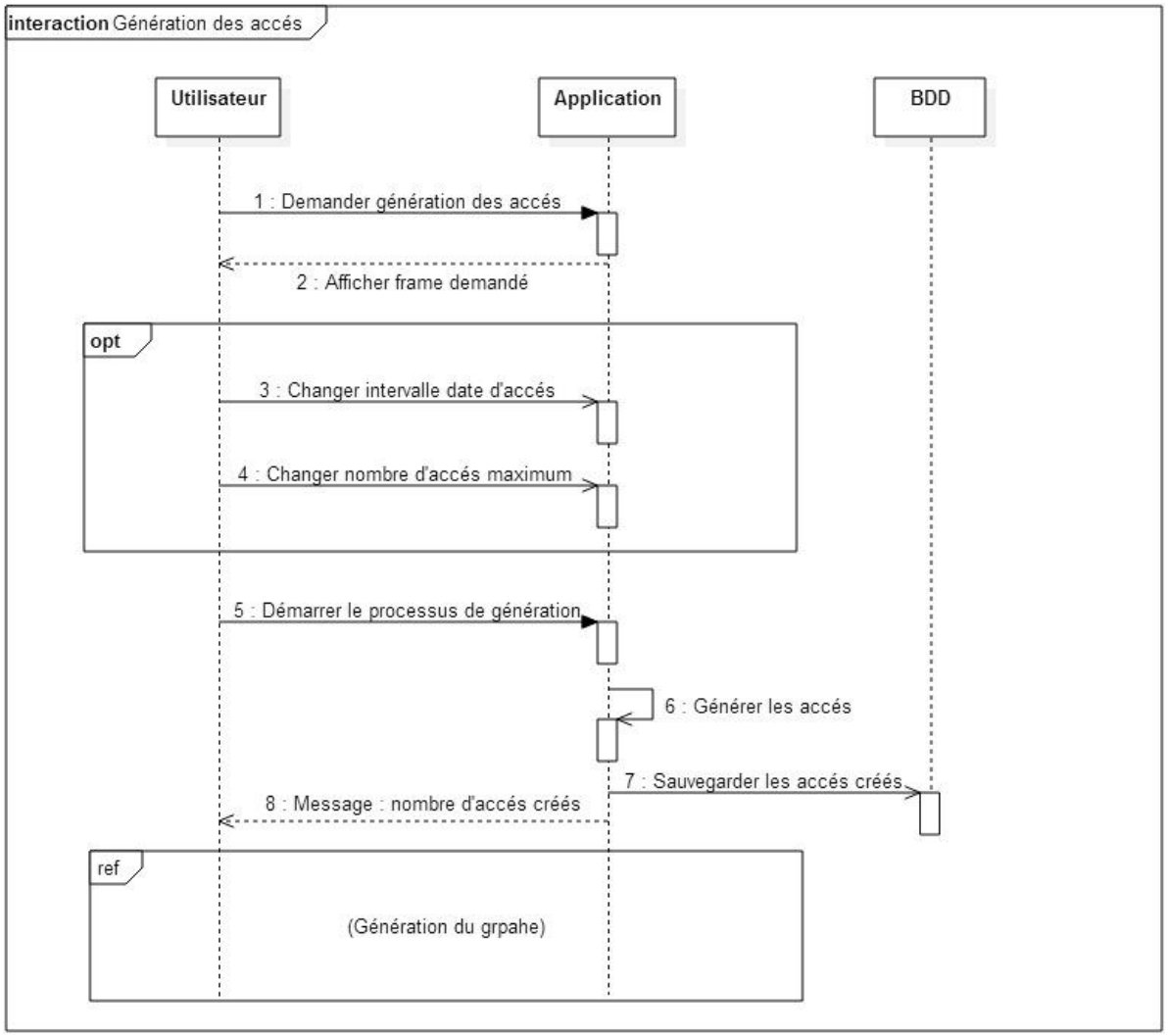


Figure II.12. Diagramme de séquence pour la génération des accès

4.3.6. Génération du graphe d'accès filtré

En faisant passer le graphe obtenu précédemment sur plusieurs filtres, on obtiendra un nouveau graphe filtré. Le diagramme qui suit explique ce qui a été dit :

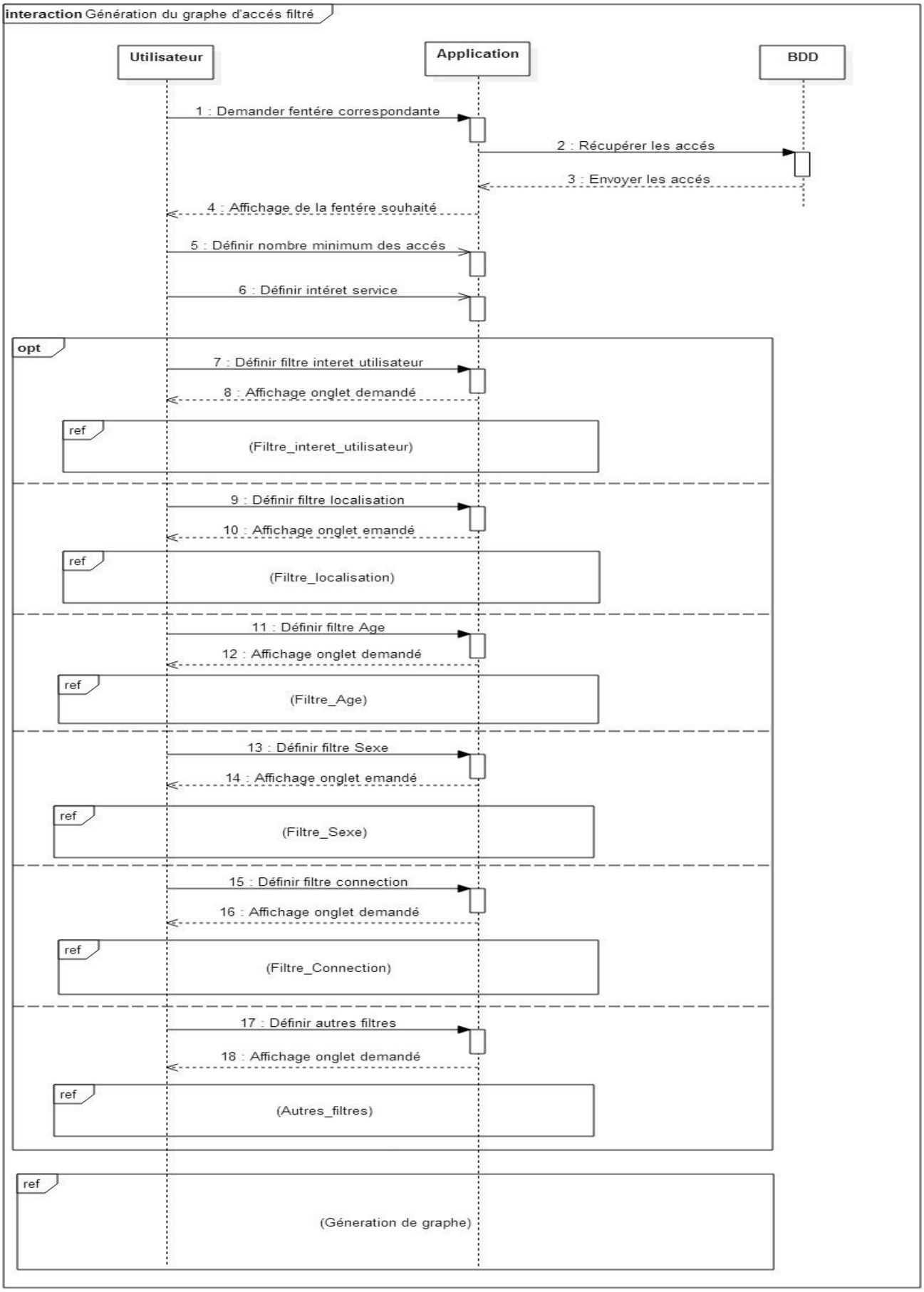


Figure II.13. Diagramme de séquence pour la génération du graphe d'accès filtrés

4.3.7. Génération de graphe

Le diagramme de séquence suivant schématise l'opération d'export d'un fichier .net :

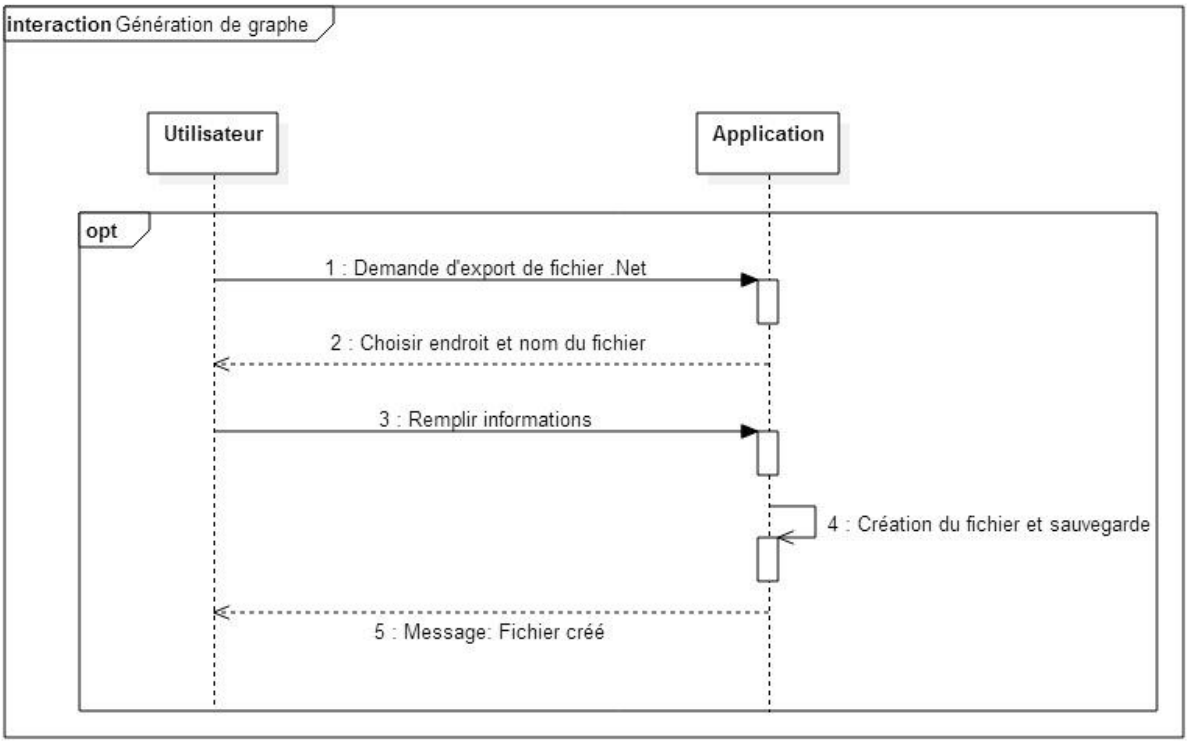


Figure II.14. Diagramme de séquence export fichier .net

4.3.7.1. Filtre intérêt utilisateur

Ce premier filtre concerne l'intérêt recherché par l'utilisateur :

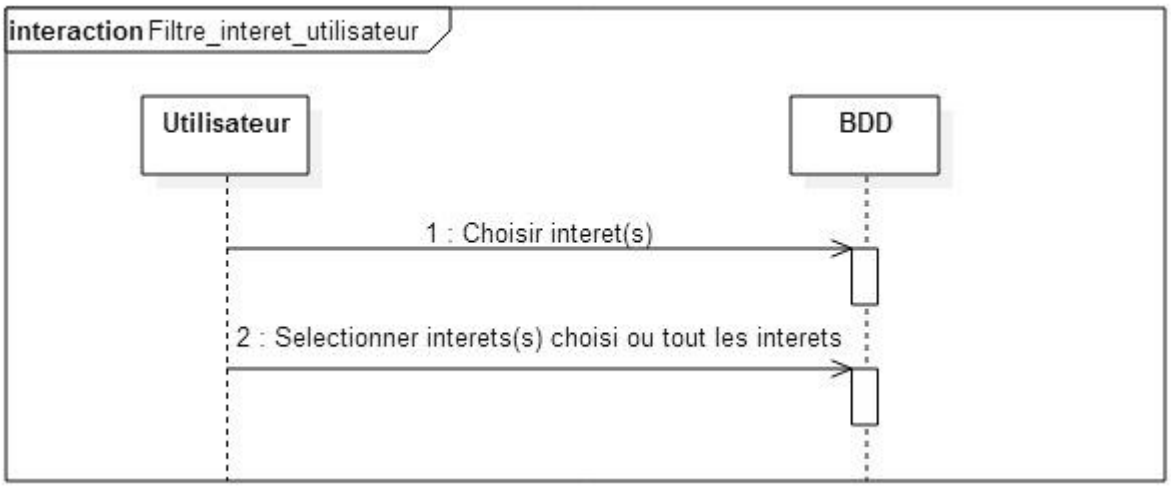


Figure II.15. Diagramme de séquence pour le filtre intérêt utilisateur

4.3.7.2. Filtre Sexe

En passant à ce filtre, l'utilisateur choisi le sexe de la communauté ciblé, comme il peut cibler tout le monde en ne saisissant pas ce filtre.

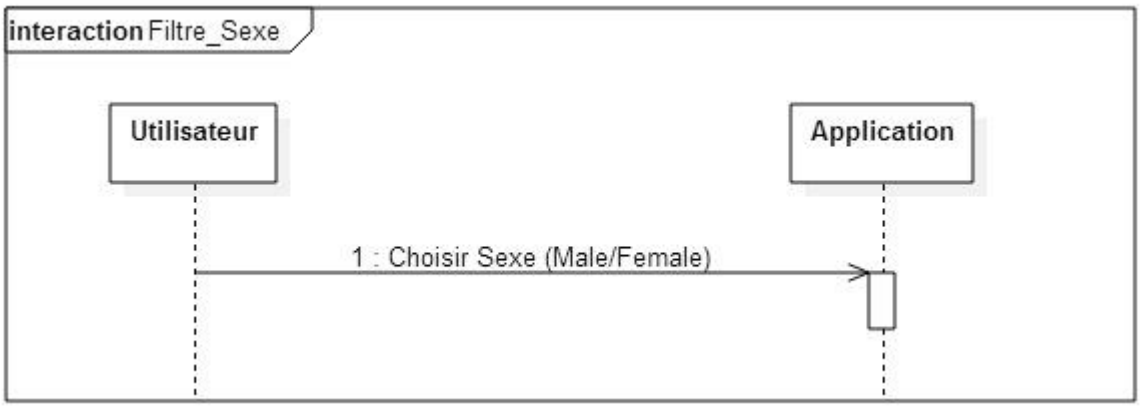


Figure II.16. Diagramme de séquence relative au sexe

4.3.7.3. Filtre Age utilisateur

Ce filtre permet de choisir la tranche d'âge ou l'âge fixe de la communauté qui sera créé :

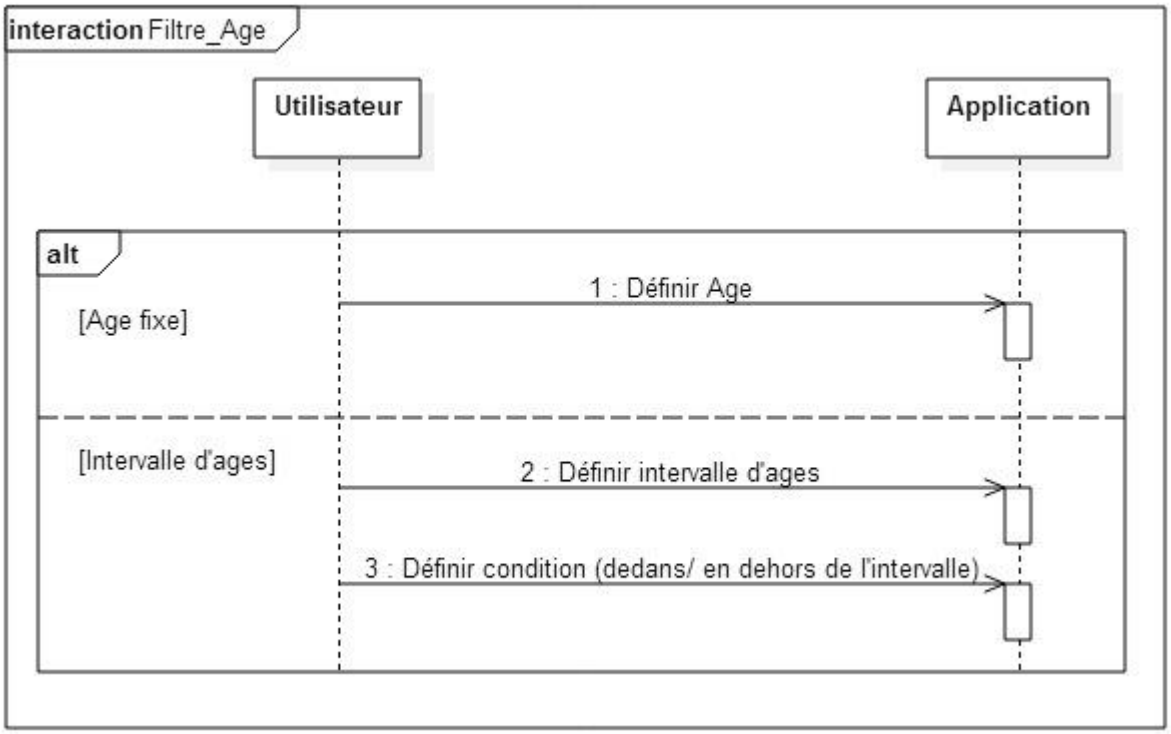


Figure II.17. Diagramme de séquence pour le filtre « Age »

4.3.7.4. Filtre Localisation

Grace à ce filtre, l'utilisateur peut choisir de filtrer la localisation soit de l'utilisateur, soit du service, en choisissant le pays de l'utilisateur, la région laquelle est destiné le service ou les coordonnées (longitude et latitude) pour les deux.

