

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Farhat Abbas - Sétif 1 -



## *THESE*

Présentée à la Faculté des Sciences

Département d'Informatique  
Pour l'Obtention du Diplôme de

## *DOCTORAT EN SCIENCES*

Option : Informatique

## Thème

---

**Plateforme sémantique Cloud Computing pour la gestion  
des applications ERP sensibles au contexte**

---

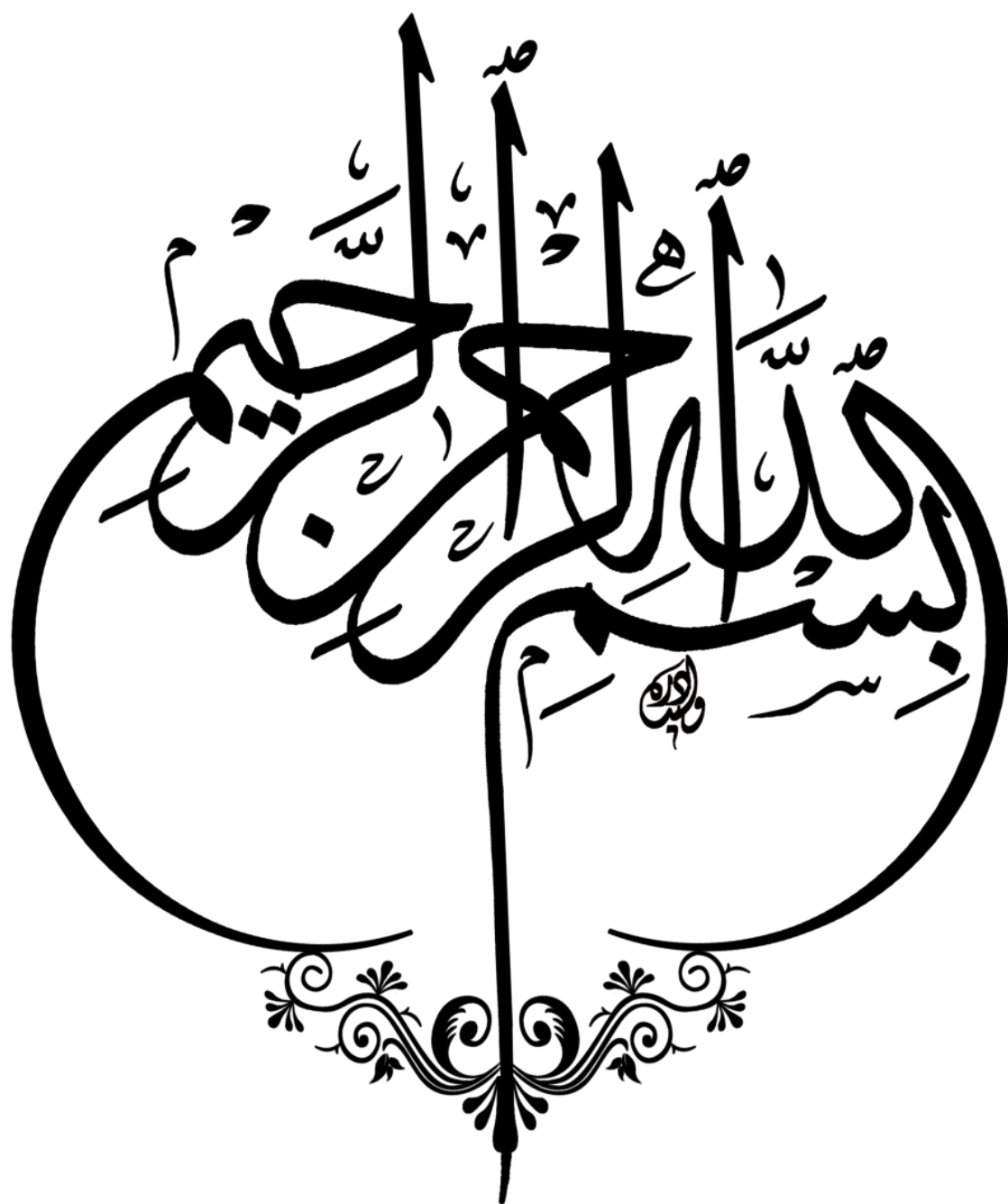
Par

**Hamza REFFAD**

Soutenu le : 27/04/2019

Devant le jury composé de :

|                         |            |                       |            |
|-------------------------|------------|-----------------------|------------|
| Pr. Makhlouf Alliouat   | Professeur | Université Setif-1    | Président  |
| Dr. Adel Alti           | MCA        | Université Setif-1    | Rapporteur |
| Pr. Abdelkrim Amirat    | Professeur | Université Souk Ahras | Examineur  |
| Dr. Farid Nouioua       | MCA        | Université BBA        | Examineur  |
| Dr. Abdelmalek Boudries | MCA        | Université Bejaia     | Examineur  |
| Pr. Fouzi Semchedine    | Professeur | Université Setif-1    | Examineur  |



# DÉDICACE

*À mes parents*

*À toute ma famille*

*À tous mes amis...*

**HAMZA**

# REMERCIEMENTS

Je remercie Dieu le tout Puissant qui m'a donné la force et la volonté pour réaliser ce modeste travail.

Je remercie mon directeur de thèse, Docteur **Adel Alti**, pour m'avoir proposé ce sujet intéressant, et pour les orientations et les conseils qu'il m'a donné.

Je remercie également les membres de jury. Professeur **Makhlouf Alliouat** de me faire l'honneur de présider le jury de ma thèse. Ainsi que, Professeur Mohamed **Abdelkarim Amirat**, Docteur **Farid Nouioua**, Docteur **Abdelmalek Boudries** et Professeur **Fouzi Semchedine** pour avoir accepté de lire et d'évaluer mon travail de thèse.

Je tiens à remercier mes amis et collègues pour leur collaboration durant toutes ces années.

Je tiens à remercier particulièrement Mme **Aicha Mebarkia** et Mr **Kamel Mebarkia** de leur aide et leurs conseils.

Enfin je remercie tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail.

# Résumé

Actuellement, la technologie Cloud est largement adoptée par les entreprises pour développer des solutions informatiques de qualité. En effet, les Petites et les Moyennes Entreprises (PME) sont à la recherche d'ERP sur mesure afin d'automatiser judicieusement leurs activités commerciales. Par conséquent, le besoin croissant de la flexibilité des processus d'affaires ERP issus de services Cloud hétérogènes de différents fournisseurs avec différentes QoS, qui optimise les coûts et les délais, devient une préoccupation majeure. En effet, il est très utile pour répondre aux évolutions requises de différents besoins fonctionnels et non fonctionnels des PME. La plupart des systèmes ERP Cloud existants (SAP, Oracle ERP Cloud, etc.) sont exposés en mode SaaS (Software as a Service) et ils ne sont pas suffisamment flexibles pour prendre en charge l'adaptation automatique des processus d'affaire ERP. L'objectif de notre travail est de construire un système qui permet de générer un processus d'affaire ERP personnalisé en tant que service composite en fonction des besoins des clients et des changements du contexte, via la composition optimale des services Cloud sémantiques. Pour ce faire, nous avons proposé un système avec une nouvelle ontologie pour la description sémantique et une gestion efficace du processus d'élaboration ERP end-to-end. En spécifiant les besoins fonctionnels de PME, le système génère automatiquement un processus d'affaire ERP virtuel. Par ailleurs, un algorithme à deux phases est proposé pour la composition des services Cloud afin d'obtenir un processus d'affaire ERP optimal, concret et personnalisé. La première phase est utilisée pour sélectionner les services composites pertinents qui respectent les contraintes de l'entreprise. La seconde est utilisée pour sélectionner la solution optimale unique à partir du résultat de la première phase en fonction des préférences de l'entreprise. Une extension de notre approche avec les aspects d'agents est également proposée pour améliorer la performance du système en termes de temps d'exécution. Afin de valider nos propositions, un prototype est développé offrant ainsi une meilleure flexibilité pour respecter la qualité des priorités globales de l'entreprise et son contexte. Les résultats expérimentaux montrent la pertinence et l'efficacité de l'approche proposée.

**Mots clés :** ERP, Cloud, ontologie, composition des services, QoS, contexte, NSGA-II, optimisation, multi-agent.

## Abstract

Nowadays, Cloud technology is widely adopted by companies to develop quality IT solutions. Small and medium enterprises (SMEs) are looking for the best preeminent customized Cloud Enterprise Resource Planning (ERP) to automate wisely their business activities. Therefore, the increasing need of flexible ERP business process from heterogeneous Cloud services of different service providers which carried out on cost and time benefits becomes a major concern. This is very useful for meeting the required evolutions of different functional and non-functional business needs. Most existing ERP Cloud systems (SAP, Oracle ERP Cloud, etc.) are exposed in Software as a Service (SaaS) mode and they are not flexible enough to support the automatic adaptation of ERP business processes. The goal of our work is to build a system that can generate a relevant ERP business process as a composite service according to customer needs and contextual changes via the optimal semantic Cloud services composition. Therefore, we have proposed a system with a new ontology for the semantic description and efficient management of the end-to-end ERP building process as well as a Cloud services composition approach dedicated to context-aware ERPs. By specifying the functional requirements, the system automatically generates a virtual business process ERP. Moreover, a two-stage algorithm for the Cloud services composition is proposed to obtain an optimal, concrete and personalized ERP business process. The first stage is used to select the relevant composite services that meet the enterprise's constraints. The second is used to select a single optimal solution from the outcome of the first phase according to the enterprise's preferences. We have also proposed an extension of our approach using the multi-agent aspect to improve the system performance in terms of execution time. To illustrate our approach, we presented a prototype that offers great flexibility to respect enterprise's global quality priorities. The experimental results show the relevance and effectiveness of the proposed approach.