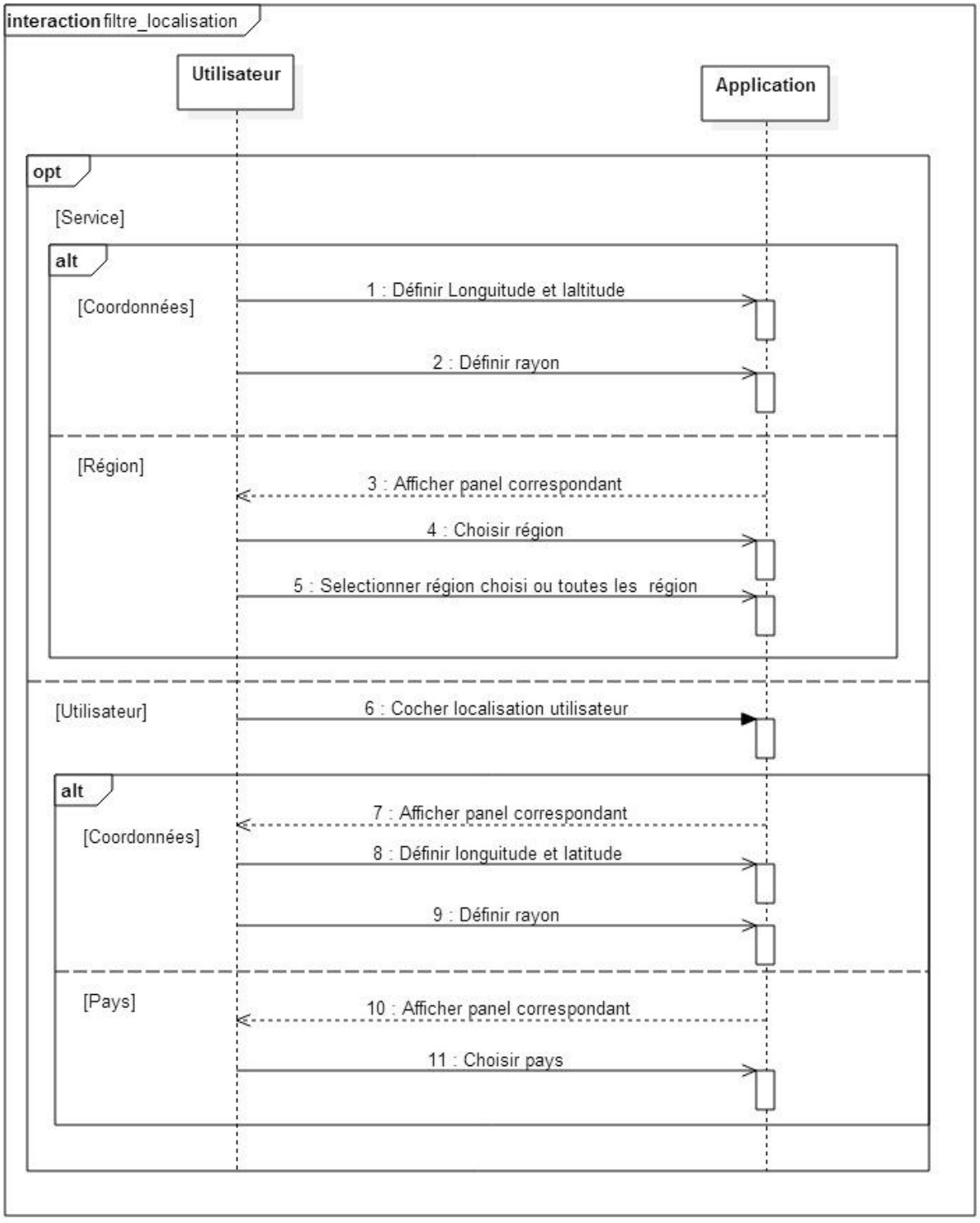


Figure II.18. Diagramme de classe pour le filtre localisation

4.3.7.5.Filtre des connections

Ce filtre cible les accès, l'utilisateur peut filtrer les accès soit par date ou par durée de connexion :

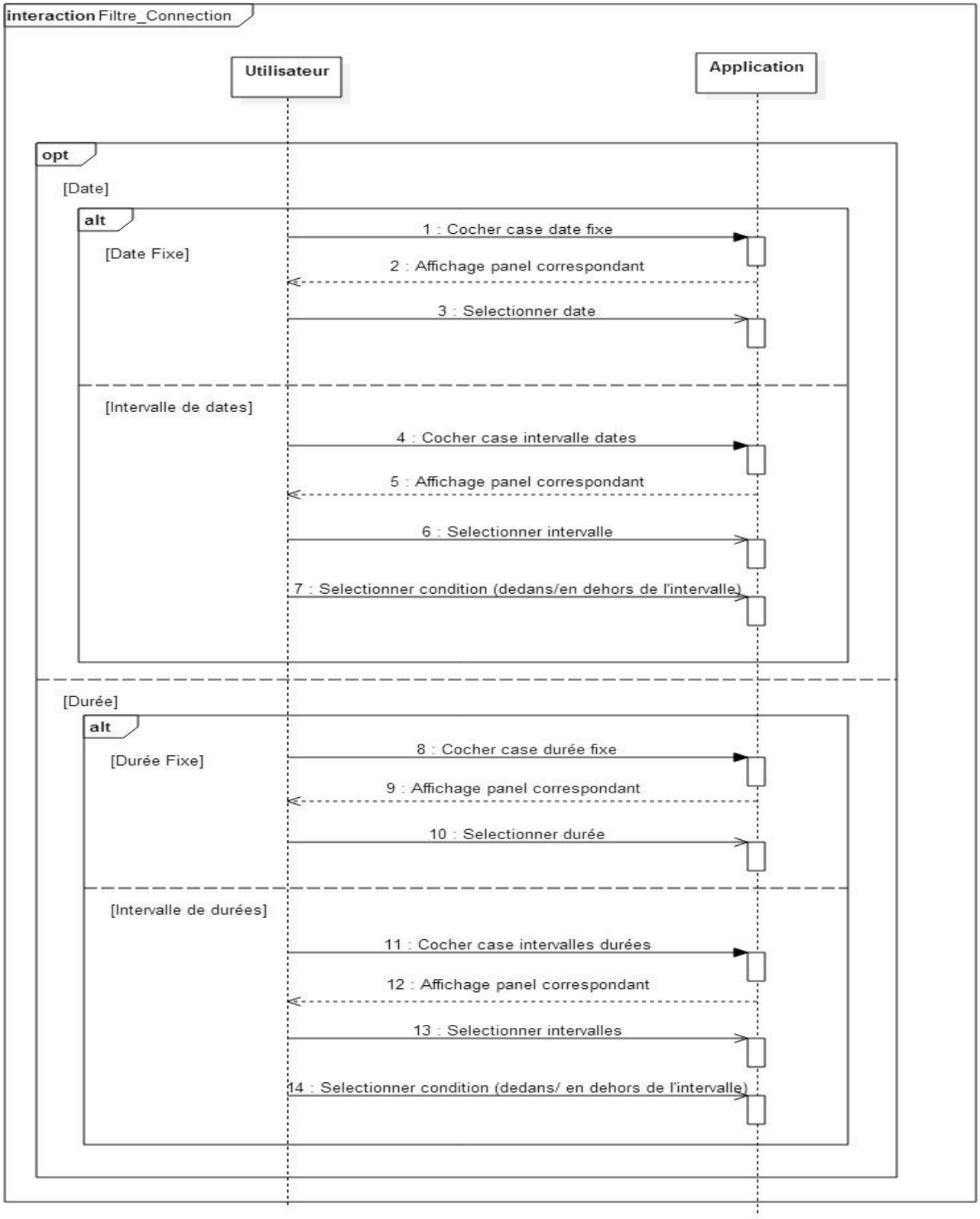


Figure II. 19. Diagramme de séquence pour le filtre « Connection »

4.3.7.6. Spécification d'autres filtres

Ce dernier filtre regroupe des sous-filtres : Religion, Niveau d'étude et Langue qui concerne l'utilisateur.

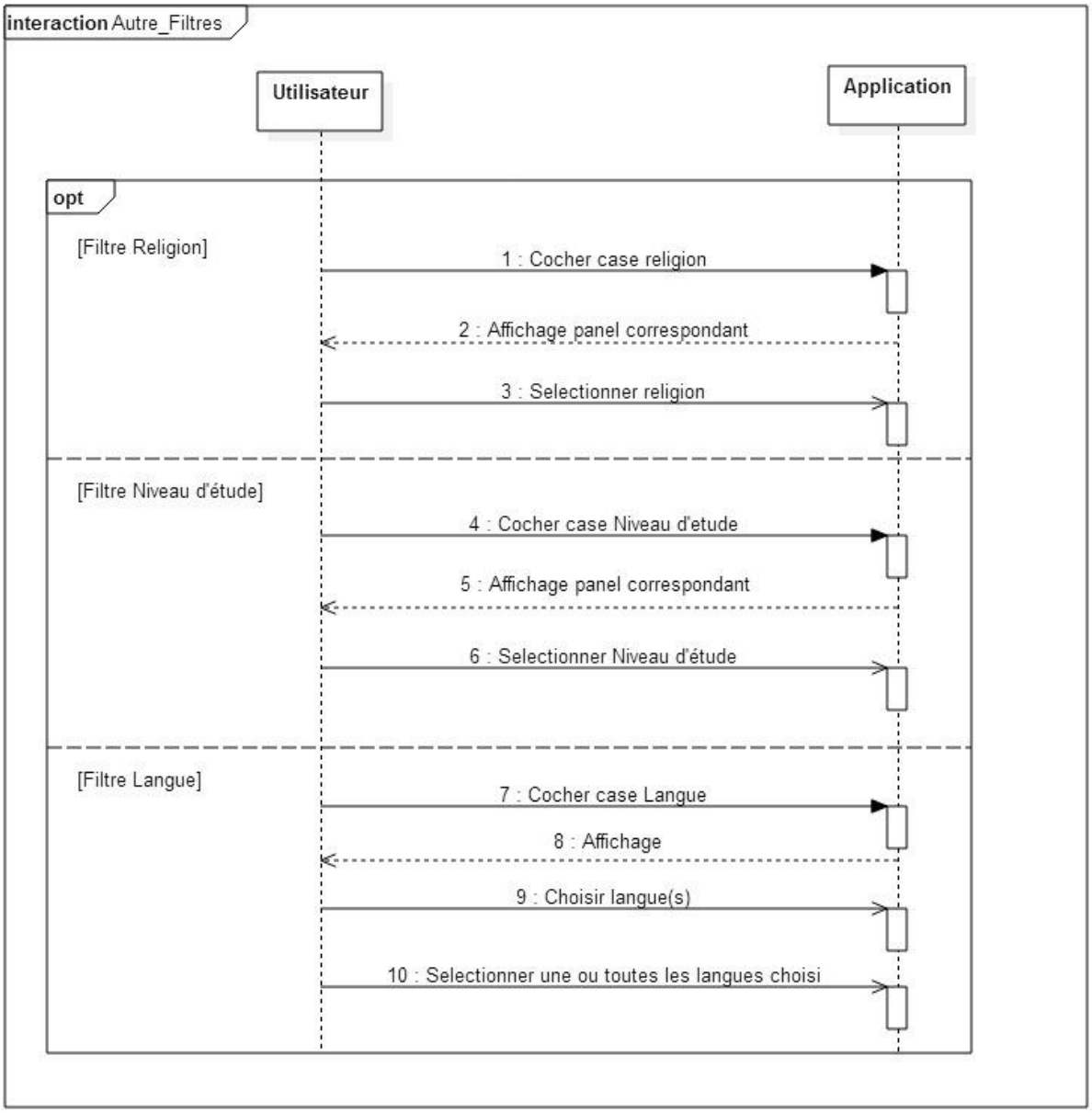


Figure II.20. Diagramme de séquence des filtres : Religion, niveau d'étude, langue

4.3.8. Génération des communautés

Le dernier diagramme schématise l'étape de génération de communautés :

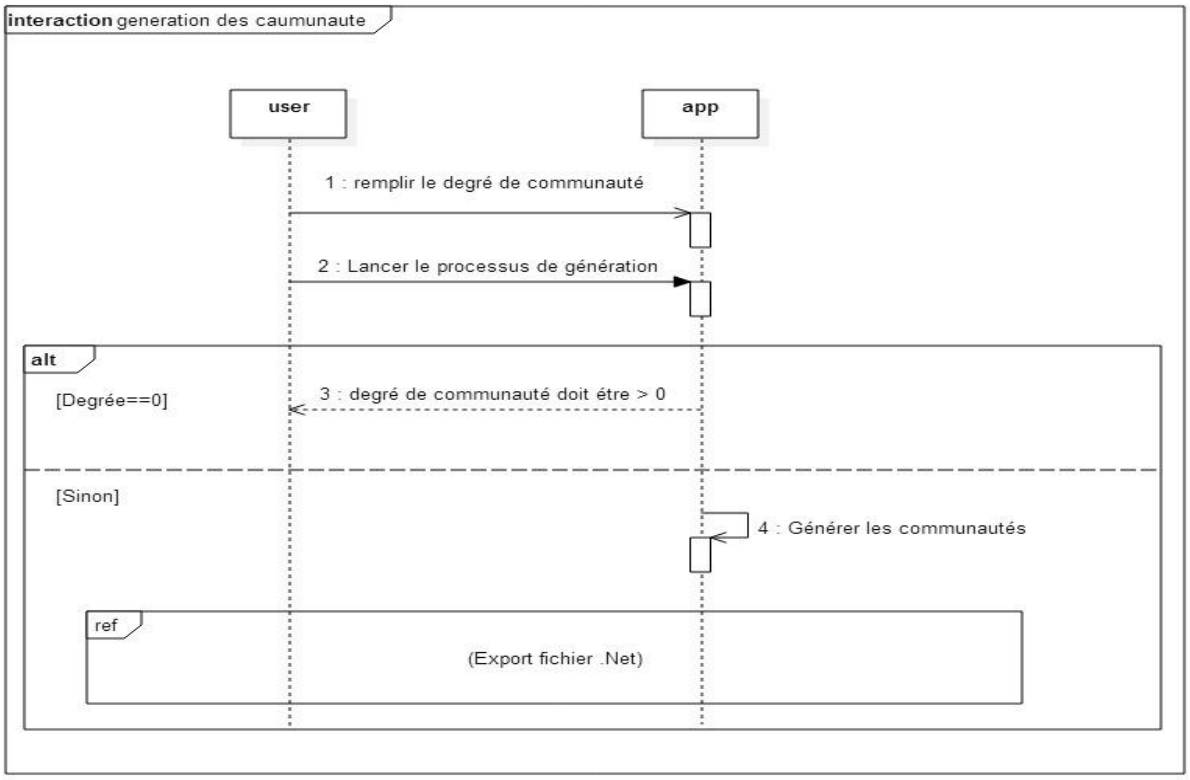


Figure II.21. Diagramme de séquence pour la génération de communautés

4.4. ASPECT STATIQUE

Les diagrammes de l'aspect statique facilitent l'analyse et la conception du système d'information, ainsi que la modélisation de sa structure logique.

4.4.1. Diagramme de classe (Class Diagram)

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il aussi décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation). [34]

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation. [35]

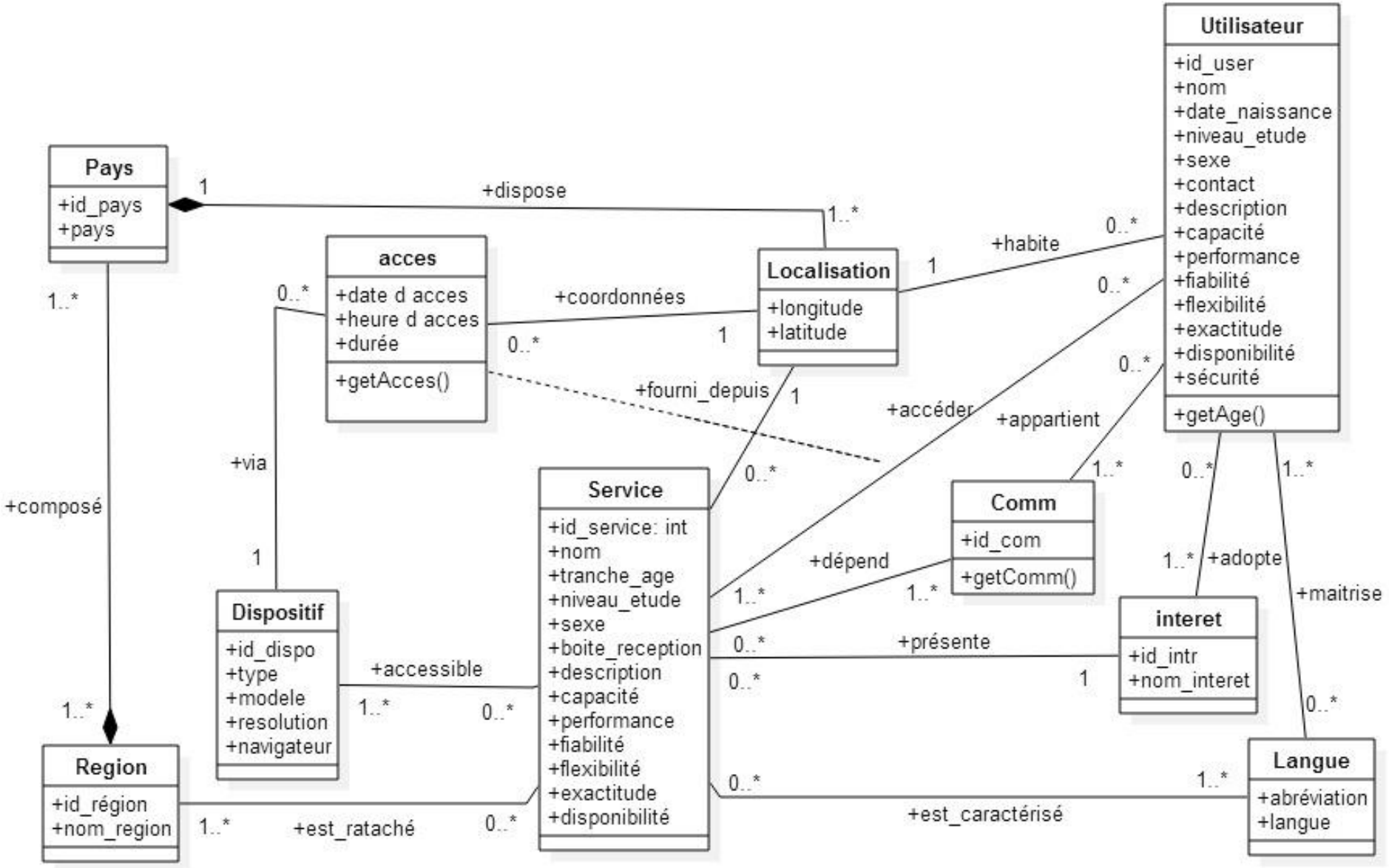


Figure II.22. Diagramme de classes

4.4.2. Règles de transformation

Après avoir élaboré le diagramme de classe, il faut le traduire en Modèle Relationnel de Données (MLD) et ceci en respectant un nombre de règles de transformation qui sont les suivantes :

- Nom de la classe UML = > Nom de la relation (table)
- Attributs de la classe UML = > Champs de la table
- Relation entre classes = > Création d'un nouveau champ ou d'une nouvelle table selon la multiplicité
  - **Liens 1-1** entre deux classes A et B : Création d'un attribut représentant l'identité de la classe B (resp. A) dans la table définie par A (resp. B) i.e. création d'une clé étrangère référençant B (resp. A) dans la table définie par A (resp. B)
  - **Lien 1-n ou 1-*** entre A et B : Création d'un attribut représentant l'identité de la classe A dans la table définie par B i.e. création d'une clé étrangère référençant A dans la table définie par B. (idem si 1 remplacé par (0..1)
  - **Lien n-m ou *-*** entre A et B : Création d'une table supplémentaire composé des deux attributs représentant l'identité des deux classes. La clé primaire de la nouvelle table est composée des deux attributs.

4.5. MODELE RELATIONNEL DE DONNEES

Utilisateur (**id_user**, nom, date_naissance, niveau_étude, sexe, contact, description, capacité, performance, fiabilité, flexibilité, exactitude, disponibilité, *Longitude *, latitude *)*

Service (**id_service**, nom, tranche_age, niveau_étude, sexe, boite_reception, description, capacité, performance, fiabilité, flexibilité, exactitude, disponibilité, id_*intr**, *Longitude *, latitude **)

Accès (id_acces, date_d'acces , duree , *id_user*, id_service*, Longitude *, latitude *, id_dispo *)*

Comm (id_com)

Intérêt (**id_intr**, nom_intérêt)

Langue (**abréviation**, langue)

Pays (id_pays, pays)

Région (**id_region**, nom_région)

Localisation (**longitude, latitude,** *id_pays **)

Dispositif (id_dispo, type, modèle, résolution, navigateur)

Maitrise (***id_user*, abréviation****)

Adopte (***id_user*, id_intr****)

Appartient (*id_user*, id_com)** ;

Est_caractérisé (***id_service*, abréviation****)

Dépend (***id_service*, id_com****)

Composé (***id_pays*, id_région**)**

Accessible (***id_dispo,* id_service****)

Est_ratché (***id_région*, id_service****)**

5. CONCLUSION

Dans ce chapitre, nous avons présenté notre approche de génération de communautés basées service dans la première partie du chapitre nous avons introduit notre idée pour le calcul de convenance entre les profils services et utilisateurs et ainsi passer à l'étape de création des accès dont est basé la génération du graphe d'accès filtrés pour ensuite générer les communautés qui satisfaites les filtres exigé par l'utilisateur.

La deuxième partie a été consacrée à la modélisation de principales fonctionnalités de l'application par les trois aspects fonctionnel, dynamique et statique. On a conclu le chapitre par l'exposition des algorithmes appliqué dans notre approche.

Dans le prochain chapitre, nous allons mettre en œuvre notre idée par la présentation des outils utilisés dans la réalisation de l'application ainsi que l'interface et les fonctionnalités essentielles.

III

Implémentation et tests

1. INTRODUCTION

Dans ce dernier chapitre, nous allons présenter l'implémentation et la mise en œuvre de notre approche en présentant en premier lieu les outils utilisés pour la conception, ainsi que celui de la visualisation.

Une seconde partie sera destinée à l'illustration de l'interface ainsi que l'explication des fonctionnalités du produit, et on terminera par une démonstration de l'usage de l'application.

On conclura ce chapitre avec une étude comparative de notre approche avec une approche très utilisés celle de l'algorithme de Louvain.

2. LES OUTILS UTILISES

Cette partie est consacrée aux outils utilisés dans l'implémentation de l'application, des langages de programmation aux outils utilisés pour la visualisation des graphes ainsi que les technologies usées :

2.1. LES LANGAGES DE PROGRAMMATIONS UTILISES

2.1.1. JAVA

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. Java est rapide, sécurisé et fiable. Des ordinateurs portables aux centres de données, des consoles de jeux aux superordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente sur tous les fronts. [36]

2.1.2. SQL

SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles.

Ce langage été créé en 1974 et normalisé depuis 1986, il est le langage reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBDR) du marché.

Il est décomposé en quatre parties :

- La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles
- La partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données
- La partie langage de contrôle de transaction permet de commencer et de terminer des transactions
- La partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes

2.2. LES ENVIRONNEMENTS DE DEVELOPPEMENT

2.2.1. NETBEANS

NetBeans est à l'origine un environnement de développement intégré (EDI) Java. NetBeans fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. Quelque part en 2002, Sun a décidé de rendre NetBeans open-source.

Mais NetBeans n'est pas uniquement un EDI Java. C'est également une plateforme, vous permettant d'écrire vos propres applications Swing. Sa conception est complètement modulaire : Tout est module, même la plateforme. Ce qui fait de NetBeans une boite à outils facilement améliorable ou modifiable.

En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Compilé en java, NetBeans est disponible sous Windows, Linux, Solaris Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

2.2.2. MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, Informix et Microsoft SQL Server.

MySQL fonctionne sur de nombreux systèmes d'exploitation différents dont Windows, Mac Os X, Linux et beaucoup d'autres.

Les bases de données sont accessibles en utilisant les langages de programmation C, C++, Java, Python, Ruby …etc.

De nombreuses entrprises, dont Google, Yahoo !, Youtube, Adobe, Airbus, Alstom, Credit agricole, AFP, Reuters, BBC News, Ernst & Young, Alcatel-Lucent l'utilisent encore.

2.3. PAJEK-XXL

Pajek-XXL est un logiciel d'analyse et de visualisation de réseaux pour Windows Mac et Linux. Développé par deux chercheurs slovènes, Vladimir Batagelj et Andrej Mrvar, il a le grand mérite d'être gratuit et puissant, ce qui lui a permis de s'imposer comme un des outils les plus utilisés en analyse des réseaux. [37]

Il permet d'étudier des graphes de grandes taille, il est compatible avec les logiciels existants. Deux autres avantages c'est qu'il est fourni avec un wiki et de nombreux jeux de données sont disponibles. [38]

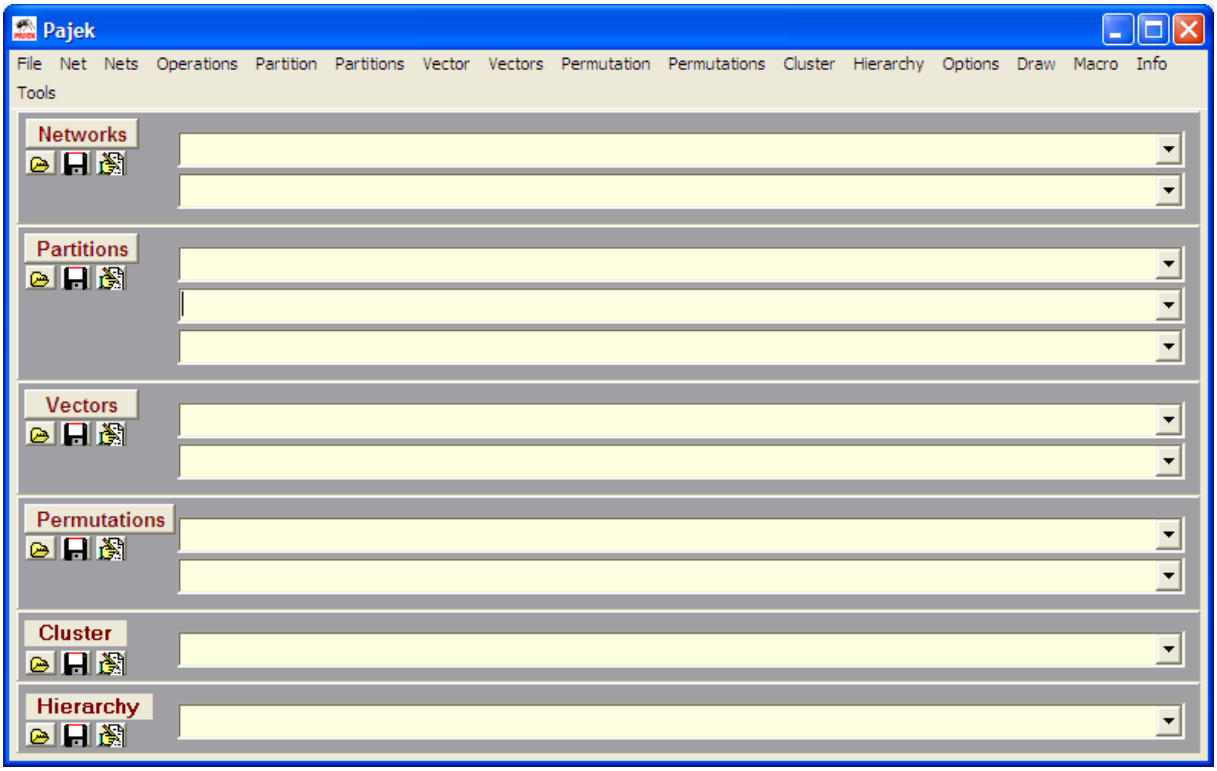


Figure III.1. Interface du logiciel Pajek-XXL

2.4. PATTERN MVC

Modèle-Vue-Contrôleur (abr. MVC) est un motif d'architecture logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web.

Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

2.4.1. LE MODELE

C'est l'élément qui contient les données ainsi que de la logique en rapport avec les données : validation, lecture et enregistrement. Le modèle représente l'univers dans lequel s'inscrit l'application. Par exemple pour une application de banque, le modèle représente des comptes, des clients, ainsi que les opérations telles que dépôt et retraits, et vérifie que les retraits ne dépassent pas la limite de crédit. Le modèle est indépendant de la vue et du contrôleur et ne s'en sert pas.

2.4.2. LA VUE

C'est la partie visible d'une interface graphique. La vue se sert du modèle, et peut être un diagramme, un formulaire, des boutons, etc. Une vue contient des éléments visuels ainsi que la logique nécessaire pour afficher les données provenant du modèle. Dans une application web une vue contient des balises HTML.

2.4.3. LE CONTROLEUR

C'est le module qui traite les actions de l'utilisateur, modifie les données du modèle et de la vue.

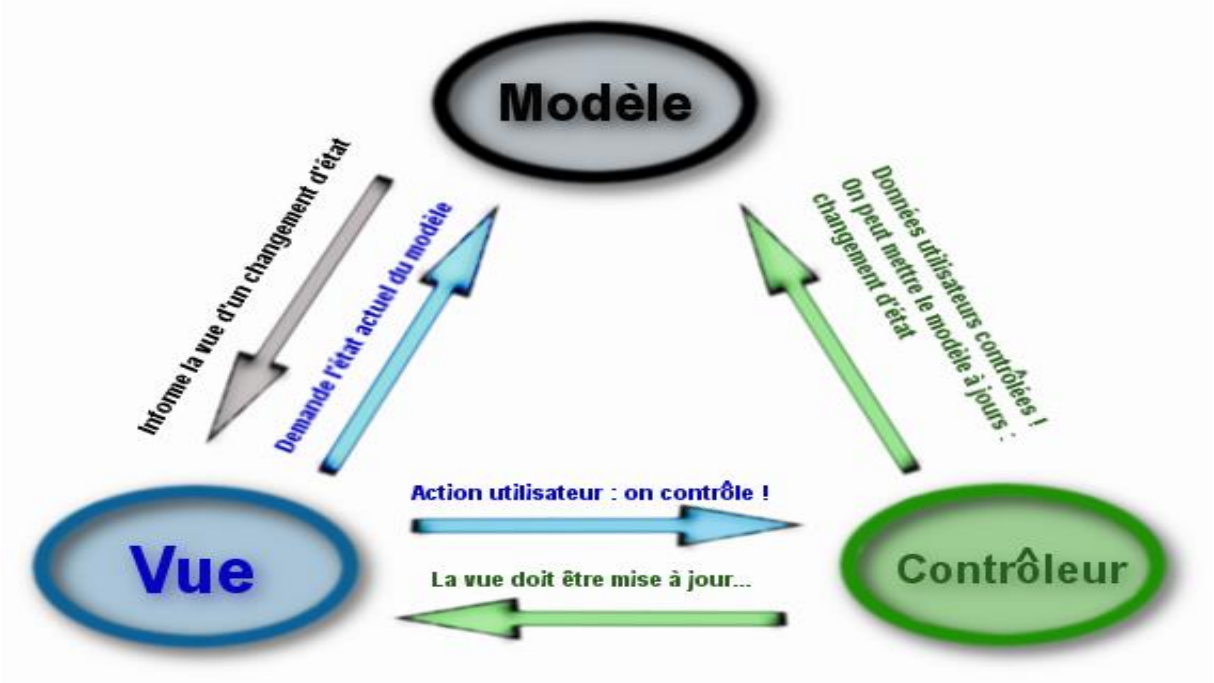


Figure III.2. Schéma du MVC [39]

2.5. TECHNOLOGIE RMI

RMI (Remote Method Invocation) est une API Java permettant de manipuler des objets distants (c'est-àdire un objet instancié sur une autre machine virtuelle, éventuellement sur une autre machine du réseau) de manière transparente pour l'utilisateur, c'est-à-dire de la même façon que si l'objet était sur la machine virtuelle (JVM) de la machine locale.

Ainsi un serveur permet à un client d'invoquer des méthodes à distance sur un objet qu'il instancie. Deux machines virtuelles sont donc nécessaires (une sur le serveur et une sur le client) et l'ensemble des communications se fait en Java. [40]

L'appel coté client d'une telle méthode est un peu plus compliqué que l'appel d'une méthode d'un objet local mais il reste simple. Il consiste à obtenir une référence sur l'objet distant puis à simplement appeler la méthode à partir de cette référence.

La technologie RMI se charge de rendre transparente la localisation de l'objet distant, son appel et le renvoi du résultat.

En fait, elle utilise deux classes particulières, le stub et le skeleton, qui doivent être générées avec l'outil rmic fourni avec le JDK.
Le stub est une classe qui se situe côté client et le skeleton est son homologue coté serveur. Ces deux classes se chargent d'assurer tous les mécanismes d'appel, de communication, d'exécution, de renvoi et de réception du résultat. [41]

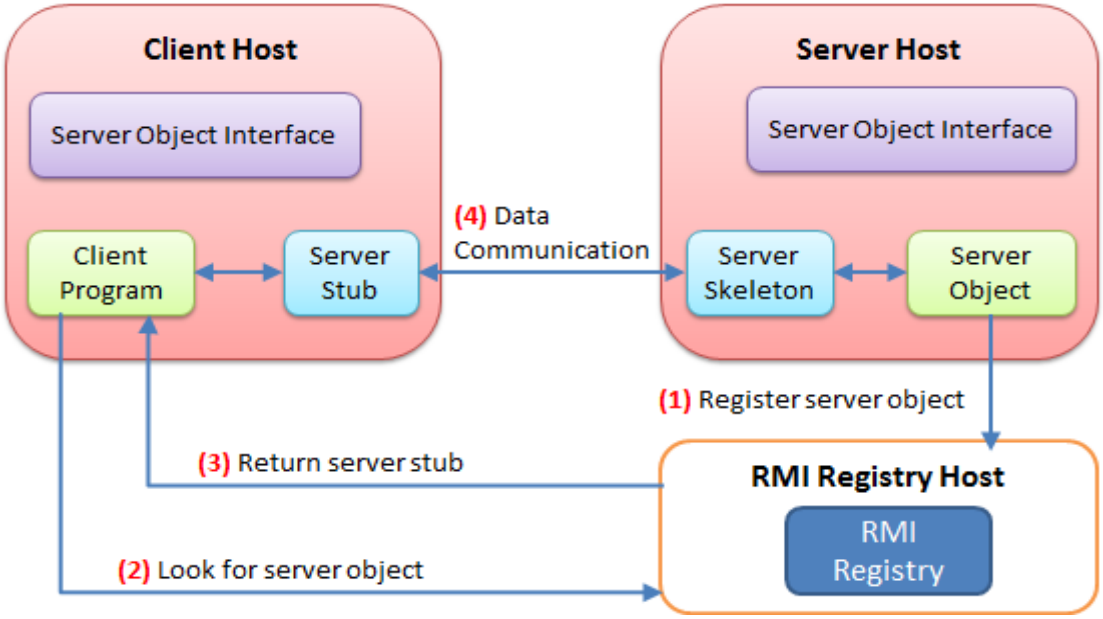


Figure III.3. Schéma expliquant l'architecture de la technologie RMI [42]

3. MISE EN MARCHE DE L'APPLICATION

Cette section décrit en détail l'application avec toutes ces fonctionnalités, de la création des profils à la génération des communautés.

3.1.Contexte des profiles

Après avoir lancé l'application cliente, qui sera connecté au serveur, la première fenêtre qui apparait est celle intitulé « Profile Context ».