**Keywords :** ERP, Cloud, ontology, service composition, QoS, context, NSGA-II, optimization, multi-agent.

## ملخص

حاليا، التكنولوجيا السحابية معتمدة على نطاق واسع من قبل الشركات لتطوير تكنولوجيا المعلومات عالية الجودة. من ناحية أخرى، تبحث الشركات الصغيرة ومتوسطة الحجم (SMEs) عن برامج تخطيط الموارد المصممة خصيصا من أجل التسيير الآلي لعملياتها التجارية بحكمة. ونتيجة لذلك، أصبحت الحاجة إلى مرونة عمليات تخطيط موارد المؤسسات الناتجة عن الخدمات السحابية الغير متجانسة المتوفرة لدى مزودي الخدمة بجودة مختلفة، والتي تعمل على تقليل التكاليف وزيادة سرعة الاستجابة، مصدر توجه كبير. هذا مفيد للغاية لتلبية التطورات المطلوبة لمختلف احتياجات الوظيفية وغير الوظيفية للمؤسسات. تتعرض معظم أنظمة تخطيط موارد المؤسسات السحابية الحالية (SAP، Oracle ERP Cloud، إلخ) في اعتماد SaaS (Software As A Service) وهي ليست مرنة بدرجة كافية لدعم التكيف التلقائي لأعمال المؤسسات. الهدف من عملنا هو بناء نظام يمكن أن ينتج عملية تخطيط موارد المؤسسات كخدمة مركبة وفقا لاحتياجات المؤسسات والتغيرات السياقية عن طريق التركيب الدلالي الأمثل للخدمات السحابية. تحقيقا لهذه الغاية، اقترحنا نظاما مبني على نظام الهياكل الدلالية (أنطولوجيا) جديد للوصف الدلالي والإدارة الفعالة لعملية بناء تخطيط موارد المؤسسات من البداية إلى النهاية وكذلك نهج تكوين الخدمات السحابية المخصصة لتخطيط موارد المؤسسات المتغيرة السياق. من خلال تحديد المتطلبات الوظيفية، يقوم النظام آليا بإنشاء تخطيط موارد الأعمال الافتراضية. بعد ذلك، تم اقتراح خوارزمية ذو مرحلتين لتكوين الخدمات السحابية للحصول على أفضل عملية أعمال ERP ملموسة ومعدلة وفقا لطلب المؤسسة. يتم استخدام المرحلة الأولى لتحديد الخدمات المركبة ذات الصلة التي تلبي شروط المؤسسة. أما الثانية فتستخدم لاختيار حل واحد مثالي من نتائج المرحلة الأولى وفقا لأولويات المؤسسة. لقد اقترحنا أيضا تمديدا لنهجنا باستخدام الجانب متعدد الأعوان (agents) لتحسين أداء النظام من حيث وقت التنفيذ. لتحقيق نهجنا قدمنا، نموذجا أوليا يوفر مرونة عالية في احترام جودة الأولويات العامة للشركة. تظهر النتائج التجريبية أهمية وفعالية النهج المقترح.

### كلمات البحث:

تخطيط موارد المؤسسات، السحابة (Cloud)، الهياكل الدلالية (أنطولوجيا)، تركيب الخدمات، جودة الخدمة (QoS)، NSGA-II، السياق، التحسين إلى الأمثل، متعدد الاعوان (multi-agent).

# Table des matières

|                                                             |              |
|-------------------------------------------------------------|--------------|
| <b>Table des figures</b>                                    | <b>xii</b>   |
| <b>Liste des tableaux</b>                                   | <b>xv</b>    |
| <b>Liste des abréviations</b>                               | <b>xvi</b>   |
| <b>Introduction générale</b>                                | <b>1</b>     |
| 1 Contexte de la thèse . . . . .                            | 1            |
| 2 Problématique . . . . .                                   | 2            |
| 3 Objectifs et Contributions . . . . .                      | 3            |
| 4 Organisation de la thèse . . . . .                        | 4            |
| <br><b>Partie I Etat de l’art</b>                           | <br><b>7</b> |
| <b>1 ERP et Cloud Computing</b>                             | <b>8</b>     |
| 1 Introduction . . . . .                                    | 9            |
| 2 Cloud Computing . . . . .                                 | 9            |
| 2.1 Définitions . . . . .                                   | 11           |
| 2.2 Caractéristiques du Cloud Computing . . . . .           | 12           |
| 2.3 Modèle de service . . . . .                             | 13           |
| 2.3.1 Software as a Service « <i>SaaS</i> » . . . . .       | 14           |
| 2.3.2 Platform as a Service « <i>PaaS</i> » . . . . .       | 15           |
| 2.3.3 Infrastructure as a Service « <i>IaaS</i> » . . . . . | 15           |
| 2.4 Modèle de déploiement . . . . .                         | 16           |
| 2.4.1 Cloud publique . . . . .                              | 16           |
| 2.4.2 Cloud privé . . . . .                                 | 17           |
| 2.4.3 Cloud hybride . . . . .                               | 17           |
| 2.4.4 Cloud communautaire . . . . .                         | 17           |

|          |                                                         |           |
|----------|---------------------------------------------------------|-----------|
| 3        | Enterprise Resource Planning (ERP)                      | 18        |
| 3.1      | C'est quoi un ERP?                                      | 18        |
| 3.2      | Pourquoi un ERP?                                        | 19        |
| 3.3      | Caractéristiques d'un ERP                               | 19        |
| 3.3.1    | Architecture client/serveur d'un ERP                    | 19        |
| 3.3.2    | Architecture modulaire d'un ERP                         | 20        |
| 3.3.3    | Fonctionnement d'un ERP                                 | 21        |
| 3.4      | Implémentation des systèmes ERP                         | 22        |
| 3.5      | ERP sur le marché                                       | 23        |
| 3.6      | Les avantages et les inconvénients des systèmes ERP     | 26        |
| 3.6.1    | Avantages                                               | 26        |
| 3.6.2    | Inconvénients                                           | 28        |
| 3.7      | Les architectures ERP                                   | 29        |
| 3.7.1    | Architecture à trois niveaux                            | 29        |
| 3.7.2    | Architecture WEB                                        | 29        |
| 3.7.3    | Architecture orientée service                           | 30        |
| 3.7.4    | Architecture Cloud                                      | 31        |
| 4        | ERP et Cloud Computing                                  | 33        |
| 4.1      | Cloud ERP hébergé                                       | 33        |
| 4.2      | Cloud ERP en mode SaaS                                  | 34        |
| 5        | Sensibilité au Contexte                                 | 36        |
| 5.1      | Définitions                                             | 36        |
| 5.2      | Catégories du contexte                                  | 37        |
| 5.3      | Applications sensibles au contexte                      | 37        |
| 5.4      | Contexte et ERP                                         | 38        |
| 6        | Conclusion                                              | 38        |
| <b>2</b> | <b>Composition de services Cloud</b>                    | <b>39</b> |
| 1        | Introduction                                            | 40        |
| 2        | Processus d'affaire                                     | 40        |
| 2.1      | Définition                                              | 41        |
| 2.2      | Exemple                                                 | 41        |
| 2.3      | Les exigences fonctionnelles et non-fonctionnelles      | 42        |
| 3        | Composition de services Cloud                           | 44        |
| 3.1      | Service Cloud                                           | 44        |
| 3.1.1    | Propriétés des services Cloud                           | 45        |
| 3.1.2    | Approches principales de description des services Cloud | 46        |

|       |                                                                                   |    |
|-------|-----------------------------------------------------------------------------------|----|
| 3.2   | Les étapes de la composition des services Cloud . . . . .                         | 47 |
| 3.3   | Sélection des services Cloud . . . . .                                            | 48 |
| 3.3.1 | Spécification de la requête utilisateur . . . . .                                 | 48 |
| 3.3.2 | Localisation des services Cloud . . . . .                                         | 49 |
| 3.3.3 | Appariement des requêtes et des services Cloud . . . . .                          | 50 |
| 3.4   | Degré d'automatisation de la composition de services Cloud . . . . .              | 51 |
| 3.4.1 | Composition manuelle . . . . .                                                    | 51 |
| 3.4.2 | Composition automatique . . . . .                                                 | 52 |
| 3.4.3 | Composition semi-automatique . . . . .                                            | 52 |
| 3.5   | Approches de la composition des services Cloud . . . . .                          | 52 |
| 3.5.1 | Composition statique . . . . .                                                    | 53 |
| 3.5.2 | Composition dynamique . . . . .                                                   | 53 |
| 3.6   | Aspects comportementaux des services composites . . . . .                         | 53 |
| 3.6.1 | Orchestration . . . . .                                                           | 53 |
| 3.6.2 | Chorégraphie . . . . .                                                            | 54 |
| 3.7   | Challenges de la composition des services Cloud . . . . .                         | 55 |
| 3.7.1 | Respect des besoins fonctionnels de l'utilisateur . . . . .                       | 55 |
| 3.7.2 | Optimisation des critères de QoS . . . . .                                        | 55 |
| 3.7.3 | Augmentation de la flexibilité : "Sensibilité au contexte" . . . . .              | 55 |
| 3.7.4 | Dynamicité . . . . .                                                              | 56 |
| 4     | Principaux travaux dans le domaine de la composition des services Cloud . . . . . | 56 |
| 4.1   | Les approches méta-heuristiques . . . . .                                         | 58 |
| 4.1.1 | Les approches fondées sur la scalarisation . . . . .                              | 58 |
| 4.1.2 | Approches basées sur le Pareto (approches multi-objectif) . . . . .               | 60 |
| 4.2   | Les ERP basés sur le WEB . . . . .                                                | 61 |
| 4.3   | Les ERP à base de services . . . . .                                              | 62 |
| 4.4   | Synthèse et Comparaison . . . . .                                                 | 62 |
| 5     | Conclusion . . . . .                                                              | 65 |

## Partie II Contributions

66

|          |                                                                      |           |
|----------|----------------------------------------------------------------------|-----------|
| <b>3</b> | <b>Modèle de description des applications ERP à base d'ontologie</b> | <b>67</b> |
| 1        | Introduction . . . . .                                               | 68        |
| 2        | Ontologie proposée . . . . .                                         | 69        |
| 3        | Modèle sémantique du service . . . . .                               | 70        |
| 3.1      | Service virtuel . . . . .                                            | 71        |

|          |                                                                                                   |           |
|----------|---------------------------------------------------------------------------------------------------|-----------|
| 3.2      | Service concret . . . . .                                                                         | 71        |
| 4        | Modèle sémantique de QoS à deux dimensions . . . . .                                              | 73        |
| 4.1      | Modèle de QoS-Utilité . . . . .                                                                   | 73        |
| 4.2      | Modèle de QoS-LongueVie . . . . .                                                                 | 74        |
| 4.3      | Modèle de QoS . . . . .                                                                           | 75        |
| 5        | Modèle sémantique du contexte . . . . .                                                           | 76        |
| 5.1      | Le contexte du service Cloud . . . . .                                                            | 76        |
| 5.2      | Le contexte du Client . . . . .                                                                   | 77        |
| 5.3      | Le contexte de l'environnement . . . . .                                                          | 80        |
| 6        | Modèle sémantique du processus d'affaire . . . . .                                                | 80        |
| 6.1      | Processus d'affaire virtuel . . . . .                                                             | 81        |
| 6.2      | Processus d'affaire concret . . . . .                                                             | 81        |
| 7        | L'interrogation et l'inférence . . . . .                                                          | 82        |
| 8        | Exemple de PA selon l'ontologie CxQSCloudSERP . . . . .                                           | 84        |
| 9        | Conclusion . . . . .                                                                              | 87        |
| <b>4</b> | <b>Nouvelle approche dédiée aux ERP sensibles au contexte à base d'ontologie</b>                  | <b>88</b> |
| 1        | Introduction . . . . .                                                                            | 89        |
| 2        | Motivation : étude de cas et exigences . . . . .                                                  | 90        |
| 2.1      | Etude de cas . . . . .                                                                            | 90        |
| 2.2      | Exigences . . . . .                                                                               | 91        |
| 3        | Architecture de composition d'ERP sensible au contexte . . . . .                                  | 91        |
| 4        | Modèle fonctionnel . . . . .                                                                      | 93        |
| 5        | Algorithme de composition et d'optimisation sensible au contexte basé sur la sémantique . . . . . | 95        |
| 6        | Application de l'algorithme proposé à l'étude de cas . . . . .                                    | 102       |
| 6.1      | Génération d'un processus d'affaire virtuel (VBP) optimal . . . . .                               | 102       |
| 6.2      | Génération des solutions Pareto-optimal . . . . .                                                 | 102       |
| 6.3      | Sélection de la solution finale . . . . .                                                         | 107       |
| 7        | Nouvelle extension : CxQSCloudERP à base d'agents . . . . .                                       | 107       |
| 7.1      | Architecture et principe de fonctionnement . . . . .                                              | 111       |
| 7.2      | Description de l'algorithme SCwA-CCS-2S . . . . .                                                 | 114       |
| 7.3      | Application de l'algorithme SCwA-CCS-2S à l'étude de cas . . . . .                                | 117       |
| 8        | Conclusion . . . . .                                                                              | 119       |

---

|          |                                                       |            |
|----------|-------------------------------------------------------|------------|
| <b>5</b> | <b>Réalisation : Prototype et évaluations</b>         | <b>120</b> |
| 1        | Introduction . . . . .                                | 120        |
| 2        | Réalisation du prototype . . . . .                    | 121        |
| 2.1      | Environnement et outils techniques . . . . .          | 121        |
| 2.2      | Implémentation de l'ontologie . . . . .               | 121        |
| 2.3      | Présentation du prototype . . . . .                   | 126        |
| 2.4      | Application du prototype à l'étude de cas . . . . .   | 127        |
| 3        | Évaluation des performances et comparaisons . . . . . | 129        |
| 3.1      | Évaluation de la pertinence des solutions . . . . .   | 131        |
| 3.2      | Comparaison de la précision des solutions . . . . .   | 132        |
| 3.3      | Comparaison des solutions Pareto-optimales . . . . .  | 133        |
| 3.4      | Comparaisons sur la solution finale . . . . .         | 134        |
| 3.5      | Amélioration des performances : Multi-agent . . . . . | 136        |
| 4        | Conclusion . . . . .                                  | 138        |
|          | <b>Conclusion générale</b>                            | <b>140</b> |
|          | <b>Bibliographie</b>                                  | <b>144</b> |

# Table des figures

|      |                                                               |    |
|------|---------------------------------------------------------------|----|
| 1.1  | Architecture du Cloud Computing . . . . .                     | 10 |
| 1.2  | Modèle de service . . . . .                                   | 14 |
| 1.3  | Modèle de déploiement . . . . .                               | 16 |
| 1.4  | Architecture client/serveur d'un ERP . . . . .                | 20 |
| 1.5  | Architecture modulaire d'un ERP . . . . .                     | 21 |
| 1.6  | Processus d'affaire . . . . .                                 | 22 |
| 1.7  | La chaîne d'activités d'un processus d'affaire . . . . .      | 22 |
| 1.8  | Top 10 des fournisseurs de logiciels ERP . . . . .            | 27 |
| 1.9  | Les différentes architectures à niveaux . . . . .             | 29 |
| 1.10 | Architecture web . . . . .                                    | 30 |
| 1.11 | Architecture SOA . . . . .                                    | 31 |
| 1.12 | Architecture Cloud . . . . .                                  | 32 |
| 1.13 | Les limites des différentes architectures ERP . . . . .       | 32 |
| 2.1  | Processus d'affaire "Commande produit" . . . . .              | 42 |
| 2.2  | Composition de services . . . . .                             | 44 |
| 2.3  | OWL-S : Semantic Markup for Web Services . . . . .            | 47 |
| 2.4  | L'orchestration des services Cloud . . . . .                  | 54 |
| 2.5  | La chorégraphie des services Cloud . . . . .                  | 54 |
| 3.1  | Un aperçu général sur l'ontologie CxQSCloudSERP . . . . .     | 70 |
| 3.2  | Représentation d'un service virtuel . . . . .                 | 71 |
| 3.3  | Représentation d'un service concret (service Cloud) . . . . . | 72 |
| 3.4  | Modèle sémantique du service . . . . .                        | 72 |
| 3.5  | Modèle sémantique de QoS . . . . .                            | 73 |
| 3.6  | Modèle sémantique du contexte . . . . .                       | 76 |
| 3.7  | Le modèle sémantique du processus d'affaire . . . . .         | 80 |
| 3.8  | Les modèles de composition . . . . .                          | 81 |