Dans cette fenêtre l'utilisateur pourra créer, supprimer les profils utilisateurs et service ou afficher les informations des profils tel que présenté dans les figures suivantes :

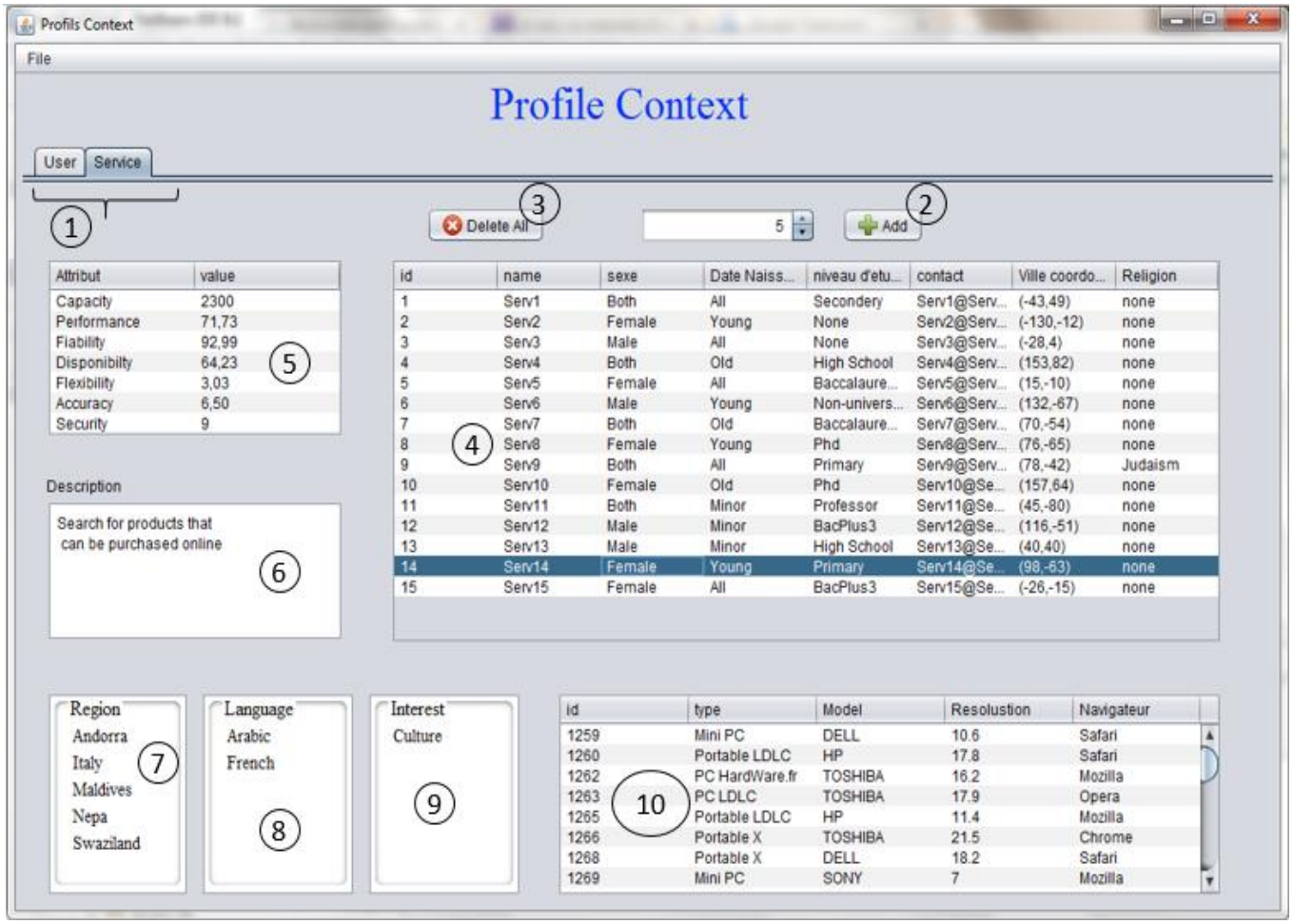


Figure III.4. Fenêtre « Profile Context (Service) »

1- Se déplacer entre les onglets Utilisateur et Service
2- Ajouter des services en spécifiant le nombre dans le champ à côté du bouton
3- Supprimer tous les services
4- Liste des services existant avec leurs informations
5- Valeur des attributs QoS (Quality of Service)
6- Description du service sélectionné
7- Région du service sélectionné
8- La langue du service sélectionné
9- Intérêt du service sélectionné
10- Extra informations (dispositif) sur le service sélectionné

La deuxième figure concerne le profil utilisateur :

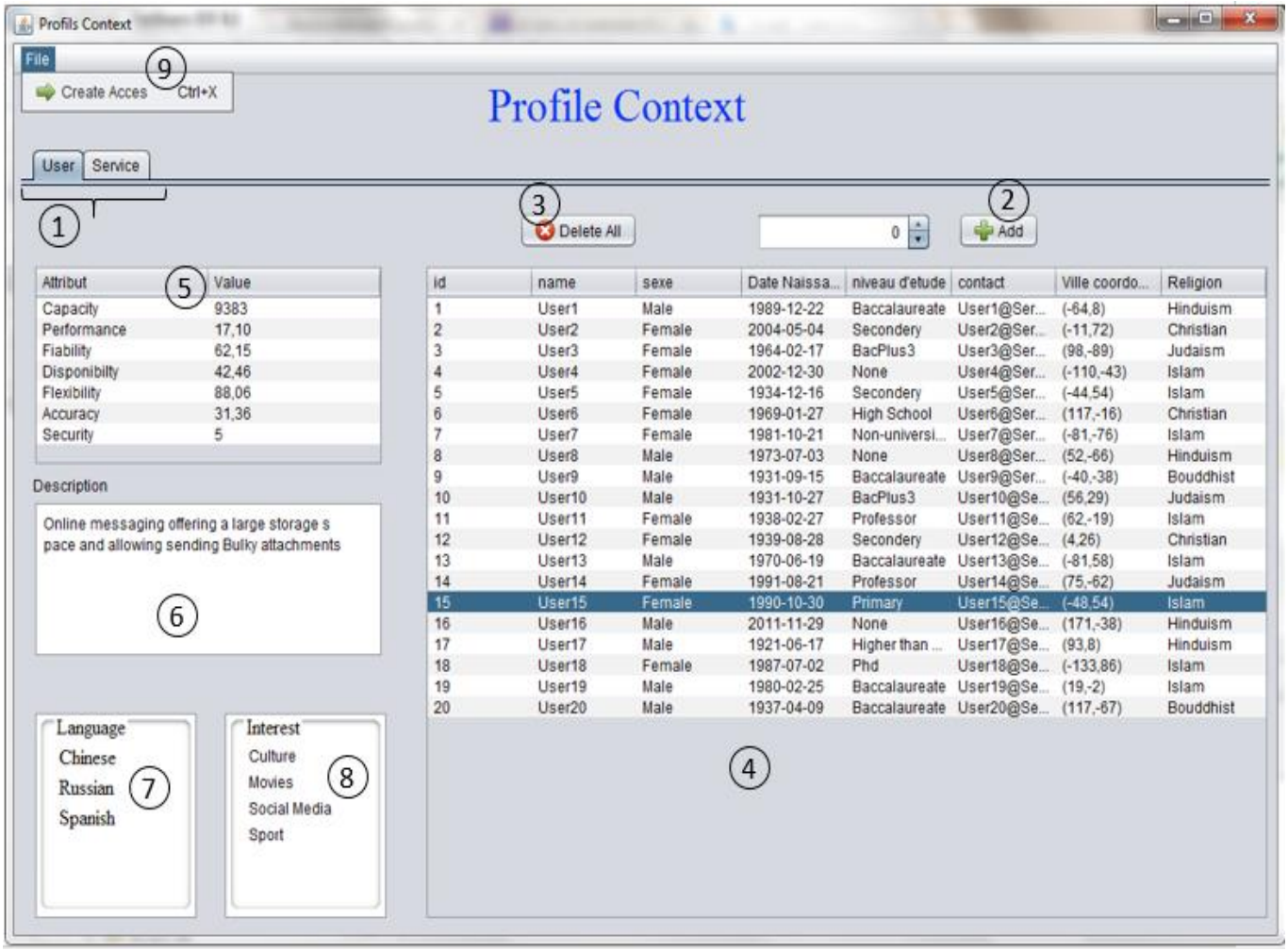


Figure III.5. Fenêtre « Profile Context (User) »

1- Se déplacer entre les onglets Utilisateur et Service
2- Ajouter des utilisateurs en spécifiant le nombre voulu
3- Supprimer tous les utilisateurs
4- Liste des services existant avec plusieurs informations ((nom, sexe, date de naissance, boite mail, coordonnée et religion)
5- Valeur des attributs QoS (Quality of Service)
6- Description de l'utilisateur sélectionné
7- La langue du service sélectionné
8- Intérêts souhaités de l'utilisateur sélectionné
9- Passer à la fenêtre de création des accès

3.2.Création des accès

Après avoir créé les profils, l'usager de l'application pourra passer à l'étape de création des accès, qui s'appuie sur le calcul de convenance et pourra ainsi donc créer le graphe des accès initial (graphe de convenance).

La figure ci-après représente la fenêtre de création des accès :

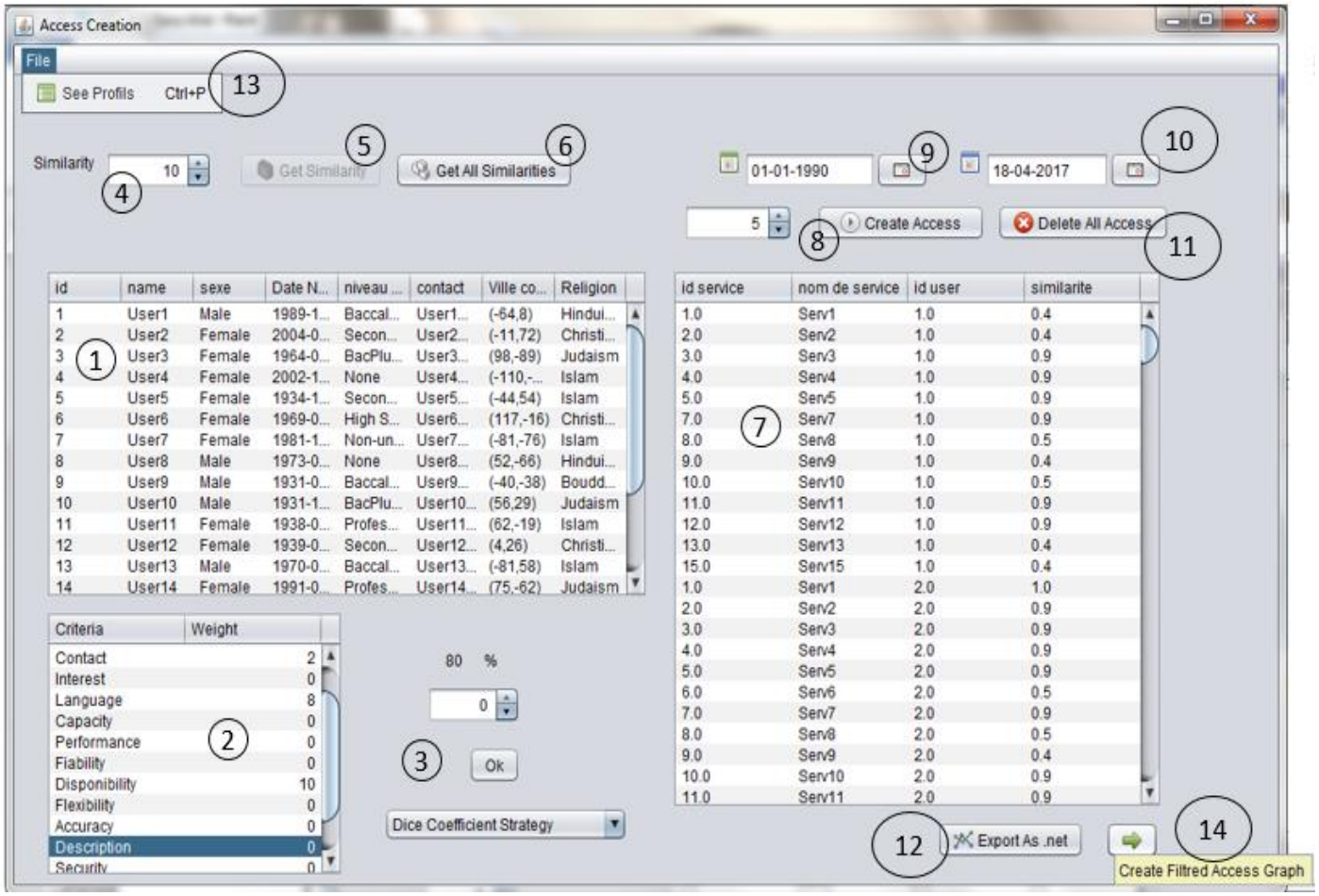


Figure III.6. Fenêtre « Access Creation »

1- Liste des utilisateurs existants
2- Critères entrant dans le calcul de convenance qui doivent être remplis
3- Panel permettant de saisir le poids des critères et aussi de consulter le pourcentage atteint
4- Champ pour saisir la convenance minimale
5- Calcul de la convenance de(s) utilisateur(s) sélectionné(s)
6- Calcul de la convenance de tous les utilisateurs existants
7- Liste des utilisateurs et les services similaires
8- Démarrage du processus de création des accès

9- Date début de création des accès
10- Date de fin de création des accès
11- Supprimer tous les accès
12- Exporter le graphe des accès sous format .net
13- Retourner à la fenêtre « Profile Context »
14- Passer à la phase de filtrage du graphe

Dans ce qui suit, les algorithmes utilisés dans le calcul de convenance et la création des accès que les structures de données utilisé :

Algorithme II.1. Algorithme Calcul de convenance

Input: Users: list <user>, Servcs: list <service>
Prmtrs_values: Map <String, Float>
Seuil : entier

Output : GraphConvenance (USER, SERVICE, Arc)

Debut

```
Pour chaque profil utilisateur faire :
    Lire (profil_User, Prmtrs_values) ;
    Pour chaque profil Service faire :
        Lire (profil_Service) ;
        Sim = Convenance (Profile_User, Profile_Service, Prmtrs_values) ;
            Si (Sim> Seuil) Alors
                Arc A= Arc (USERi, SERVICEi, Sim);
                GrapheSimi.EtabilirLien (USERi, SERVICEi, A) ;
            FinSi ;
    FinPour ;
FinPour ;
Retourner GraphSimi ;
```

FIN.

Structure d'un Arc :

Type Arc = enregistrement

Extrémity1 : User

Extrémity2 : Service

Convenance : réel

FinEnregistrement

Le deuxième algorithme est celui de la création des accès :

Structure d'un Arc (Graphe d'accés) :

Type ArcAcc = enregistrement

Extrémity1 : User

Extrémity2 : Service

Nbr_access : entier

Convenance : réel

FinEnregistrement

Algorithme II.2. Algorithme Génération des accès

Algorithme Génération des accés

Input : GraphSimi : Graph

Seuil, max_access, nbr_acces : entier

Output : GraphAccess

Début

Pour chaque Arc in GraphSimi **faire** :

Si (Arc.getSimilarity () >= Seuil) **alors** :

Nbr_access:= Random ([0, max_access]);

GraphAccess.EtablirLien (Arc.getExtremity1 (), Arc.getExtremity2(), nbr_access, Arc.Convenance);

FinSi;

FinPour;

FIN.

3.3.Filtrage du graphe des accès et génération de la communauté

L'étape suivante avant la génération de communautés est celle du filtrage.

A ce stade l'usager pourra éliminer les accès qui ne comblent pas les filtres expliqué dans le chapitre précédent.

En premier lieu, le décideur choisira le nombre d'accès minimum entre l'utilisateur et le service ainsi que la convenance minimale entre eux afin de prendre réellement le lien entre ces sommets (voir chapitre II – section 3.3), ensuite il doit choisir les filtres présentés dans la figure ci-dessous qui est décomposée en 5 onglets chacun représentant un filtre, avant de créer le graphe d'accès filtré.

En premier on présente l'algorithme de génération du graphe d'accès filtrés :

Dans ce qui suit nous allons détaillés les cinq onglets des filtres :

3.3.1. Onglet intérêts :

Le premier onglet présenté est « Intérêts » :

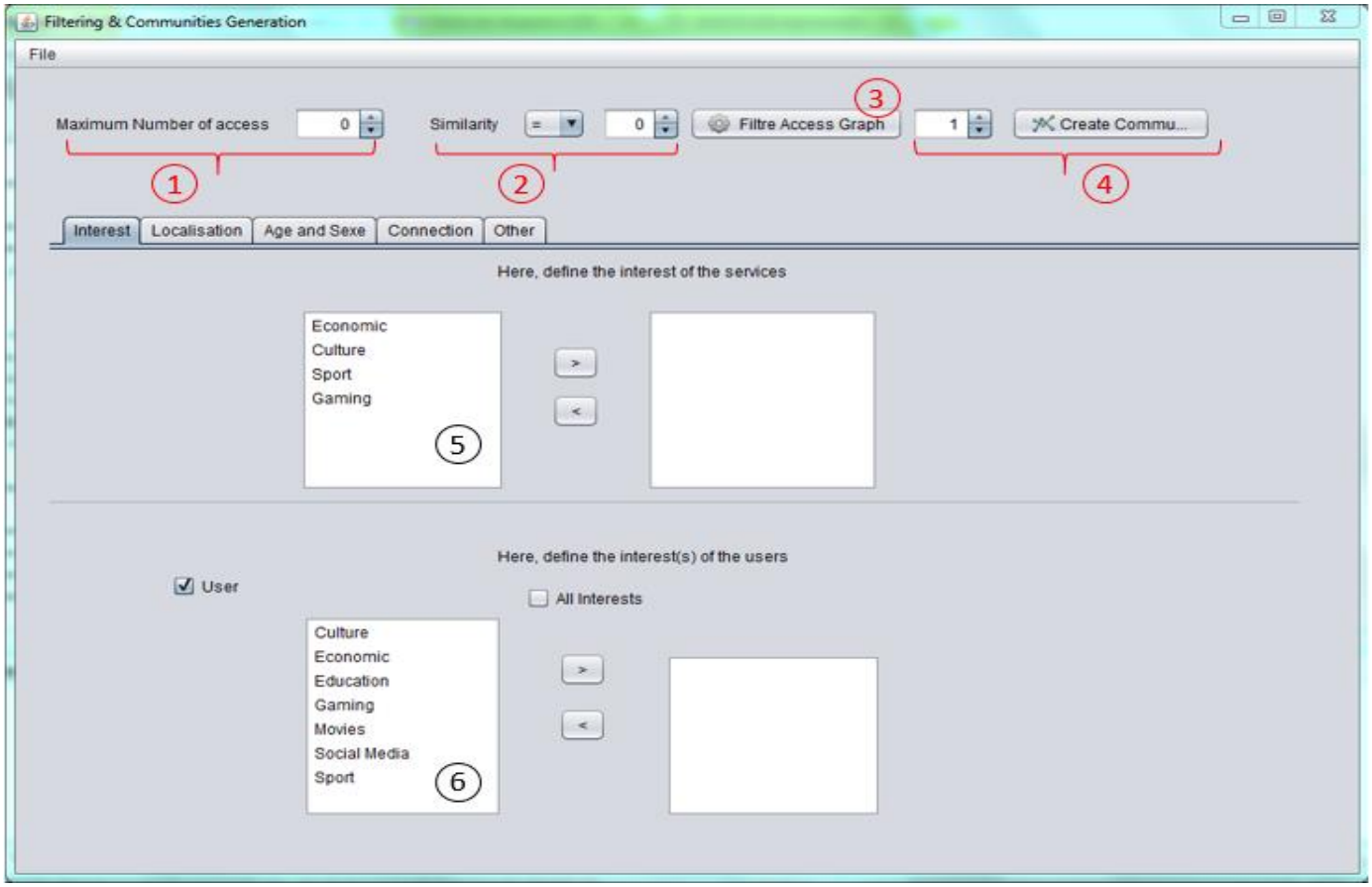


Figure III.7. Onglet filtre intérêt

1- Champ pour remplir le nombre minimum des accès
2- Saisir la convenance et l'opérateur logique
3- Lancer le processus de filtrage
4- Lancer le processus de génération de communauté après avoir saisi le degré
5- Sélectionner l'intérêt du service en le faisant passer de l'autre coté
6- Sélectionner l'intérêt de l'utilisateur en le faisant passer de l'autre coté