|      |                                                                                |     |
|------|--------------------------------------------------------------------------------|-----|
| 3.9  | Processus de vente en ligne représenté par CxQSCloudSERP . . . . .             | 86  |
| 4.1  | L'architecture de composition d'ERP sensible au contexte . . . . .             | 93  |
| 4.2  | Le modèle fonctionnel de composition d'ERP sensible au contexte . . . . .      | 93  |
| 4.3  | L'algorithme d'optimisation CCS-2S . . . . .                                   | 97  |
| 4.4  | Codage de la solution . . . . .                                                | 98  |
| 4.5  | Les opérateurs génétiques. (a) Croisement, (b) Mutation. . . . .               | 100 |
| 4.6  | Processus d'affaire virtuel optimal de la commande de produits en ligne . .    | 103 |
| 4.7  | Population initiale . . . . .                                                  | 105 |
| 4.8  | Individu de la population . . . . .                                            | 105 |
| 4.9  | Croisement . . . . .                                                           | 106 |
| 4.10 | Mutation . . . . .                                                             | 106 |
| 4.11 | Recombinaison et sélection . . . . .                                           | 106 |
| 4.12 | L'ensemble des individus du front Pareto-optimal . . . . .                     | 108 |
| 4.13 | Les valeurs des attributs du front Pareto-optimal . . . . .                    | 109 |
| 4.14 | La solution finale . . . . .                                                   | 109 |
| 4.15 | Les valeurs des attributs de QoS de la solution finale . . . . .               | 110 |
| 4.16 | Stratégie maître-esclave pour SCwA-CCS-ERP . . . . .                           | 110 |
| 4.17 | Stratégie en anneau pour SCwA-CCS-ERP . . . . .                                | 111 |
| 4.18 | Stratégie hybride pour SCwA-CCS-ERP . . . . .                                  | 112 |
| 4.19 | L'architecture générale de SCwA-CCS-ERP . . . . .                              | 113 |
| 4.20 | Modèle fonctionnel de l'approche SCwA-CCS-ERP . . . . .                        | 113 |
| 4.21 | Diagramme de Classe de SCwA-CCS . . . . .                                      | 115 |
| 4.22 | Diagramme d'interaction de SCwA-CCS . . . . .                                  | 116 |
| 4.23 | Les étapes de SCwA-CCS-2S appliquées à l'étude de cas . . . . .                | 118 |
| 5.1  | L'ontologie CxQSCloudSERP . . . . .                                            | 122 |
| 5.2  | Les classes et les propriétés de l'ontologie CxQSCloudSERP . . . . .           | 124 |
| 5.3  | Les individus de l'ontologie CxQSCloudSERP . . . . .                           | 126 |
| 5.4  | GUI pour les besoins fonctionnels du client . . . . .                          | 128 |
| 5.5  | Interface pour la spécification des contraintes QoS de l'ERP . . . . .         | 129 |
| 5.6  | Interface pour la spécification de l'ordre des préférences . . . . .           | 130 |
| 5.7  | Le résultat de la solution optimale du cas d'étude par notre prototype . . . . | 130 |
| 5.8  | La distance entre le Pareto-optimal attendu et le Pareto-optimal trouvé . . .  | 132 |
| 5.9  | Comparaison des solutions Pareto-optimales entre CCS-2S et NSGA-II . .         | 134 |

|      |                                                                                                                                                                                       |     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.10 | La solution finale optimale obtenue par CCS-2S en utilisant les préférences du client (temps et prix) : (a) avec violation de contraintes, (b) sans violation de contraintes. . . . . | 135 |
| 5.11 | Temps d'exécution de CCS-2S vs SCwA-CCS-2S . . . . .                                                                                                                                  | 138 |

# Liste des tableaux

|     |                                                                                                                  |     |
|-----|------------------------------------------------------------------------------------------------------------------|-----|
| 2.1 | Comparaison des approches de composition de services . . . . .                                                   | 64  |
| 3.1 | Description du modèle des critères de QoS-Utilité . . . . .                                                      | 74  |
| 3.2 | Description du modèle des critères de QoS-LongueVie . . . . .                                                    | 74  |
| 3.3 | Correspondance entre les spécifications linguistiques du client et les valeurs sémantiques . . . . .             | 78  |
| 3.4 | Les fonctions d'agrégations de QoS . . . . .                                                                     | 82  |
| 3.5 | Requête d'interrogation des services Cloud en SPARQL. . . . .                                                    | 83  |
| 3.6 | Règle d'inférence des services virtuels d'un BP en SWRL. . . . .                                                 | 84  |
| 3.7 | Exemple de DL de notre ontologie. . . . .                                                                        | 85  |
| 4.1 | Les valeurs des attributs de QoS de chaque service concret . . . . .                                             | 104 |
| 5.1 | Environnement technique . . . . .                                                                                | 122 |
| 5.2 | Description des règles SWRL . . . . .                                                                            | 125 |
| 5.3 | Description des règles SPARQL . . . . .                                                                          | 127 |
| 5.4 | Le taux de violation de CCS-2S . . . . .                                                                         | 131 |
| 5.5 | La distance entre la solution Pareto-optimal attendu est la solution Pareto-optimal trouvé . . . . .             | 133 |
| 5.6 | Valeurs de QoS "temps" et "prix" obtenues par CCS-2S avec les différentes spécifications de préférences. . . . . | 136 |
| 5.7 | Comparaisons sur la solution finale avec l'optimisation mono-objectif. . . .                                     | 137 |

# Liste des abréviations

|               |                                                     |
|---------------|-----------------------------------------------------|
| ACO-WSC       | Ant Colony Optimization - Web Service Composition   |
| API           | Application Programming Interface                   |
| ATM           | Asynchronous Transfer Mode                          |
| B2B           | Business to Business                                |
| B2C           | Business to Consumer                                |
| BPMN          | Business Process Modeling Notation                  |
| CaaS          | Composition as a Service                            |
| CCS           | Composite Concrete Service                          |
| CCS-2S        | Concrete Composition Service with 2 Stages          |
| CRM           | Customer Relationship Management                    |
| CS            | Cloud Service                                       |
| CxQSCloudSERP | Context-aware Quality of Semantic Cloud Service ERP |
| DL            | Description logic                                   |
| ERP           | Enterprise Resource Planning                        |
| ERP-VCS       | ERP Virtual Composite Service                       |
| GUI           | Graphic User Interface                              |
| IaaS          | Infrastructure as a Service                         |
| ICA           | Imperialist Competitive Algorithm                   |
| IT            | Information Technology                              |
| LSA           | Latent Semantic Analysis                            |
| MOGA          | Multiple Objective Genetic Algorithm                |
| MOPSO         | Multiple Objective Particle Swarm Optimization      |
| NIST          | National Institute of Standards and Technology      |
| NSGA-II       | Non-dominated Sorting Genetic Algorithm             |
| OWL-S         | Semantic Markup for Web Services                    |
| PaaS          | Platform as a Service                               |

---

|                  |                                                                                         |
|------------------|-----------------------------------------------------------------------------------------|
| PGI              | Progiciel de Gestion Intégré                                                            |
| PME              | Petite et Moyenne Entreprise                                                            |
| PSO              | Particule Swarm Optimization                                                            |
| QoS              | Quality of Service                                                                      |
| QoS <sub>L</sub> | QoS-Long-Life                                                                           |
| QoS <sub>U</sub> | QoS-Utility                                                                             |
| RDF              | Resource Description Framework                                                          |
| REST             | Representational State Transfer                                                         |
| SaaS             | Software as a Service                                                                   |
| SAW              | Simple Additive Weighting                                                               |
| SAWSDL           | Semantic Annotations for WSDL and XML Schema                                            |
| SCwA-CCS-2S      | Semantic-Based Context-aWare Agents for the Composition of Cloud Services with 2 Stages |
| SCwA-CCS-ERP     | Semantic-Based Context-aWare Agents for the Composition of Cloud Services for ERP       |
| SI               | Système d'Information                                                                   |
| SLA              | Service Level Agreement                                                                 |
| SOA              | Service Oriented Architecture                                                           |
| SOAP             | Simple Object Access Protocol                                                           |
| SPARQL           | SPARQL Protocol and RDF Query Language                                                  |
| SPEA2            | Strength Pareto Evolutionary Algorithm                                                  |
| SWRL             | Semantic Web Rule Language                                                              |
| UDDI             | Universal Description Discovery and Integration                                         |
| UML              | Unified Modeling Language                                                               |
| URI              | Uniform Resource Identifier                                                             |
| USP              | Unique Selling Proposition                                                              |
| VBP              | Virtual Business Process                                                                |
| VS               | Virtual Service                                                                         |
| W3C              | World Wide Web Consortium                                                               |
| WS-Policy        | The Web Services Policy Framework                                                       |
| WSDL             | Web Services Description Language                                                       |
| WSIndex          | Web Services Index                                                                      |
| WSMO             | Web Service Modeling Ontology                                                           |
| XML              | Extensible Markup Language                                                              |

# Introduction générale

## 1 Contexte de la thèse

Ces dernières années, plusieurs innovations technologiques ont émergé pour répondre aux besoins du partage des ressources et de la communication entre les entreprises. En effet, ce sont de très bonnes stratégies et techniques pour développer des solutions informatiques de qualité. Les Petites et Moyennes Entreprises (PME) sont à la recherche de solutions flexibles et efficaces pour automatiser leurs activités commerciales. Ils visent à accéder à de meilleures opportunités de cycle de vie et de trouver des techniques avancées pour gérer à haut niveau les ressources informatiques tout en conservant leurs avantages compétitifs. Pour relever ces défis, les entreprises adoptent un logiciel de planification des ressources d'entreprise (ERP : Enterprise Resource Planning ) comme un outil principal dans la gestion et l'optimisation de leurs activités à travers un central de base de données dans lequel l'information est distribuée au sein des départements de l'entreprise dans un cadre transparent.

Un ERP est un logiciel générique qui sert à gérer les différentes activités d'une entreprise en intégrant l'ensemble de ses fonctionnalités telles que : la gestion des ressources humaines, la gestion comptable et financière, les relations client "Customer Relationship Management (CRM)", les achats, la gestion des stocks et inventaires, la gestion de la production, la distribution, l'approvisionnement et le e-commerce [1]. Cependant, les petites et moyennes entreprises souffrent du prix élevé de ces ERP, et par conséquent elles penchent vers des solutions verticales spécifiques à leurs besoins pour économiser les coûts d'investissement. Néanmoins, les évolutions des besoins des entreprises les obligent à réinvestir pour intégrer de nouvelles fonctionnalités pour répondre à ces évolutions, ce qui augmente encore plus le coût. Beaucoup d'entreprises, hésitent du choix de la solution à adopter. D'une part, les solutions verticales sont rigides mais moins chère, d'autre part, la solution ERP est plus flexible mais très chère. En outre, lors des évolutions des activités, les entreprises doivent passer par une étape de configuration qui est elle-même couteuse dans la plupart des cas. L'investissement dans un ERP ne se limite pas seulement au prix de l'ERP, il inclut également le coût de l'infrastructure matérielle nécessaire pour son déploiement et sa maintenance. Pour

réduire ces coûts, plusieurs fournisseurs d'ERP exposent des ERP dans le Cloud en mode SaaS dont l'entreprise ne se préoccupe ni du déploiement et la maintenance de l'ERP, ni de la reconfiguration de l'ERP en cas de changement des besoins de l'entreprise. Ce mode permet également aux entreprises d'accéder à l'ERP de n'importe où et à n'importe quel moment.

Par ailleurs, le contexte de chaque entreprise est spécifique et évolutif. Les entreprises souhaitent avoir un ERP sensible au contexte qui prend en considération les évolutions des activités de l'entreprise et ses besoins non-fonctionnels, ainsi que les changements de l'environnement afin de réagir continuellement aux éventuels changements.

## 2 Problématique

Avec la prolifération du Cloud Computing, les grands fournisseurs des systèmes ERP sur le marché tels que SAP, Oracle, Sage et Microsoft positionnent leurs offres ERP dans le modèle SaaS (Software as a Service) [2]. Néanmoins, ces systèmes ne sont pas suffisamment flexibles pour s'adapter aux évolutions dynamiques des besoins des entreprises (clients), ainsi que leurs contraintes et préférences contextuelles [3].

Le paradigme de la composition des services Cloud est adopté par plusieurs chercheurs en tant que solution des systèmes dynamiques en raison de deux avantages essentiels : la facilité d'intégration et la réduction des coûts via un modèle d'application hébergé [4]. Cependant, ce paradigme est un processus complexe, il passe par plusieurs étapes telles que la découverte, la sélection, la composition ou encore le déploiement [5]. Il doit également vérifier la cohérence et l'interopérabilité des services hétérogènes pour construire des plans de composition afin de générer un processus d'affaire correspondant aux besoins fonctionnels et non-fonctionnels des utilisateurs. L'ensemble de services Cloud hétérogènes avec différentes QoS (temps d'exécution, coût, disponibilité, fiabilité, sécurité, etc.) est offert par plusieurs fournisseurs de services Cloud.

Les techniques de planification [6–8] basées sur l'intelligence artificielle sont souvent considérées comme efficaces pour résoudre automatiquement les problèmes de composition [9]. Néanmoins, elles s'intéressent à offrir un service composite qui satisfait les besoins utilisateur en négligeant l'optimisation des QoS du processus d'affaire engendré.

Par ailleurs, plusieurs techniques d'optimisation de la composition de services Cloud sensible à la QoS ont été proposées en se basant sur la QoS (prix, temps d'exécution, disponibilité, fiabilité, sécurité, etc.) en tant que paramètres à optimiser [10–28]. Cependant, ces techniques ne tiennent pas en compte les évolutions des contraintes contextuelles du client, ni de ses préférences (haut débit, confiance élevée, accessibilité facile, etc.) lors du processus de sélection et de composition des services Cloud pertinents. De plus, elles ne gèrent pas de

manière efficace et flexible un grand nombre de services hétérogènes avec des qualités de services différentes fournis par différents fournisseurs de Cloud. Cette hétérogénéité implique une dégradation de qualité de contrôle dans la sélection et la composition des services.

Les entreprises ne se soucient pas des fournisseurs de services Cloud et les technologies utilisées. Ils souhaitent que les applications ERP puissent être personnalisées automatiquement en fonction de leurs besoins fonctionnels actuels, leurs contraintes et leurs préférences contextuelles. En effet, les technologies Web sémantiques jouent un rôle important pour comprendre le contexte client. Elles permettent une meilleure sélection de services pertinents parmi un grand nombre de candidats de services Cloud pour le processus de composition afin d'obtenir un ERP selon les besoins fonctionnels du client, ainsi que ses contraintes et ses préférences contextuelles. L'ontologie OWL-S est proposée pour la détection, la sélection et la composition des services Cloud [29]. Cependant, elle ne permet pas la représentation sémantique des QoS en tant que contraintes et préférences avec des valeurs qualitatives. En outre, elle ne prend pas en considération les informations contextuelles (client, service Cloud et l'environnement) dans le processus de sélection et de composition.

Face aux besoins évolutifs des entreprises ainsi qu'à l'augmentation des services Cloud offrant différentes QoS, le développement des ERP personnalisés nécessite de proposer des solutions efficaces et pertinentes qui répondent aux attentes des entreprises.

### 3 Objectifs et Contributions

L'objectif principal de cette thèse est de concevoir et réaliser une plateforme qui permet de générer des ERPs personnalisés flexibles et sensibles au contexte. En effet, chaque entreprise a ses propres besoins fonctionnels qui peuvent évoluer au cours du temps. Le système doit être flexible pour s'adapter rapidement à ces évolutions. Pour cela, nous avons adopté le paradigme de composition de service Cloud basé sur l'ontologie pour réutiliser les services existants dans le Cloud et les faire coopérer afin de fournir à l'entreprise le processus d'affaire qui répond à ses besoins. L'ontologie est utilisée pour guider le processus de composition en attribuant des propriétés sémantiques aux différents éléments nécessaires à la composition. En revanche, les entreprises exigent des contraintes et des préférences non-fonctionnelles (QoS)<sup>1</sup> selon leur contexte. Pour cette raison, nous avons proposé une approche pour l'optimisation des QoS du processus d'affaire ERP en fonction des contraintes

---

1. Les QoS sont souvent conflictuelles.

et préférences non-fonctionnelles de l'entreprise.

Les principales contributions de notre travail sont :

- ✓ Définition d'une nouvelle ontologie baptisée "CxQSCloudSERP" (Context-aware Quality of Semantic Cloud Service ERP) pour gérer un grand nombre de services Cloud hétérogènes regroupés sémantiquement en fonction de leur catégorie de service, de leurs descriptions fonctionnelles et contextuelles. Cette ontologie facilite le processus de découverte des services Cloud pertinents, l'interopérabilité des services Cloud et met en évidence les contraintes et les préférences sémantiques des entreprises. Elle est accompagnée de règles d'inférence pour la déduction automatique des services virtuelles et la qualité de service selon la spécification des besoins de l'entreprise.
- ✓ Proposition d'une nouvelle approche basée sur l'ontologie pour la composition de services Cloud afin de fournir un processus d'affaire optimal correspond aux contraintes et préférences de QoS (prix, temps d'exécution, disponibilité, sécurité et fiabilité) de l'entreprise avec l'évolution continue de son contexte (confiance et débit). L'approche proposée comporte un planificateur pour générer un processus d'affaire virtuel et un algorithme à deux phases pour sélectionner le processus d'affaire concret optimal. Nous avons également décrit une extension de l'approche proposée en introduisant l'aspect multi-agent afin d'améliorer la performance du système en matière de temps d'exécution.
- ✓ Implémentation d'un prototype en Java et Protégé pour la génération d'ERP personnalisé afin de valider notre approche en termes d'efficacité et de qualité de solutions correspondantes aux besoins de l'entreprise et de son contexte.

## 4 Organisation de la thèse

Ce manuscrit est structuré en cinq chapitres. Les deux premiers chapitres sont consacrés à l'état d'art liés à notre recherche, tandis que les chapitres 3, 4 et 5 exposent nos contributions.

Le chapitre 1 recouvre les principaux axes de recherches appartenant à notre problématique. Ce chapitre est structuré en trois sections. La première décrit le Cloud Computing, son utilité et ses caractéristiques. La deuxième section présente les systèmes ERP en mentionnant leurs objectifs, leurs caractéristiques, les architectures et leurs avantages/inconvénients. Quant à la dernière section, elle envisage la notion du contexte et son rapport avec les applications sensibles au contexte, et notamment avec les systèmes ERP.

Le chapitre 2 est consacré à la description du paradigme de composition de services Cloud. Il décrit la définition du processus d'affaire et les éléments capitaux des approches de