3.3.2. Onglet Localisation

Les deux premières figures sont relatives à la localisation du service :

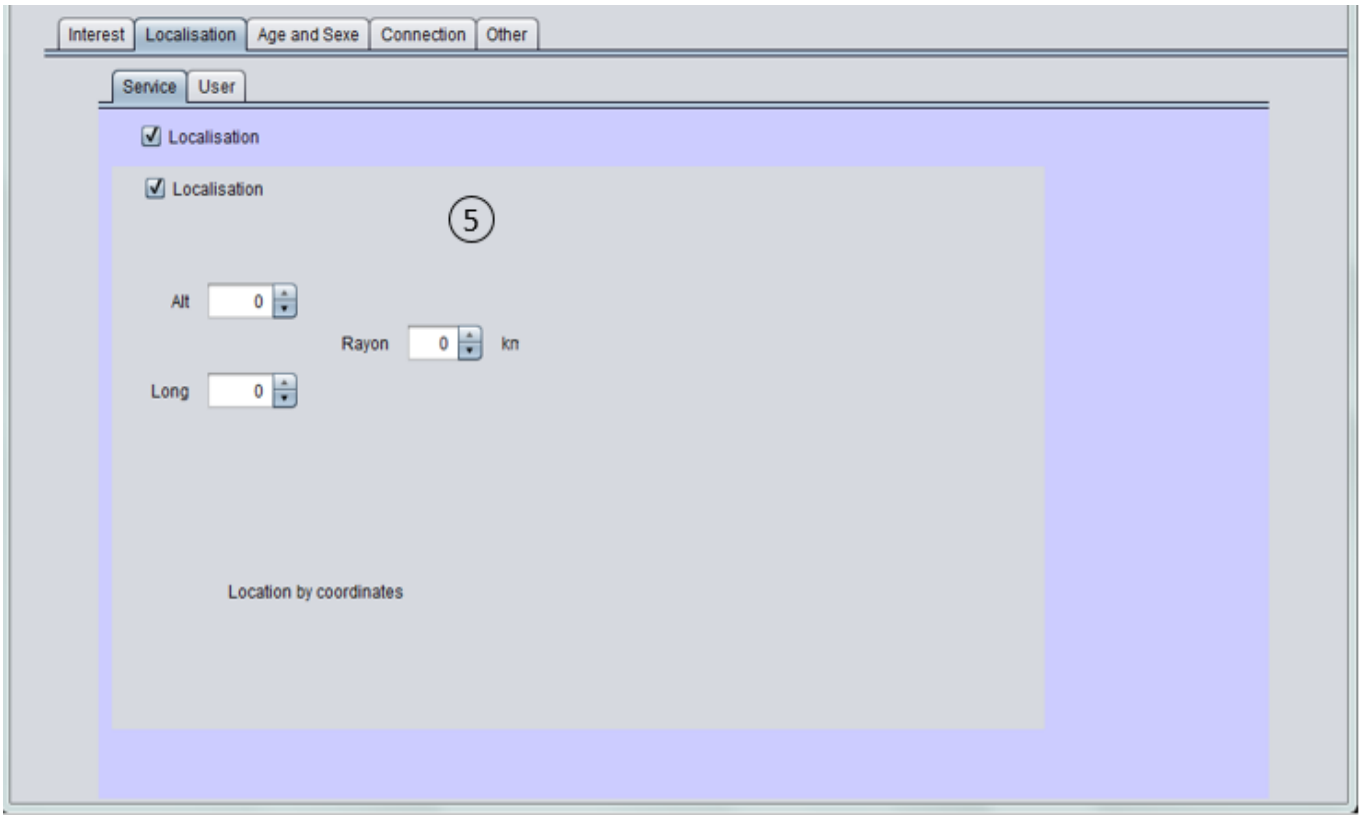


Figure III.8. Onglet filtre localisation_1 (Service)

5- Spécifier la localisation du service par coordonnées (Latitude et Longitude) et le rayon en kilomètre

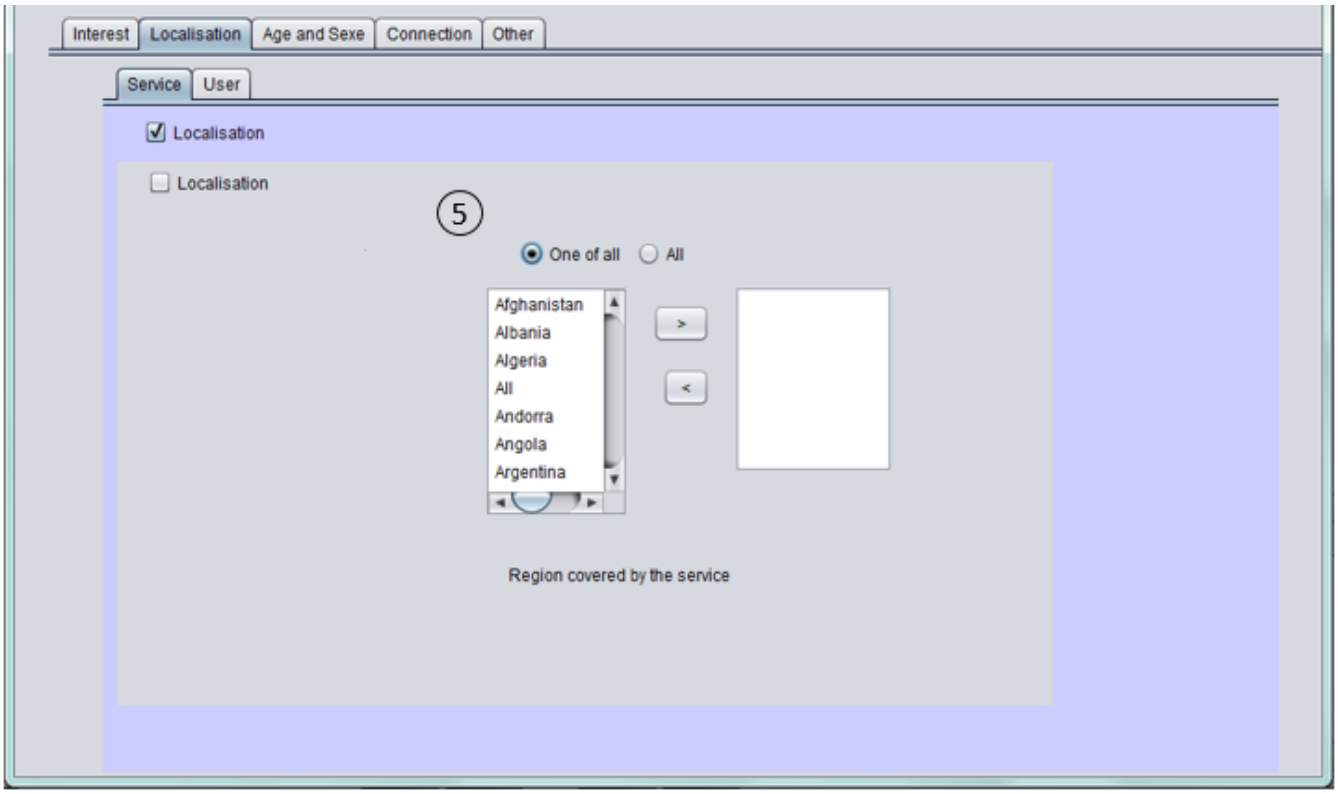


Figure III.9. Onglet filtre localisation_2 (Service)

5- Choisir la ou les régions couvertes par le service

Les trois figures qui suivent correspondent à la localisation de l'utilisateur :

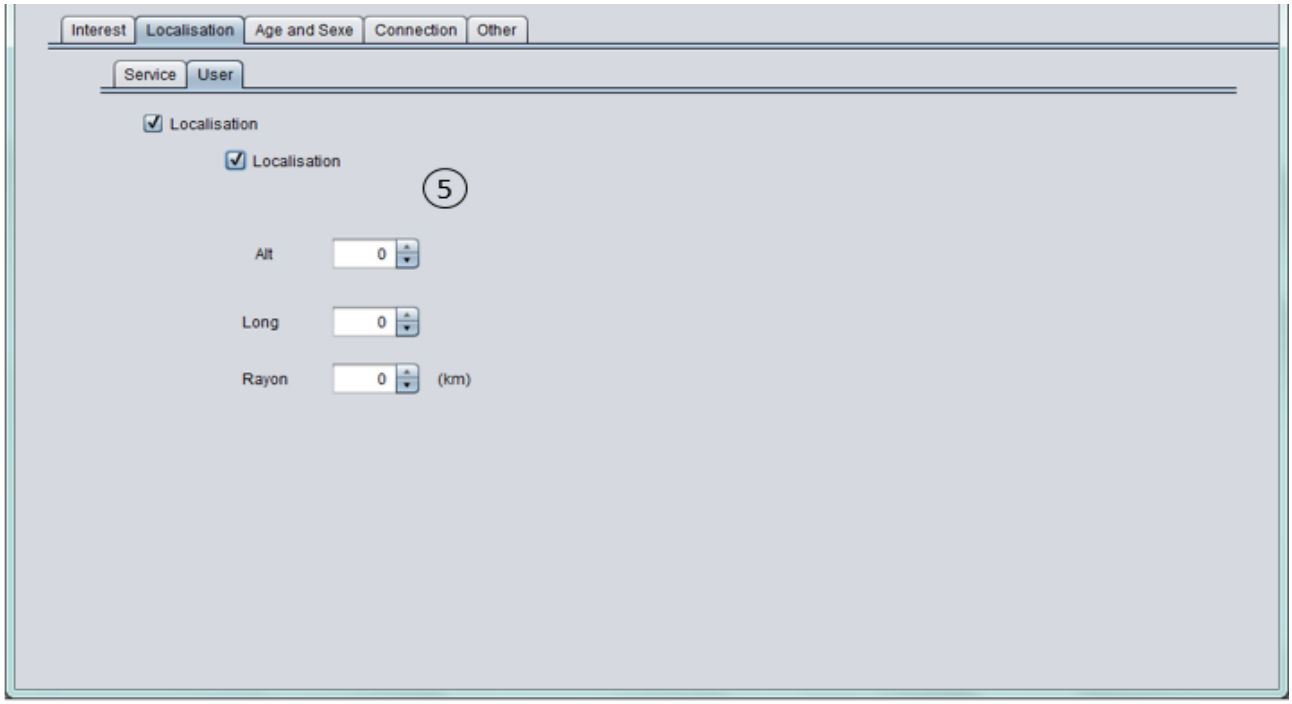


Figure III.10. Onglet filtre localisation_1 (Utilisateur)

5- Saisir les coordonnées de l'utilisateur

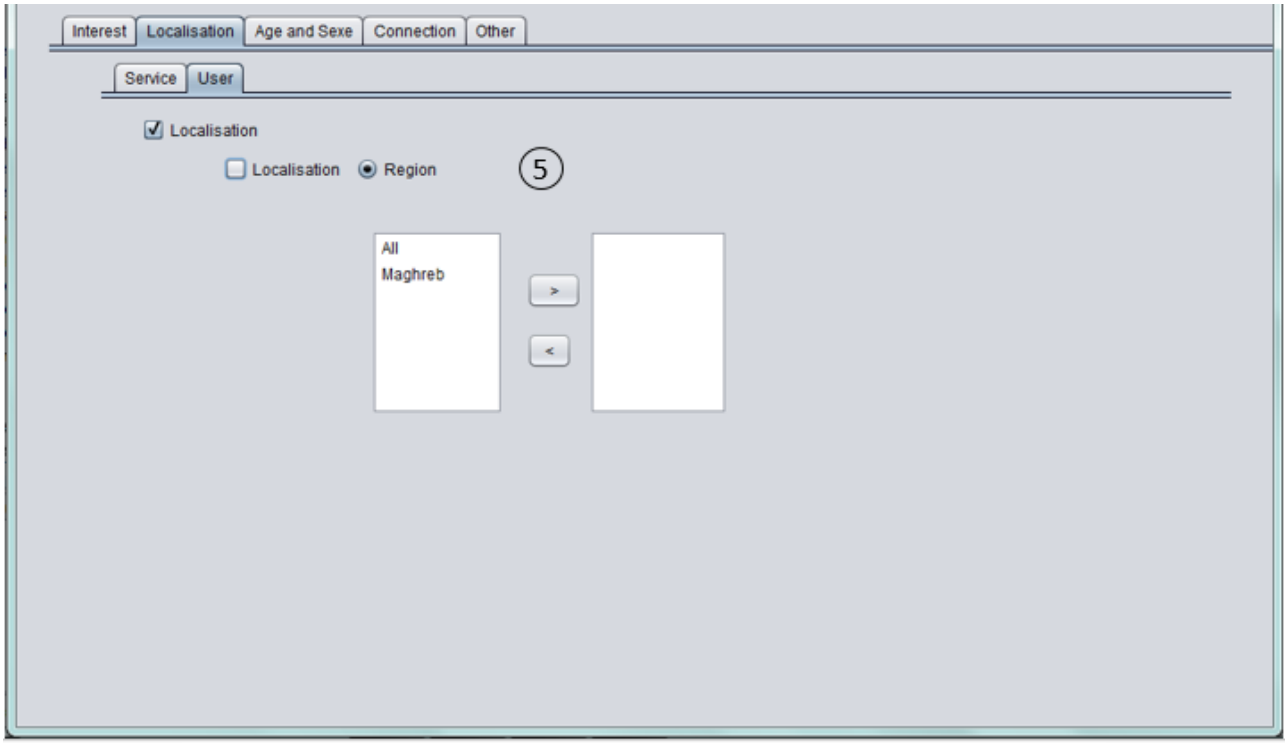


Figure III.11. Onglet filtre localisation_2 (Utilisateur)

5- Choisir la région de l'utilisateur en la faisant passer de l'autre coté

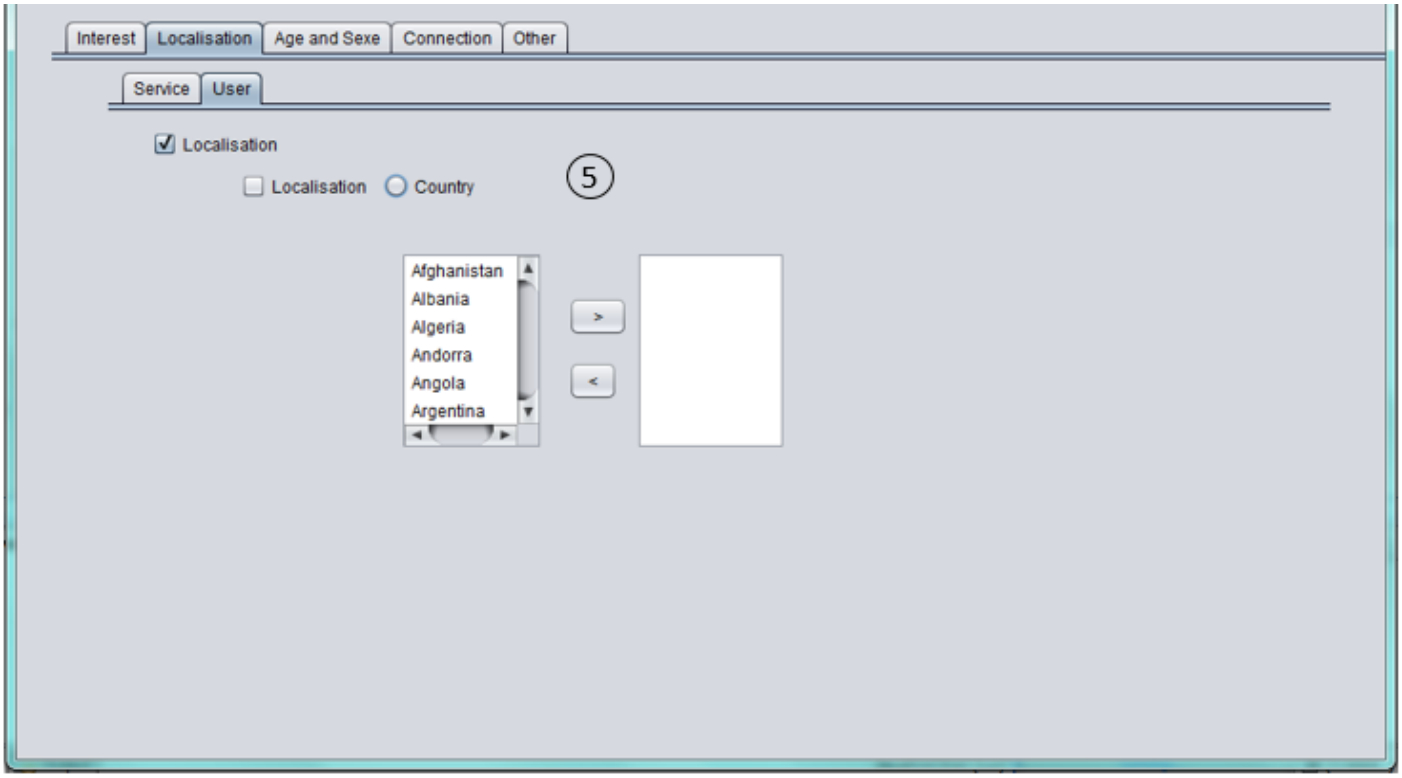


Figure III.12. Onglet filtre localisation_3 (Utilisateur)