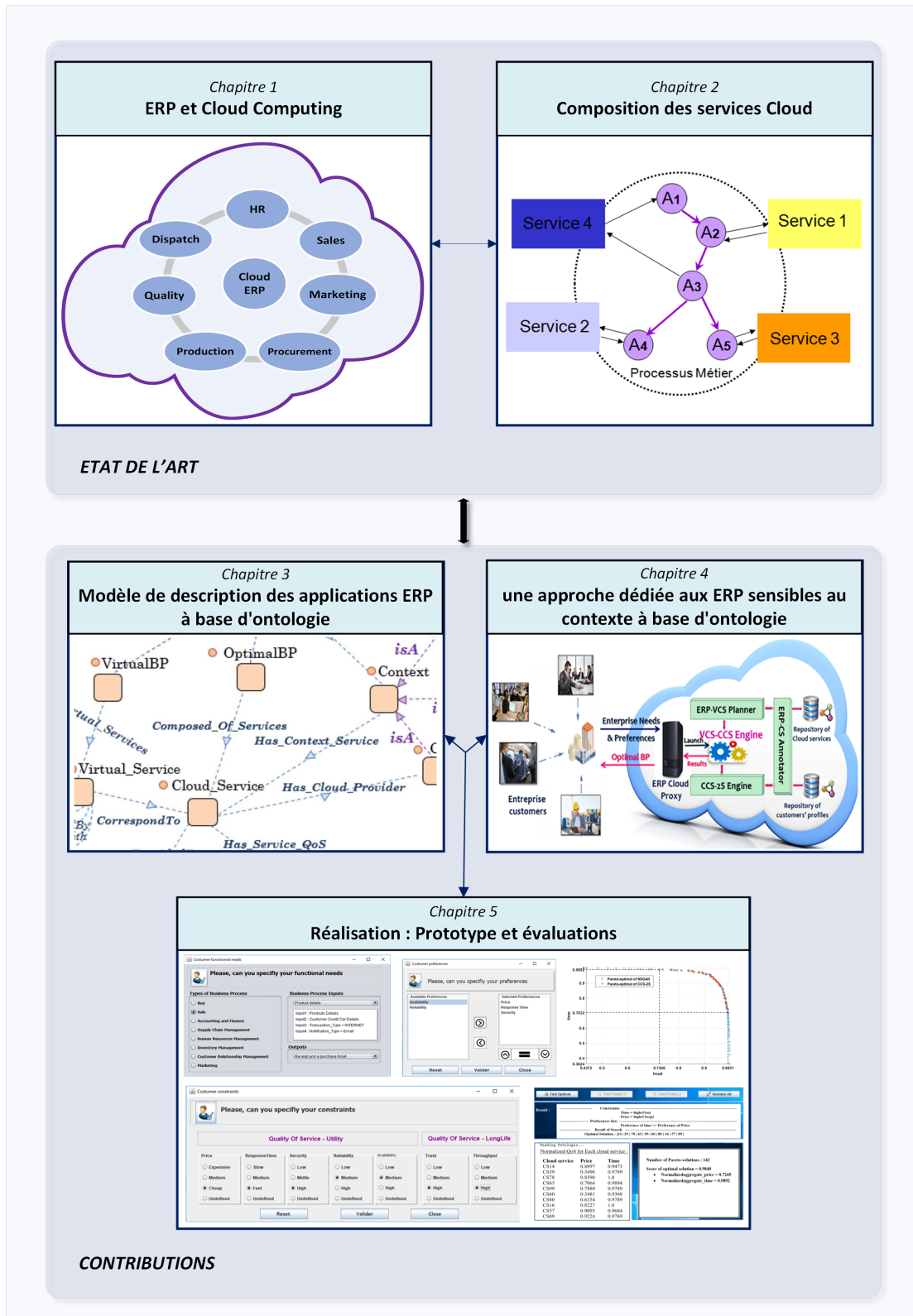
la composition de services Cloud. Il décrit également, une synthèse des principaux travaux connexes dans le domaine de la composition et l'optimisation des services Cloud.

Le chapitre 3 présente une nouvelle ontologie pour la description sémantique et la gestion efficace des services Cloud afin d'engendrer automatiquement un processus d'affaire ERP sensible au contexte via la composition de services Cloud. Cette ontologie prend le rôle du guide tout au long du processus de composition des services Cloud afin de satisfaire les besoins dynamiques du client. En outre, elle facilite l'interopérabilité sémantique des services Cloud hétérogènes représentées par les différents fournisseurs Cloud ainsi que l'interprétation des qualités de services.

Le chapitre 4 décrit une nouvelle approche qui vise à produire un système à base d'ontologie qui permet de fournir aux entreprises un ERP personnalisé sensible au contexte. Ce système permet de générer automatiquement un processus d'affaire en tant que combinaison de services virtuels en utilisant des règles d'inférence. Ensuite, il offre à l'entreprise un CCS (Composite Concrete Service) optimal en termes de QoS qui respecte ses contraintes et préférences contextuelles en utilisant un algorithme à deux phases. La première phase basée sur l'algorithme NSGA-II, permet de générer un ensemble de solutions Pareto-optimal qui respecte les contraintes de l'entreprise. La deuxième sert à sélectionner la solution optimale finale à partir des solutions engendrées par la première phase en fonction des préférences de l'entreprise. Ce chapitre présente également une extension de cette nouvelle approche basée sur les agents pour améliorer la performance du système.

Le chapitre 5 est consacré à la réalisation et à la validation expérimentale de nos propositions. Ce chapitre contient deux parties. La première partie présente un prototype implémenté en JAVA pour la génération de systèmes ERP personnalisés sensibles au contexte. Elle présente également les mécanismes de l'implémentation du prototype ainsi que son fonctionnement sur une étude de cas. La deuxième partie illustre l'évaluation et la validation expérimentale de notre approche. Elle décrit également des comparaisons entre nos propositions et d'autres approches existantes en se basant sur des métriques et critères d'évaluations. Enfin, ce manuscrit de thèse est clôturé par une conclusion générale décrivant un constat du travail effectué et quelques perspectives considérées comme une continuation intéressante de notre thème de recherche.

La figure suivante résume les différents chapitres présentés dans notre manuscrit.



# **Première partie**

## **Etat de l'art**

# Chapitre 1

## ERP et Cloud Computing

### Sommaire

---

|          |                                                     |           |
|----------|-----------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                 | <b>9</b>  |
| <b>2</b> | <b>Cloud Computing</b>                              | <b>9</b>  |
| 2.1      | Définitions                                         | 11        |
| 2.2      | Caractéristiques du Cloud Computing                 | 12        |
| 2.3      | Modèle de service                                   | 13        |
| 2.4      | Modèle de déploiement                               | 16        |
| <b>3</b> | <b>Enterprise Resource Planning (ERP)</b>           | <b>18</b> |
| 3.1      | C'est quoi un ERP ?                                 | 18        |
| 3.2      | Pourquoi un ERP ?                                   | 19        |
| 3.3      | Caractéristiques d'un ERP                           | 19        |
| 3.4      | Implémentation des systèmes ERP                     | 22        |
| 3.5      | ERP sur le marché                                   | 23        |
| 3.6      | Les avantages et les inconvénients des systèmes ERP | 26        |
| 3.7      | Les architectures ERP                               | 29        |
| <b>4</b> | <b>ERP et Cloud Computing</b>                       | <b>33</b> |
| 4.1      | Cloud ERP hébergé                                   | 33        |
| 4.2      | Cloud ERP en mode SaaS                              | 34        |
| <b>5</b> | <b>Sensibilité au Contexte</b>                      | <b>36</b> |
| 5.1      | Définitions                                         | 36        |
| 5.2      | Catégories du contexte                              | 37        |
| 5.3      | Applications sensibles au contexte                  | 37        |
| 5.4      | Contexte et ERP                                     | 38        |
| <b>6</b> | <b>Conclusion</b>                                   | <b>38</b> |

---

## 1 Introduction

Dans ces dernières décennies, une des clés majeures pour la réussite et l'optimisation des gains de l'entreprise est la manière d'organiser et de gérer ses ressources. Avec la prolifération des systèmes d'information (SI), la majorité des entreprises actuelles optent des logiciels pour gérer leurs activités afin d'augmenter la concurrence économique. Les grandes firmes des SI offrent des solutions intégrales pour la gestion efficace des activités de l'entreprise appelées "Enterprise Resource Planning (ERP)". Ces ERP sont des progiciels qui contiennent les différents modules nécessaires pour l'automatisation de la gestion intégrale de l'entreprise. D'une autre part, le Cloud Computing est fortement émergé ces dernières années dans le domaine IT (Internet Technology) grâce à ses multitudes avantages, notamment ; la disponibilité des ressources Cloud n'importe où , n'importe quand et avec n'importe quelle machine.

Cependant, les entreprises ont des caractéristiques différentes telles que l'échelle, la topologie et les moyens financiers. De plus, chaque entreprise a ses propres activités personnelles. Ce qui mène l'entreprise à chercher un ERP personnalisé pour satisfaire ses besoins spécifiques. D'autres entreprises ont un caractère évolutif (c-à-d que leurs activités changent au fil du temps) sont constamment à la recherche de solutions dynamiques et plus flexibles capables de s'adapter à leurs évolutions.

Dernièrement, plusieurs firmes proposent des ERP Cloud avec des différents modes. Un ERP Cloud offre à l'entreprise des solutions à certains problèmes tels que la mobilité, la modularité, l'accessibilité et la flexibilité. Cependant, ces ERP ne sont pas assez flexibles pour répondre à l'évolution continue du contexte de l'entreprise et de l'environnement.

Dans ce chapitre nous allons présenter les principaux axes du domaine de notre thème de recherche. Nous commençons d'abord par la description du Cloud Computing, ensuite nous allons focaliser sur les applications ERP et finalement, nous allons décrire et expliquer la sensibilité au contexte et son rapport avec l'ERP.

## 2 Cloud Computing

Comme tous les concepts relevant autant de l'économie que de la technologie, il est difficile de dire avec précision quand a été inventé le Cloud Computing. Certains pensent que le concept de Cloud Computing est le même que ce que John McCarthy appelait "la capacité de fournir et d'organiser le calcul en tant qu'*utilité*" dans les années 1960. Les principales caractéristiques du Cloud Computing étaient également discutées par Parkhill [30]. D'autre part, le terme Cloud et son symbole graphique ont été utilisés pendant des décennies dans la

littérature de réseau informatique ; d'abord pour la représentation des grands réseaux ATM (Asynchronous Transfer Mode) dans les années 1990, puis pour la description d'Internet "un grand nombre d'ordinateurs distribués".

Même si le concept n'est pas nouveau, il existe aujourd'hui des centaines voire des milliers de tentatives pour définir le "Cloud Computing". Par exemple, Vaquero et *al.* [31] ont comparé 22 différentes définitions dans une tentative de fournir une définition unifiée. Certaines de ces définitions sont générales : "les applications livrées en tant que services sur Internet et le matériel et les logiciels de système dans les centres de données qui fournissent ces services "[32]. D'autres sont plus spécifique : "Un Cloud est un type de système parallèle et distribué constitué d'une collection des ordinateurs interconnectés et virtualisés dynamiquement provisionnés et présentés comme une ou plusieurs ressources informatiques unifiées basées sur les ententes de niveau de service (SLA : Service Level Agreement)<sup>1</sup> établis par voie de négociation entre le fournisseur de services et les consommateurs". Le fait que le Cloud Computing ne soit pas un terme purement technique, ainsi que la participation à la caractérisation du Cloud Computing par un grand nombre de technologies interdisciplinaires, sont les raisons derrière toutes les différentes définitions. La figure 1.1 représente l'architecture générale du Cloud Computing.

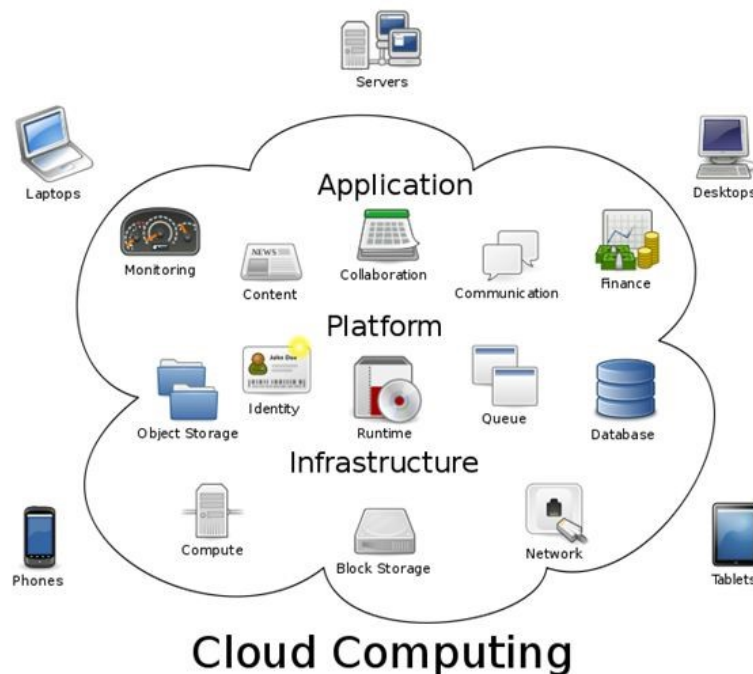


FIGURE 1.1 Architecture du Cloud Computing [33]

1. *entente de niveau de service* est la traduction en français de l'expression "Service Level Agreement".

## 2.1 Définitions

L'industrie des technologies de l'information a cherché à normaliser la définition de Cloud Computing. L'une des premières définitions normalisées est celle de Forrester Research, Inc<sup>2</sup> ; dans laquelle ils ont défini le Cloud Computing comme suit :

"Le Cloud Computing est une capacité informatique standardisée (services, logiciels ou infrastructures) fournis via Internet dans un mode *pay-per-use, self-service* " [34].

La définition de Forrester se concentre sur les modèles de service et modèle d'affaires du Cloud. Cependant, elle ignore les modèles de déploiement parce qu'elle ne croit pas aux Cloud privés en tant qu'un moyen de réduction des coûts. Selon un document de Forrester Research de juillet 2010 intitulé «Vous n'êtes pas prêt pour l'interne Cloud " [35], la construction d'un centre de données Cloud privé qui satisfait tous les exigences légales est une tâche ardue. La définition normalisée la plus récente et la plus acceptée du Cloud Computing par l'Institut National des Normes et de la Technologie<sup>3</sup> (NIST) est la suivante :

"Le Cloud Computing est un modèle qui permet un accès réseau pratique et sur demande à un pool partagé de ressources informatiques configurables (par exemple, des réseaux, des serveurs, du stockage, des applications et des services) qui peut être rapidement approvisionné et disponible sans trop d'efforts de gestion ou d'interaction d'opérateurs. Ce modèle de Cloud favorise la disponibilité. Il est composé de cinq caractéristiques essentielles, de trois modèles de service et de quatre modèles de déploiement" [36].

La définition du NIST est relativement technique et couvre tous les modes des services Cloud (IaaS, PaaS, SaaS) et les modèles de déploiement (public, privé, hybride, communautaire). La définition du NIST est concise, précise et distingue entre les caractéristiques et les technologies habilitantes (c-à-d la virtualisation) et entre les caractéristiques principales et les caractéristiques dérivées (élasticité rapide par rapport à l'extensibilité massive). Néanmoins, la définition ignore le modèle économique du Cloud Computing, qui est le principal moteur pour passer au Cloud. Cependant, le "pay-per-use" modèle n'a pas été omis de la définition par ignorance, mais parce que le NIST a essayé de couvrir tous les modèles de déploiement Cloud, y compris le "Cloud privé", qui n'implique pas nécessairement l'utilisation du "pay-per-use".

Les principaux acteurs du Cloud Computing sur le marché sont : Amazon, Google, IBM et Microsoft, ainsi que des entreprises spécialisées comme Equinix et RackSpace.

---

2. Forrester Research est une entreprise indépendante qui fournit à ses clients des études de marché sur l'impact des technologies dans le monde des affaires.

3. NIST est l'acronyme de "National Institute of Standards and Technology".

La définition du Cloud Computing par les fournisseurs Cloud est généralement d'ordre commerciale. Par exemple, le Cloud Computing est défini par Amazon comme suit :

### **La définition du Cloud Computing par Amazon :**

*Le Cloud Computing est la livraison à la demande de puissance de calcul, de stockage sur base de données, d'applications et d'autres ressources informatiques par le biais d'une plate-forme de services de Cloud sur Internet, avec une tarification progressive [37].*

## **2.2 Caractéristiques du Cloud Computing**

Les principales caractéristiques du Cloud Computing sont [36] :

1. *Libre-service à la demande* : dans un environnement de type Cloud Computing, il est possible à un utilisateur de consommer les services ou ressources sans pour autant devoir faire une demande d'intervention auprès de son fournisseur : équipe IT ou fournisseur externe (par exemple, un développeur qui souhaite tester son application sur une machine virtuelle représentative d'un poste standardisé de son entreprise peut, seul et au travers d'un portail Web, provisionner et utiliser une machine sans devoir solliciter l'équipe IT).
2. *Accessibilité sur l'ensemble du réseau* : Les services sont disponibles sur le réseau et sont accessibles via des mécanismes standards qui favorisent l'utilisation par des plates-formes client hétérogènes (PC, Mac, TV, Tablette, Smartphone, etc.).
3. *Mise en commun des ressources*<sup>4</sup> : les ressources informatiques du fournisseur sont mises en commun pour servir plusieurs consommateurs à l'aide d'un modèle à locataires multiples, avec différentes ressources physiques et virtuelles attribuées dynamiquement et réaffectées en fonction de la demande des consommateurs. Il existe un sentiment d'indépendance de localisation en ce sens que le client n'a généralement aucun contrôle ou connaissance sur l'emplacement exact des ressources fournies mais peut spécifier un emplacement à un niveau d'abstraction supérieur (par exemple, pays, état ou centre de données). Des exemples de ressources incluent le stockage, le CPU, la mémoire et la bande passante réseau.
4. *Élasticité* : grâce au Cloud, il est possible de disposer de plus de ressources très rapidement pour soutenir une forte demande (par exemple, pour garantir une bonne expérience d'achat aux clients sur une plateforme Web de e-commerce durant les fêtes de fin d'année). Inversement, au-delà de l'approvisionnement de ressources, il

---

4. Mise en commun des ressources ou "Pooling" en anglais.

est possible avec le Cloud de diminuer les ressources utilisées (par exemple, en cas de baisse d'activité sur cette même plateforme Web de e-commerce) si celles-ci sont supérieures à ce qui est réellement nécessaire.

5. *Service mesurable et facturable* : le fournisseur de la solution de Cloud est capable de mesurer de façon précise la consommation des différentes ressources (CPU, stockage, bande passante, etc.). Cette mesure lui permet ensuite de facturer le client selon l'usage.

La liste ci-dessus est basée sur la définition du NIST. La liste décrit le Cloud Computing basé sur ce qui est actuellement disponible sur le marché, et représente les principales caractéristiques de Cloud Computing en général. De nombreuses autres caractéristiques peuvent être ajoutées à cette liste à l'avenir. Des listes complètes de caractéristiques peuvent être établies pour chaque couche et chaque type de service fourni dans l'environnement Cloud. Par exemple, au niveau de l'application, une caractéristique possible du service de Cloud est qu'un service doit être portable, pré-configurable ou adaptable.