5- Choisir le pays de l'utilisateur

3.3.3. Onglet Age & Sexe

Ici l'usager choisi l'âge de l'utilisateur soit par une seule valeur ou par intervalle :

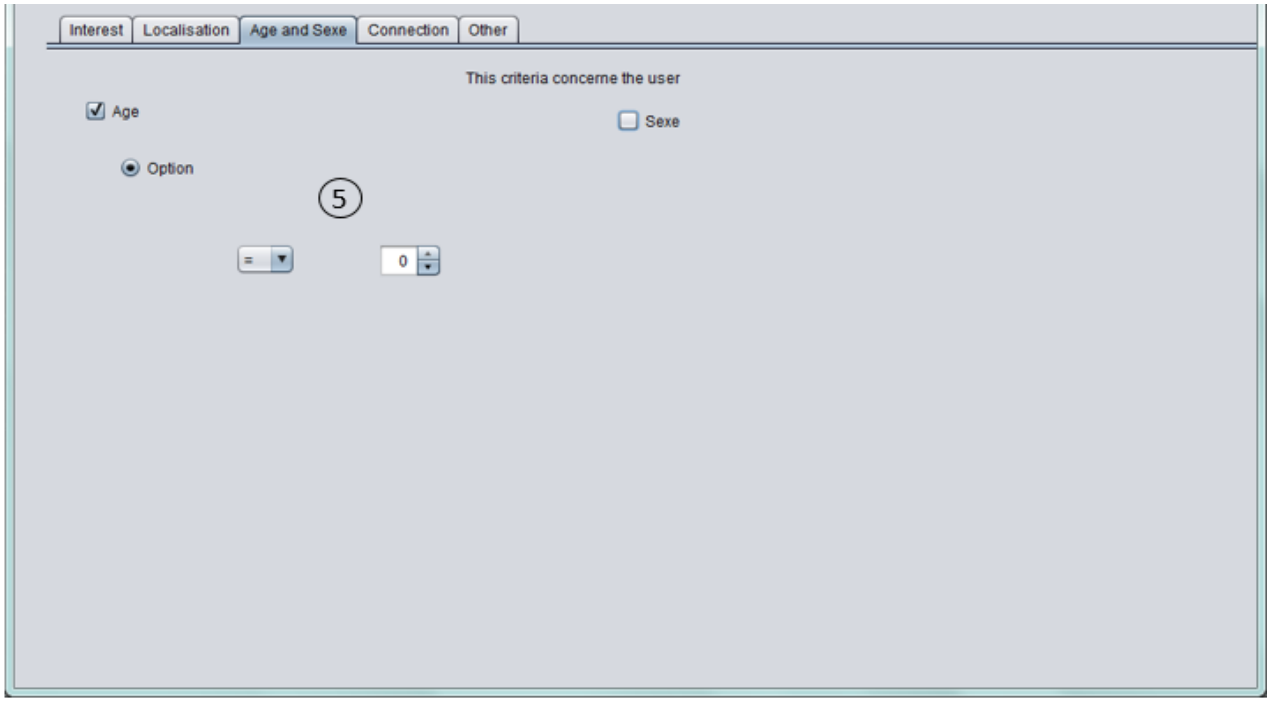


Figure III.13. Onglet filtre Age et sexe (Age-option 1)

5- Saisir l'âge de l'utilisateur ainsi que l'opérateur logique

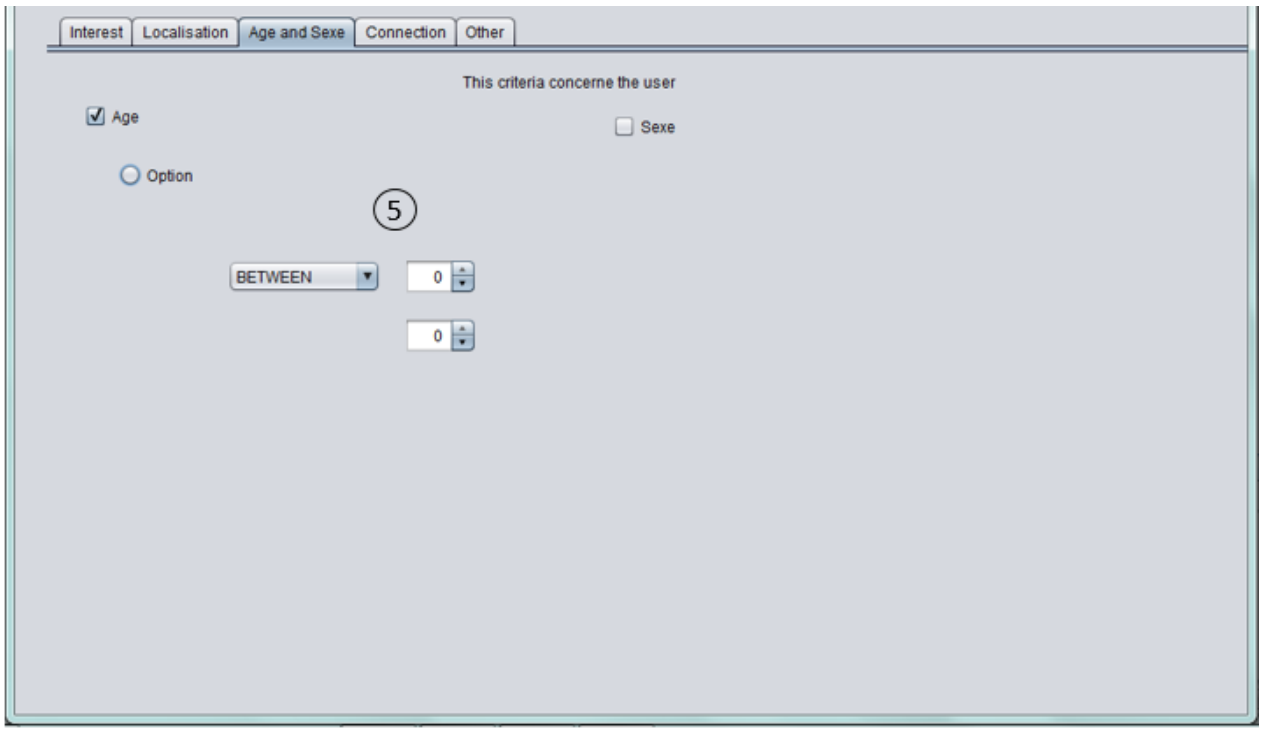


Figure III.14. Onglet filtre Age et Sexe (Age-option 2)

5- Définir l'intervalle d'âge

La figure suivante schématise le filtre sexe :

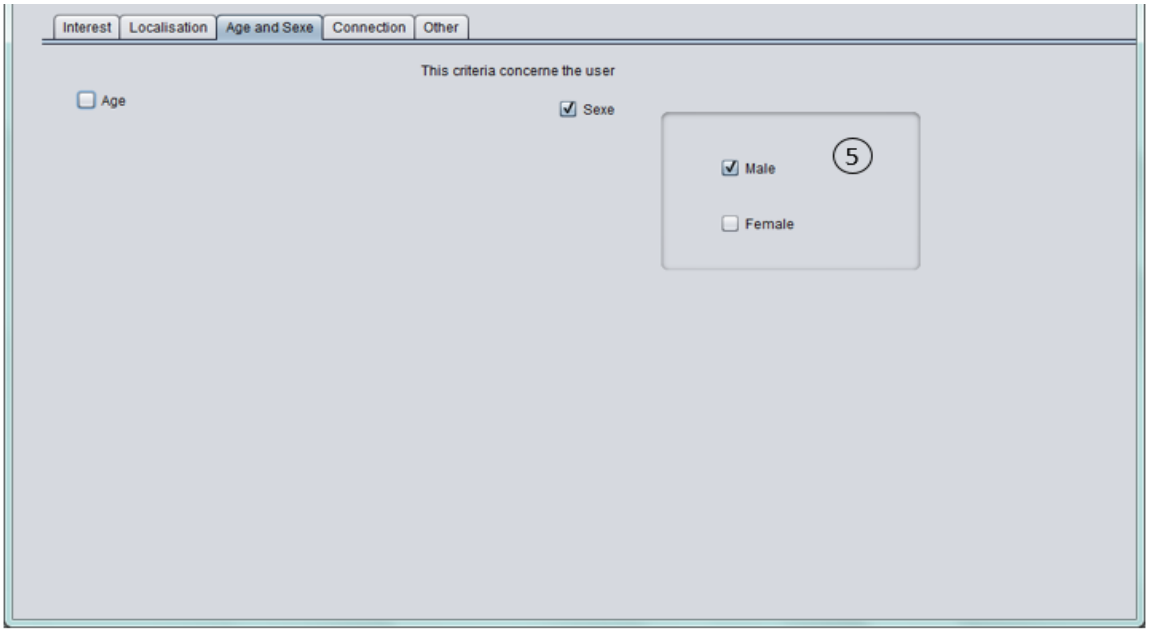


Figure III.15. Onglet filtre Age et Sexe (Sexe)

5- Cocher l'une des cases pour choisir homme ou femme, ou décocher le filtre pour cibler les deux sexes

3.3.4. Onglet Connection

Ce filtre concerne les connections, il peut être établie soit par date ou par durée, soit par valeur fixe ou par intervalle, l'usager n'a qu'à choisir la ou les valeurs désirés, et donc ne conserver que les accès respectant le temps déterminé.

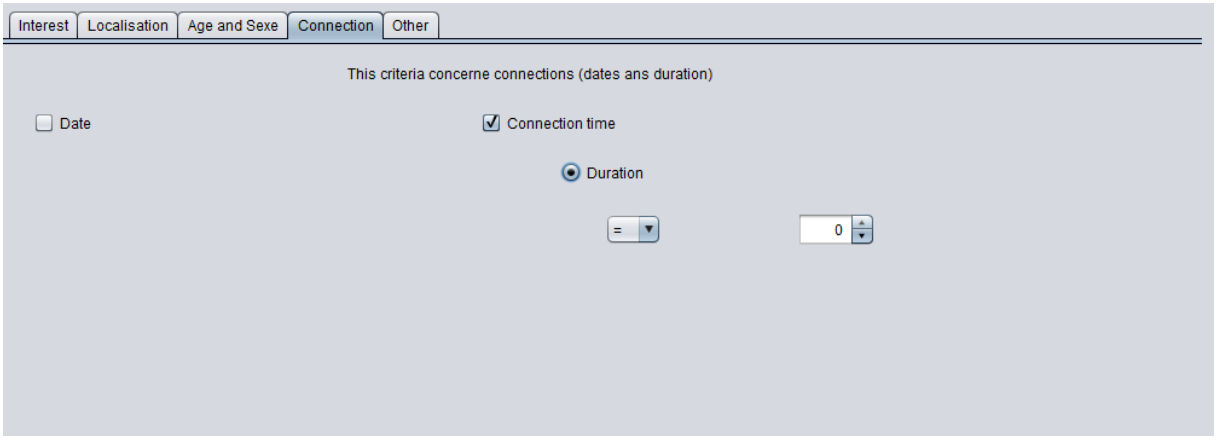


Figure III.16. Onglet filtre Connection_1

5- Saisir la date et l'opérateur logique

6- Saisir la durée et l'opérateur logique

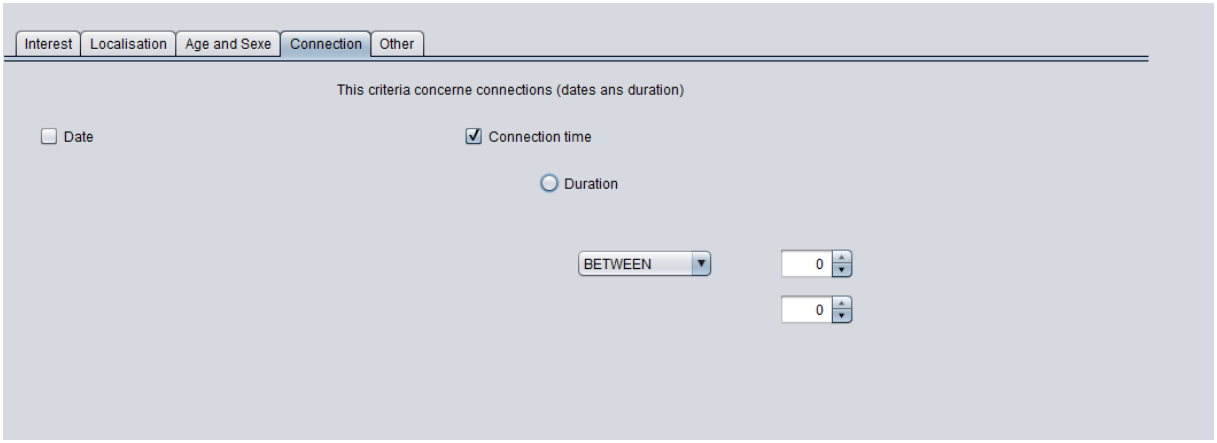


Figure III.17. Onglet filtre Connection_2

5- Choisir l'intervalle de dates

6- Choisir un intervalle de valeurs de durées

3.3.5. Onglet autres filtres

Ce dernier onglet regroupe trois filtres : la religion, la langue et le niveau d'étude, l'usager doit choisir une ou plusieurs valeurs depuis les listes fournis.

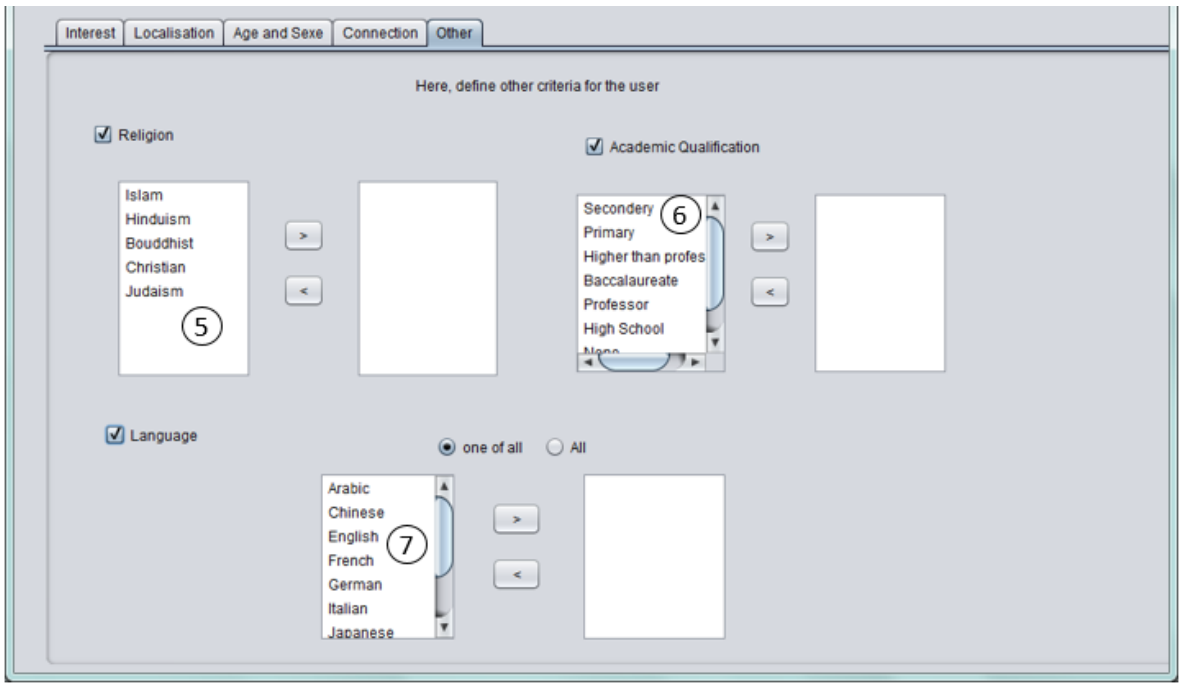


Figure III.18. Onglet Autre Filtres

5- Choisir la religion des utilisateurs

6- Choisir le niveau d'étude des utilisateurs

7-Choisir la/les langue(s) maitrisé(s) par l'utilisateur

Dans ce qui suit, l'algorithme de génération du graphe d'accès filtrés :

Algorithme II.3. Algorithme Génération du graphe d'accès filtrés

```
Input : Nbr_access_min, seuil_sim : entier,
        GraphAcc : Graph<User, Service, ArcAcc >
        FiltreUser, FiltreService, FiltreAccess: Map<String, Object>;
Output : GraphAccFiltré : Graphe ;
Debut
    listServices = créer_listeServices_vide () ;
    listUsers = créer_listeUsers_vide () ;
    Pour chaque ArcAcc dans GraphAcc faire :
        Si Satisfait (ArcAcc.Extremity2, FiltreService) alors :
            Ajouter (listServices, ArcAcc.Extremity2) ;
        Fin Si ;
        Si Satisfait (ArcAcc.Extremity1, FiltreUser) alors :
            Ajouter (listUsers, ArcAcc.Extremity1) ;
        Fin Si ;
    FinPour ;
    Graphe GrapheAccFiltré = créer_graphe_vide () ;
    Pour chaque Service dans listServices faire :
        Pour chaque User dans listUsers faire :
            Si (Convenance (User,Service) >= Seuil && nbrAcees(User,Service)
            >= Nbr_access) alors :
                EtablirLien (Service, User, GrapheAccFiltré) ;
            Fin Si ;
        Fin pour ;
    Fin Pour ;
    Retourner GraphAccFiltré ;
FIN.
```

3.4.GENERATION DES COMMUNAUTES

La dernière étape représente le processus de génération des communautés qui sera lancé en cliquant sur le bouton correspondant en choisissant le degré de communauté désiré :

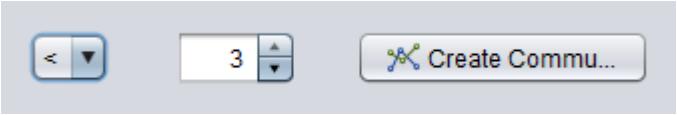


Figure III.19. Lancer la génération de communautés

Le dernier algorithme, celui de la génération de communautés, est présenté ci-dessous ainsi que la structure de donnée représentant une communauté :

Structure d'une communauté :

Type Communauté = enregistrement

id : entier,

listeUsers : ListeUsers,

listeServices : ListeServices ;

FinEnregistrement

Algorithme II.4. Algorithme Génération des communautés ;

```
Input : GraphAccFiltré : Graphe,
        Degré : entier ;
Output : listeCommunauté : ListeCommuanuté ;
Début
listeCommunauté = créer_liste_communauté_vide() ;
    Pour chaque Arc dans GraphAccFiltré faire :
        Service S := Arc.Extremity2 ;
        Commuanuté Com = créer_communauté() ;
        AddService(Com, S) ;
        Si Exist(Com, listeCommunauté) alors :
            Int ind := Indice(Com, listeCommunauté) ;
            Ajouter (ind, Arc.Extremity1, listeCommunauté) ;
        Sinon
            Com.addUser(Arc.Extremity1) ;
            Ajouter (Com, listeCommunauté) ;
    Fin pour ;
ListeComId = Permutations (len(listeCommunauté), degré);
Pour i de 1 à ListeComId.size () faire
    Communauté com = listeCommunauté.get(i) ;
    ListSerCom= com.getService ()
    Si (ListSerCom size () != 1) alors :
        Sommet= ListSerCom sommet () ;
        Liste<User> l= Sommet.getUser()) ;
        Pour j de 2 à ListSerCom. getService ().size () faire :
            L=intersection (L, listeCommunauté.get (ListeComId.get(j)) .getUser()) ;
        Fin Pour ;
        Communauté c = créer_communauté (ListSerCom, L) ;
        Enfiler(c, listeCommunauté) ;
    Fin Si ;
Fin Pour ;
FIN.
```

Notons que visualiser les différents graphes résultants, un fichier est généré dans chaque étape qui peut être exploité dans le logiciel de visualisation des graphes afin d'afficher les résultats.

4. EXEMPLE D'USAGE

Afin de tester l'application on a va essayer un exemple avec 50 service et 1000 utilisateurs créés aléatoirement, et avec un taux de convenance est égale à 50%.

Après la création des profils on passe à la création des accès mais avant il faut calculer la convenance en appuyant sur le bouton « Get All Similarities » après avoir spécifié les poids des critères intervenant dans le calcul.

Le processus de création des accès se déclenchera lorsqu'on clique sur le bouton « Generate Access », le nombre maximum d'accès est égale à 5.

Un nombre d'accès sera créés aléatoirement, dans notre exemple la figure qui suit démontre le résultat obtenue :

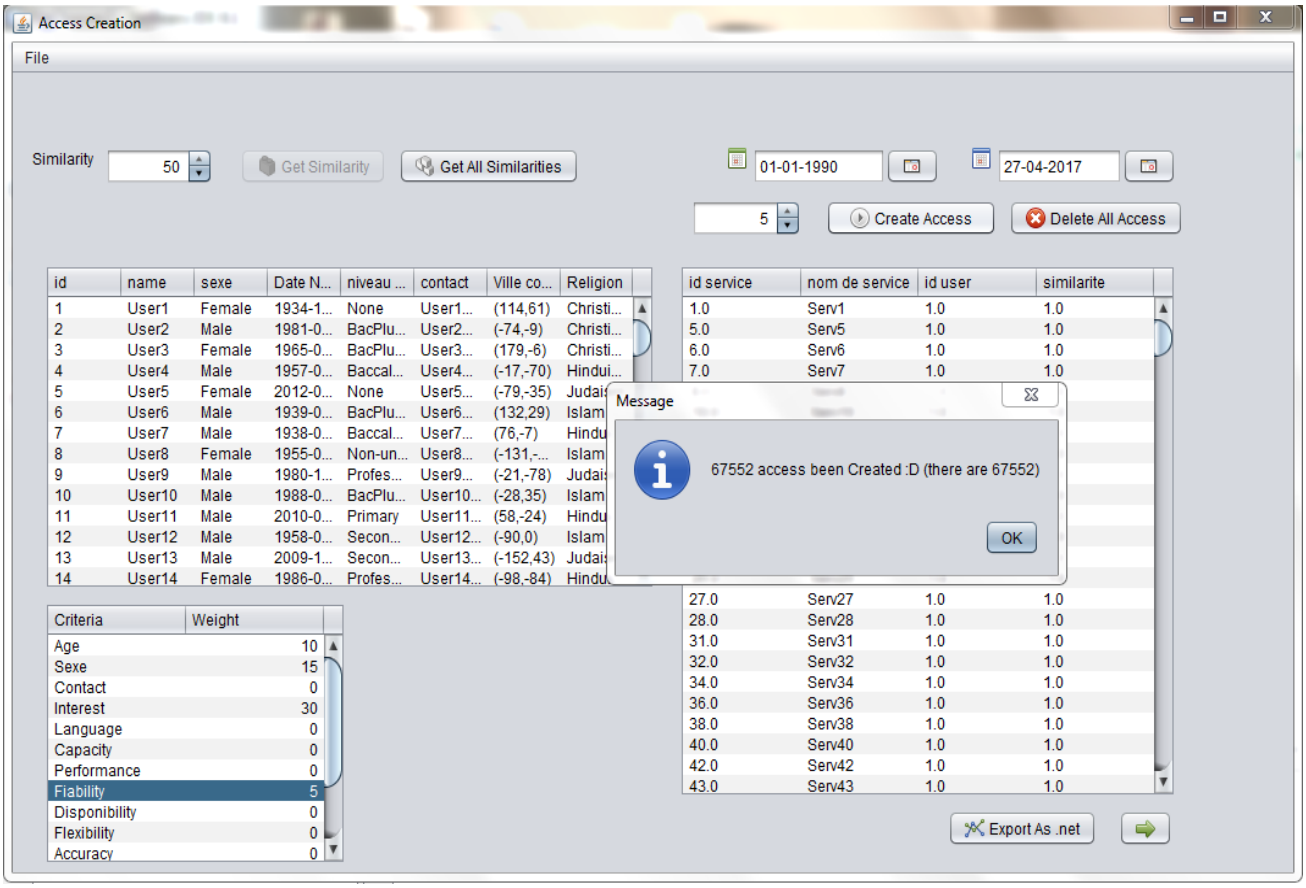


Figure III.20. Message de nombre d'accès créés

L'étape suivante est la génération du graphe d'accès filtré. Pour cela on a choisi des filtres tels que les membres de la communauté sont :

- Les intérêts des services sont « Social Media », « Gaming » et « Sport »
- Les membres des communautés sont des hommes
- La durée de connexion est supérieure à 110
- La valeur de convenance est supérieure à 60%

En cliquant sur le bouton «Filter access graph », la BDD est filtré d'après les critères choisi.

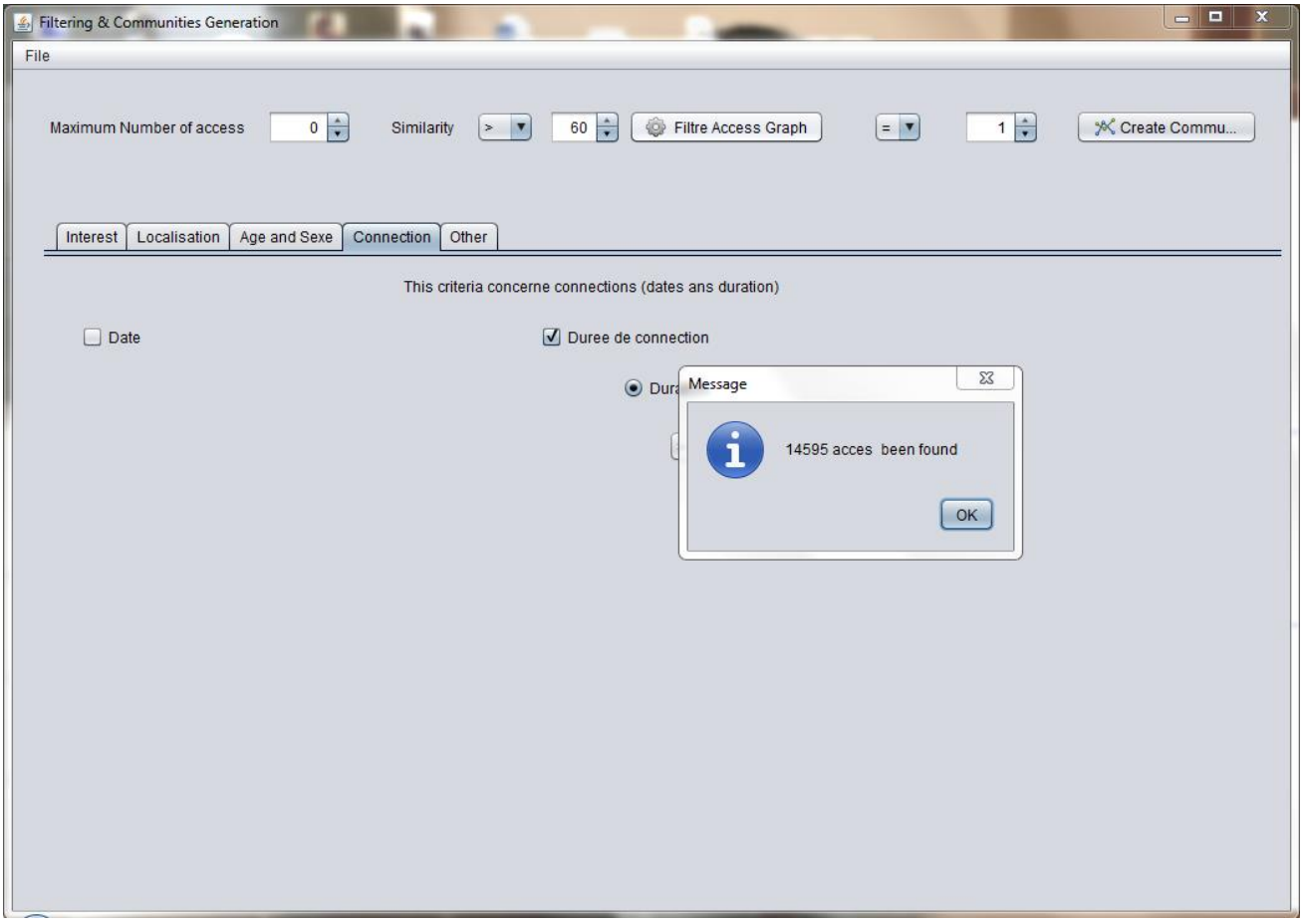


Figure III.21. Message nombre d'accès trouvés

En vue de visionner le graphe des accès créé, il faut cliquer sur le bouton « Export as .Net » et choisir le chemin dont le fichier va être enregistré. L'étape suivante est le lancement du logiciel Pajek et ouvrir le fichier au format .net enregistré précédemment :

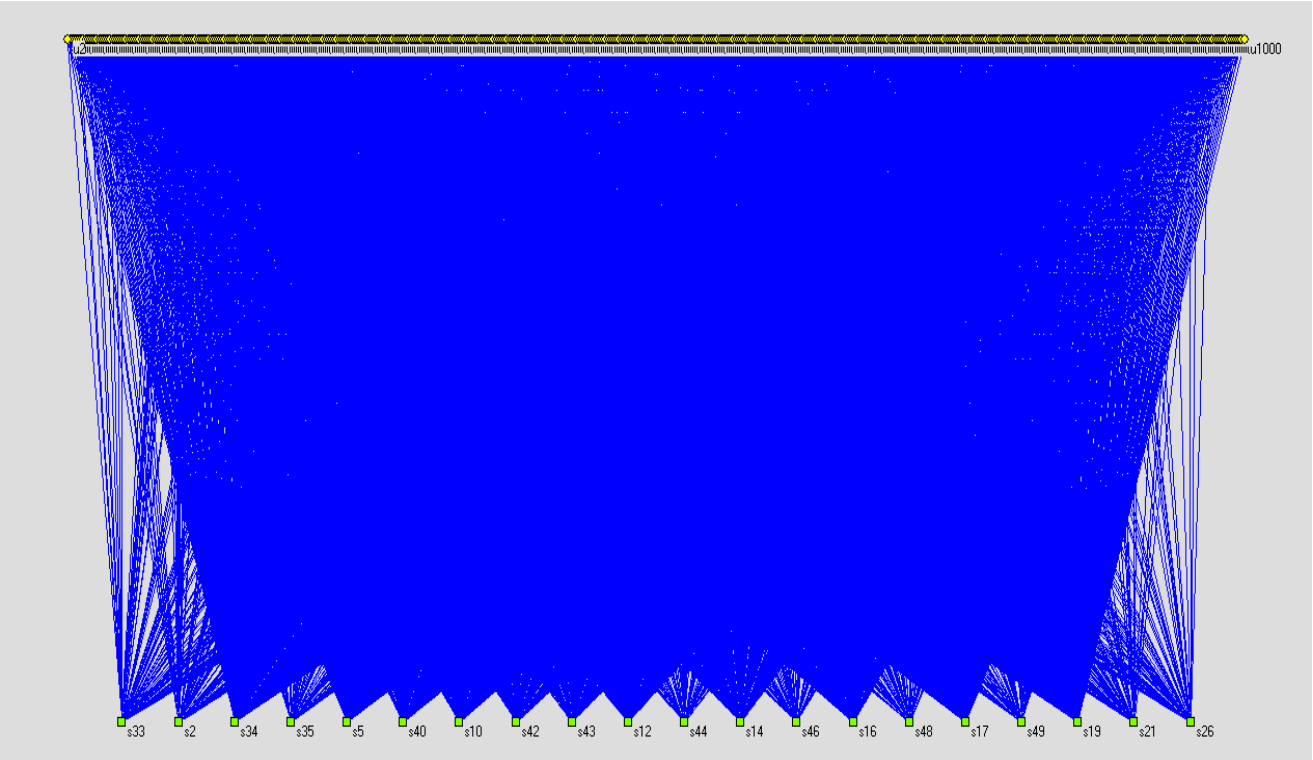


Figure III.22. Visualisation du graphe de l'exemple

Le degré de communautés saisi est égal à 1, la visualisation dans Pajek a donné le résultat ci-après :

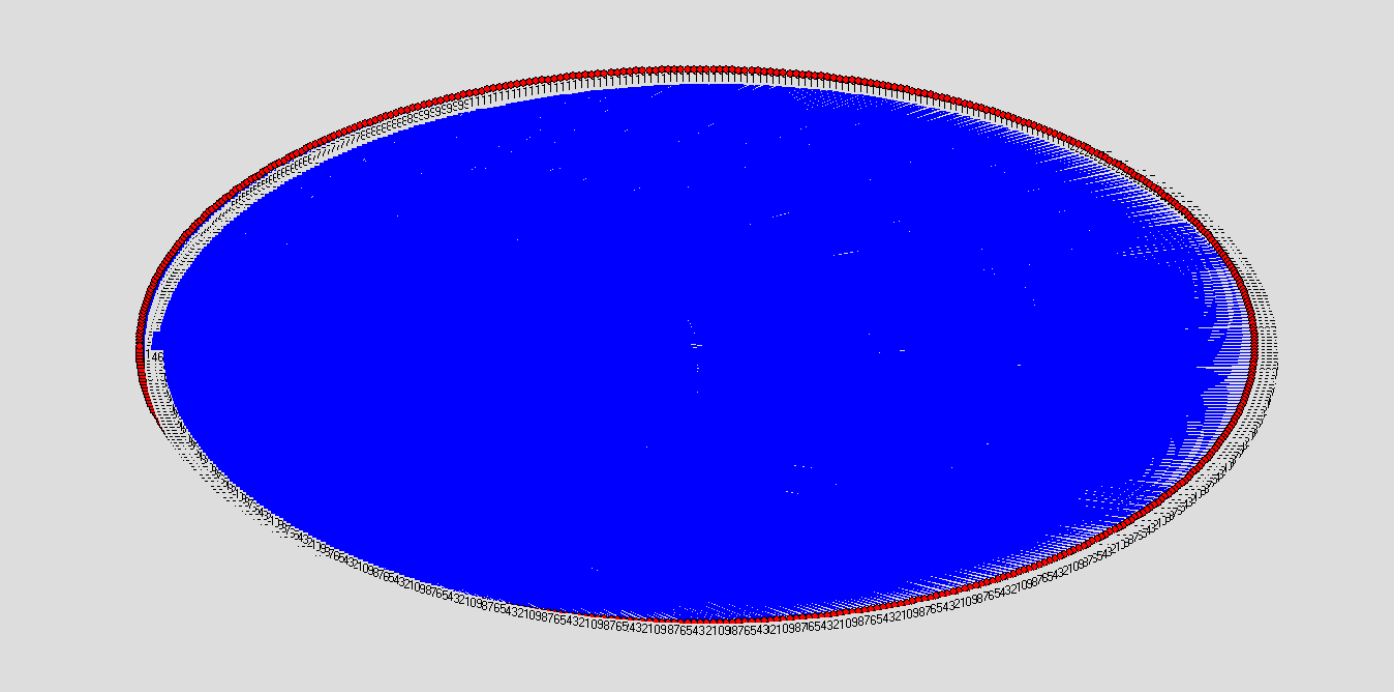


Figure III.23. Visualisation du graphe des communautés

Les utilisateurs sont représentés par des cercles rouges et le service par des carrés rouges, l'illustration suivante est plus détaillée :

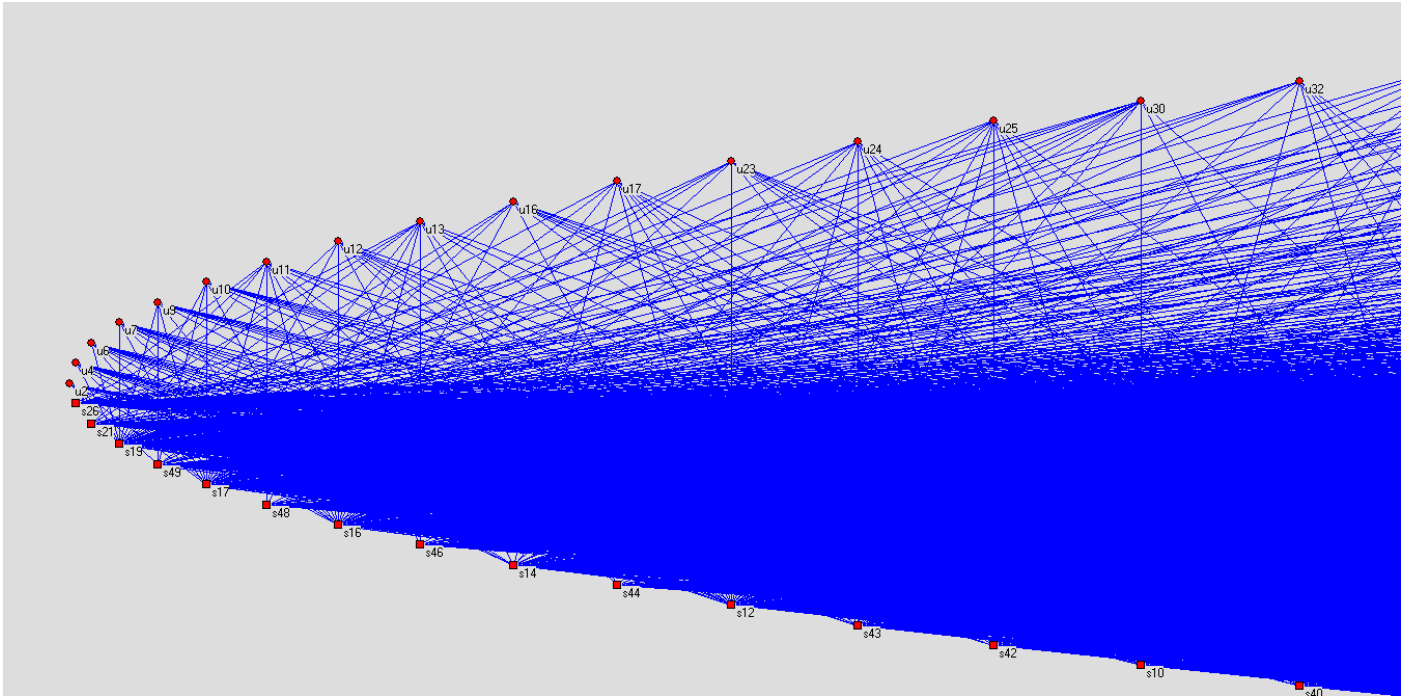


Figure III.24. Zoom sur le graphe précédant

5. ETUDE COMPARATIVE

Afin d'évaluer les performances de notre approche, on a établi une comparaison en temps d'exécution lors de la génération des communautés de cette dernière avec l'approche sans critère. L'étude a était réalisée avec une série d'expérimentation sur un échantillon de profils créé aléatoirement en utilisant une machine dotée d'un processeur Intel Core i5 et d'une RAM de 4 Go.

Pour le calcul de convenance en a sélectionné les attributs suivants avec leur coefficient :

- Age, Intérêt, Capacité, Langage, Performance : 10 %
- Description : Méthode de Jaro, 9 %
- Religion : 13 %

En ce qui concerne les filtres, on a choisi les suivants :

- Intérêt : Gaming, Social Media
- Sexe : Hommes

- Age : entre 13 - 40 ans
- Durée de connexion : 110 – 15000 milli secondes
- Niveau scolaire : n'importe lequel
- Convenance >= 20

Pour ce qui est de l'approche sans critère, on n'en a saisi aucun et on a choisi tous les intérêts concernant les services web.

Les tests ainsi que les résultats sont montrés dans le tableau et le graphe qui suivent :

	Découverte multi Critères basée services		Découverte Sans Critères	
N°essai	[Nombre utilisateurs, Nombre services]	Temps d'exécution (Milliseconde)	[Nombre utilisateurs, Nombre services]	Temps d'exécution (Milliseconde)
1	[100, 10]	0	[100, 10]	0
2	[250, 25]	0	[250, 25]	62
3	[500, 50]	0	[500, 50]	71
4	[750,75]	31	[750,75]	499
5	[1000, 100]	47	[1000, 100]	1279

Tableau III.1. Tableau Comparatif de variation de temps d'exécution en fonction du nombre d'utilisateurs et de services.

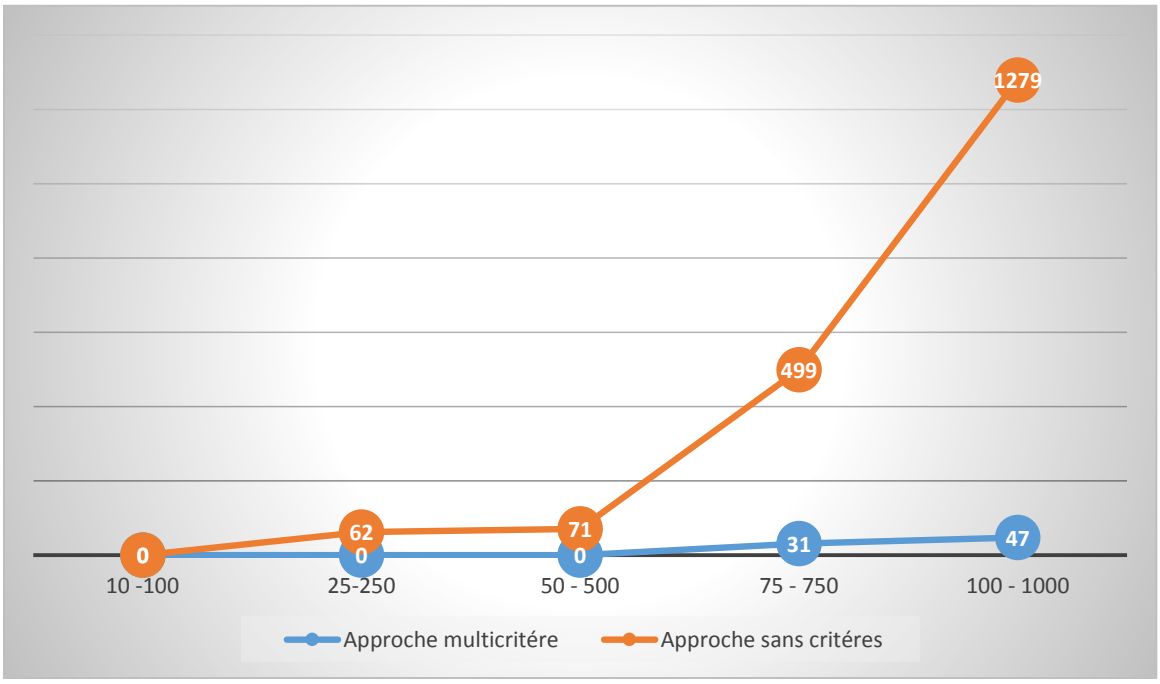


Figure III.25. Variation du temps d'exécution en fonctions du nombre d'utilisateurs et de services

On remarque, d'après la courbe ci-dessus, que le temps d'exécution augmente dès que le nombre de profils augmente.

Ainsi on constate qu'il existe une variation entre le temps d'exécution et le nombre de profils ; mais cette variation est différente entre les deux approches.

On remarque que la variation est grande dans l'approche sans critère par rapport à notre approche, ce qui montre la robustesse de notre approche dans le traitement de données et surtout dans les réseaux complexe.

Une autre comparaison effectuée est celle de notre méthode avec l'algorithme de Louvain [Chapitre I section 6.1.1] est implémenté dans le logiciel Pajek-XXL.

Les poids des coefficients ainsi que les filtres sont les même que ceux cité dans la 1ére comparaison, et l'étude concernera le temps d'exécution.

Les résultats obtenus sont présenté dans le tableau et la figure qui suivent :

	Découverte multi Critères basée services		Algorithme Louvain	
N°essai	[Nombre utilisateurs, Nombre services]	Temps d'exécution (Milliseconde)	[Nombre utilisateurs, Nombre services]	Temps d'exécution (Milliseconde)
1	[100, 10]	0	[100, 10]	0
2	[250, 25]	0	[250, 25]	0
3	[500, 50]	0	[500, 50]	0
4	[750,75]	31	[750,75]	16
5	[1000, 100]	47	[1000, 100]	34
6	[2000, 200]	1000	[2000, 200]	1013

Tableau III.2. Tableau Comparatif de variation de temps d'exécution en fonction du nombre d'utilisateurs et de services (Algorithme Louvain).

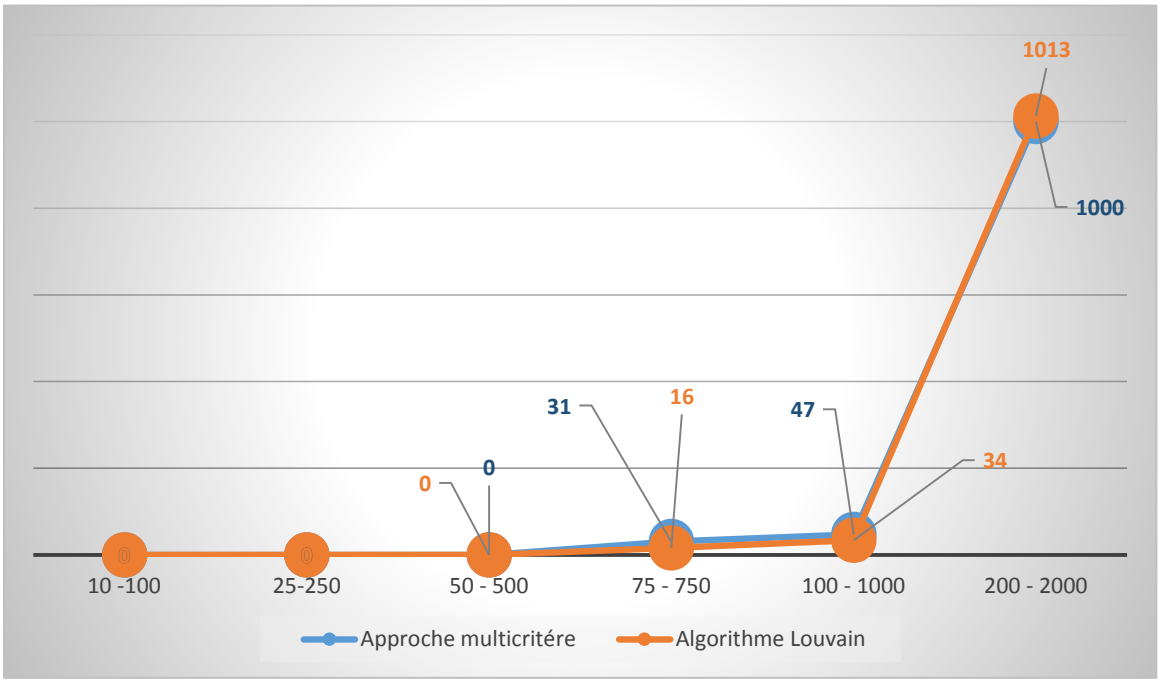


Figure III.26. Variation du temps d'exécution en fonctions du nombre d'utilisateurs et de services (Algorithme Louvain)

Par rapport à l'algorithme de Louvain qui est performant et développé dans une plateforme puissante et par des professionnels, notre application a montré de bons résultats qui montrent la robustesse des algorithmes et qui peut montrer plus de performance d'avantage en utilisant une machine plus puissante et dans le cas de réseaux complexes contenant un grand nombre d'utilisateurs.

Notons que l'algorithme de Louvain a été exécuté dans un logiciel professionnel (Pajek-XXL) dans sa version 5.0.1 alors que notre application est implémentée dans sa première version où on n'a pas très focalisé notre travail sur l'optimisation des algorithmes. Mais avec un peu plus de temps de développement et une implémentation sur une machine multiprocesseur (HPC par exemple) l'application peut donner de meilleurs résultats, malgré ça on remarque qu'avec l'augmentation du nombre d'utilisateurs, le temps d'exécution des deux approche se rapprochent considérablement.

6. CONCLUSION

Dans ce chapitre nous avons présenté l'application réalisée en commençant en premier lieu par exposer les outils et les technologies utilisées dans l'implémentation de l'application, ensuite on a présenté l'interface de l'application ainsi que ses fonctionnalités et on a effectué une démonstration de la mise en marche de l'application.

A la fin du chapitre nous avons fait une comparaison entre notre méthode et une autre méthode de génération de communautés ; les résultats ont montré l'efficacité ainsi que la robustesse en temps d'exécution de notre approche par rapport aux autres méthodes qui peuvent planter dans le cas de réseaux complexes.

Conclusion générale

Conclusion générale

Le travail réalisé dans cette thèse s'inscrit dans le domaine de détection de communautés dans les réseaux. Nous nous sommes particulièrement intéressés à la détection de communautés en suivant une approche basée sur l'interaction des utilisateurs avec les services web.

L'approche s'appuie sur le calcul de convenance entre les profils utilisateurs et services ainsi que le filtrage par une multitude de critères choisi par l'usager de l'application.

On a démontré tout au long des trois chapitres de ce mémoire la méthode ainsi que l'approche utilisé commençant par un tour d'horizon sur la découverte de communautés, les algorithmes utilisés ainsi que des exemples de la vie réelle. On a aussi parlé du domaine de la théorie des graphes en vue de son importance et de sa grande présence dans le domaine de découverte de communautés, ces dernières étant représentées par des graphes dit n-partie.

La génération de communauté selon notre approche est fondée sur quatre processus essentiels, le premier est la création d'une base de donnée qui peut être utilisé comme un benchmark gratuit étant donné sa conception robuste, le deuxième est celui du calcul de convenance entre deux types de nœud différents en utilisant la formule décrite dans la section 3.2.1 du chapitre II, le troisième concerne la génération des accès entre les profils ainsi que le filtrage selon des paramètres saisi par le décideur et le dernier est la génération de communautés.

A la fin de l'étude comparative, les résultats obtenus ont montré que l'approche de découverte de communauté multicritère est plus précise et plus rapide que l'approche n'utilisant pas de critères. Ainsi on a montré l'importance d'introduire les services web dans le processus de détection de communautés, en espérant en long terme que plus d'amélioration seront introduites dans cette approche en ajoutons de nouveaux critères, une base de données réelles et en utilisant des machines plus puissantes afin d'augmenter la rapidité d'exécution.

Bibliographie

[1] : http://deptinfo.unice.fr/twiki/pub/Linfo/Organisation%20Rapports/rapport-WebServices.pdf

[2] : http://www-igm.univ-mlv.fr/~dr/XPOSE2004/woollams/objectifs.html

[3] : http://www.w3ii.com/fr/webservices/web_services_architecture.html

[4] :http://docs.embarcadero.com/products/rad_studio/radstudio2007/RS2007_helpupdates/HUpdate4/FR/html/devnet/webservicesprotocol_xml.html

[5] : Les services web – Openclassrooms

[6] : https://fr.slideshare.net/mohamedyoussfi9/support-web-services-soap-et-restful-mr-youssfi

[7] : http://pierremerckle.fr/2011/10/comment-enseigner-les-reseaux-sociaux-au-lycee-par-lenquete/

[8]: J. Scott, Social network analysis, a handbook. Deuxième edition, Edition Sage,2000.

[9] : L.C. Freeman, Centrality in social networks: Conceptual Clarification. Social Networks. 1, 215-239.

[10]: N. J., On Centrality in a graph". Scandinavian Journal of Psychology 15:322-336

[11]: https://fr.wikipedia.org/wiki/Modularit%C3%A9_(r%C3%A9seaux)

[12]: Guimerà R, Sales-Pardo M, Amaral LAN. 2007 Module identification in bipartite and directed networks. Phys. Rev. E 76, 036102. (doi:10.1103/PhysRevE.76.036102)

[13]: Thébault E. 2013 Identifying compartments in presence–absence matrices and bipartite networks: insights into modularity measures. J. Biogeogr. 40, 759–768

[14]: http://babelon.babel31.com/definition-de-la-notion-de-communaute

[15]: Chihab Hanachi .Détection de communautés dynamiques dans les réseaux temporels .Thèse de doctorat, Université Toulouse3 Paul Sabatier (UT3 Paul Sabatier). 2013.

[16]: Thomas Aynaud. Détection des communautés dans les réseaux dynamiques.Université Pierre et Marie Curie. 2010.

[17]: Thèse Doctorat ‘Détection de communautés dynamique dans des réseaux temporels’. Rémy CAZABET université de Toulouse

[18]: https://www.slideshare.net/SatyakiSikdar1/community-detection-in-social-networks-a-brief-overview

[19] : Rushed Kanawati. Détection de communautés dans les grands graphes d'interactions (multiplexes) : état de l'art. 2013. <hal-00881668v2>

[20]: A. Lancichinetti, M. Kivelä, J. Saramäki, S. Fortunato, “Characterizing the Community

[21] : Saidani Nacera , Terras Ikram ,découverte de communautés basée service , mémoire de Master en Informatique Université de Dr Yahia Farès de Médéa ,Algérie 2015.

[22] : Chantal Cherifi « Classification et Composition de Services Web : Une Perspective Réseaux Complexes» Thèse de doctorat Le 9 décembre 2011

[23]: Les Web Services, Rapport de TE, Université de Nice-Sophia Antipolis

[24] : https://www.macreationdentreprise.fr/definition-reseaux-sociaux/

[25] : Pons and M. Latapy, “Computing communities in large networks using random walks,” Journal of Graph Algorithms and Applications , vol. 10, no. 2, pp. 191-218, 2005

[26]: Newman ME, Watts DJ, Strogatz SH. 2002 Random graph models of social networks. Proc. Natl Acad. Sci. USA99, 2566–2572. (doi:10.1073/pnas.012582999)

[27]: Saavedra S, Stouffer DB, Uzzi B, Bascompte J. 2011 Strong contributors to network persistence are the most vulnerable to extinction. Nature 478, 233–235. (doi:10.1038/nature10433)

[28]: Olesen JM, Bascompte J, Dupont YL, Jordano P. 2007 The modularity of pollination networks. Proc. Natl Acad. Sci. USA 104, 19891–19896. (doi:10.1073/pnas.0706375104)

[29]: Découverte multicritères des réseaux sociaux basée service – Mesrati Hicham & Abbas Amina, Université Yahia Fares, Médéa

[30] : GABAY, Joseph, GABAY, David, UML 2 analyse et conception. Paris : Dunod ,2008

[31] : http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation#L2

[32]: http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation#L2-3

[33] : http://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-interaction#L7-3-2

[34] : http://laurent-audibert.developpez.com/Cours-UML/?page=introduction-modelisation-objet#L1-4-3-b

[35]: http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes#L3-2-1

[36] : https://www.java.com/fr/download/faq/whatis_java.xml

[37]: http://quanti.hypotheses.org/512/

[38]: https://cel.archives-ouvertes.fr/docs/00/56/44/14/PDF/outils_pajek.pdf

[39] : https://openclassrooms.com/courses/apprenez-a-programmer-en-java/mieux-structurer-son-code-le-pattern-mvc

[40]: http://www.commentcamarche.net/contents/1030-introduction-a-rmi-remote-method-invocation

[41]: https://www.jmdoudoux.fr/java/dej/chap-rmi.htm

[42] : https://github.com/harry1357931/Remote_Method_Invocation-RMI