## 2.3 Modèle de service

Un modèle de service Cloud représente une abstraction de haut niveau en couches des principales classes de services fournies par le modèle Cloud Computing, et comment ces couches sont liées entre elles. Cette séparation entre les couches permet à chaque fournisseur de Cloud de se concentrer sur les services de base fournis. Ainsi, le modèle de service Cloud permet la réutilisation des services des couches inférieures en suivant l'ensemble des mécanismes et protocoles de communication standard entre les différentes couches. Les couches diffèrent en fonction de la portée de gestion couverte par le fournisseur [38], ce qui signifie qu'un utilisateur des couches supérieures ne peut pas contourner les interfaces fournies par la couche inférieure, de manière à accéder directement aux ressources (voir figure 1.2). Cette séparation n'aide pas seulement dans l'intégration de service mais permet également d'avoir une architecture distribuée, évolutive et tolérante aux pannes. Les différentes couches avec différents niveaux d'abstraction permettent aux fournisseurs de Cloud d'avoir une meilleure gestion des ressources, ainsi qu'une plus grande contrôlabilité et sécurité.

Les principales couches du modèle de services du Cloud Computing sont [36] : Software as a Service (SaaS), Platform as a Service (PaaS) et Infrastructure as a Service (IaaS)<sup>5</sup>.

---

5. On utilise souvent les acronymes anglais : "SaaS" pour logiciel sous forme de service, "PaaS" pour plateforme sous forme de service et "IaaS" pour Infrastructure sous forme de service.

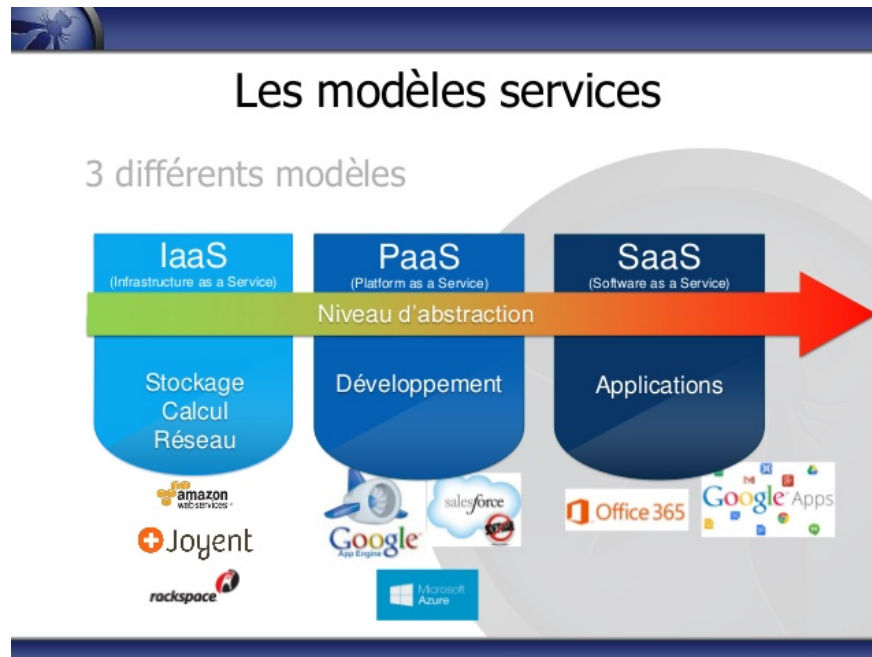


FIGURE 1.2 Modèle de service [39]

### 2.3.1 Software as a Service «SaaS»

Elle donne aux utilisateurs l'accès aux logiciels d'un fournisseur Cloud. Les utilisateurs n'installent pas les applications sur leur terminal. Elles résident sur un réseau Cloud distant auquel ils accèdent par le web ou par une API. Avec les applications, les utilisateurs peuvent stocker et analyser des données et collaborer sur des projets. Le modèle *SaaS* peut être caractérisé par :

- Les logiciels et applications sont proposés aux utilisateurs par les fournisseurs de SaaS selon un modèle par abonnement.
- Les utilisateurs n'ont pas à gérer, installer ou mettre à niveau les logiciels ; ce sont les fournisseurs qui s'en chargent.
- Les données sont sécurisées dans le Cloud ; une panne d'un équipement n'entraîne pas de perte de données.
- L'utilisation des ressources peut être dimensionnée en fonction des besoins de l'entreprise.
- Les applications sont accessibles depuis n'importe quel terminal connecté à Internet et depuis virtuellement n'importe où dans le monde.

De bons exemples de SaaS sont les logiciels de messagerie au travers d'un navigateur comme Gmail ou Yahoo mail. Ces infrastructures fournissent le service de messagerie à des centaines

de millions d'utilisateurs et à des dizaines de millions d'entreprises. D'autres exemples de SaaS tel que Office365 et GoogleApps qui sont très populaires et très utilisés dans le monde du Cloud.

### 2.3.2 Platform as a Service «PaaS»

Elle fournit aux utilisateurs un environnement Cloud dans lequel ils peuvent développer, gérer et mettre à disposition des applications. Outre le stockage et les autres ressources informatiques, les utilisateurs peuvent utiliser une suite d'outils disponibles pour développer, personnaliser et tester leurs applications [38]. Le PaaS est caractérisé par les points suivants :

- offre une plateforme avec des outils pour tester, développer et héberger les applications dans le même environnement.
- permet aux entreprises de se concentrer sur le développement sans avoir à se soucier de l'infrastructure sous-jacente.
- exempte les utilisateurs de gérer la sécurité, les systèmes d'exploitation, les logiciels de serveur et les sauvegardes.
- facilite le travail collaboratif même lorsque les équipes travaillent à distance.

Comme exemples de PaaS, nous citons "Intuit Quickbase" qui permet de déployer ses applications bases de données en ligne, ou "Google Apps Engine (GAE)" pour déployer des services Web. Dans ces plateformes, l'utilisateur de ces services n'a pas à gérer des serveurs ou des systèmes pour déployer ses applications en ligne et dimensionner des ressources adaptées au trafic.

### 2.3.3 Infrastructure as a Service «IaaS»

C'est une offre de Cloud Computing dans laquelle un fournisseur offre aux utilisateurs l'accès à des ressources informatiques comme des serveurs, du stockage et de l'équipement de réseau [38]. Les entreprises utilisent leurs propres plateformes et applications dans l'infrastructure du fournisseur. Ses principales caractéristiques sont :

- les utilisateurs paient pour l'IaaS à la demande et ils n'ont pas besoin d'acheter le matériel.
- l'infrastructure est évolutive en fonction des besoins de traitement et de stockage.
- les entreprises économisent les coûts d'achat et de maintenance de leur propre matériel.
- la sécurité de l'infrastructure est assurée par le fournisseur du Cloud et n'est pas à l'utilisateur de s'en occuper.

- la virtualisation des tâches d'administration, ce qui libère du temps pour d'autres activités à plus forte valeur ajoutée.

L'exemple emblématique de ce type de service est Amazon Web Services qui fournit du calcul (EC2), du stockage (S3, EBS), des bases de données en ligne (SimpleDB) et quantité d'autres services de base. Il est maintenant imité par de très nombreux fournisseurs.

## 2.4 Modèle de déploiement

Un modèle de déploiement Cloud Computing est un modèle qui décrit l'environnement où les applications et les services de Cloud peuvent être installés, afin d'être disponibles pour les consommateurs. Un environnement de déploiement représente l'emplacement physique, les infrastructures, les contraintes de la plateforme, ainsi que tout ce qui peut affecter les mécanismes d'accès aux applications déployées. Il existe quatre principaux modèles de déploiement Cloud Computing : public, privé, hybride et communautaire [38] (voir figure 1.3).

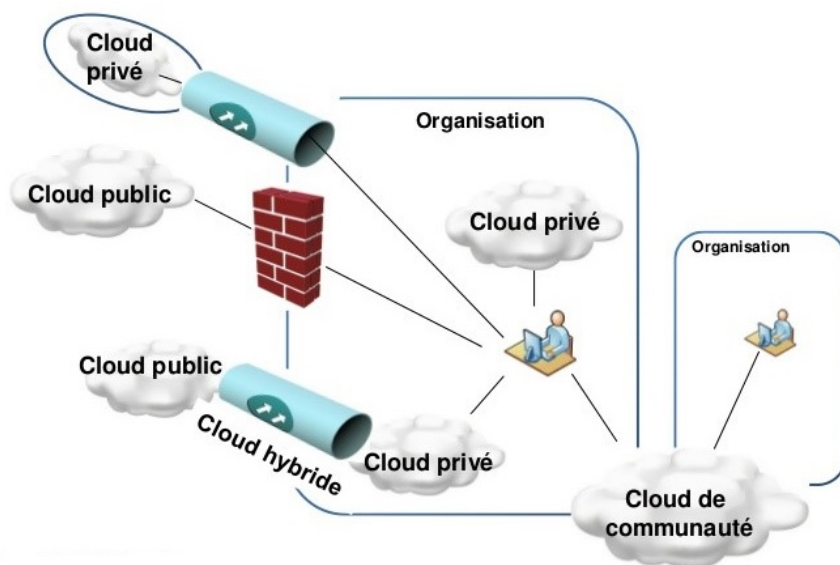


FIGURE 1.3 Modèle de déploiement [40]

### 2.4.1 Cloud publique

Un Cloud public ou Cloud externe est un modèle ouvert dans lequel les infrastructures sont fournies par un tiers (les fournisseurs de Cloud). Les services d'infrastructure et de plateforme sont fournis au public selon l'accord de niveau de service (SLA : Service Level Agreement) entre le fournisseur et le consommateur. Ce type de partage de ressources d'infrastructure

entre plusieurs organisations ou consommateurs est appelé le modèle de multi-location. Le Cloud public est le choix le moins coûteux pour l'hébergement d'applications. Cependant, l'absence d'un modèle de confiance entre les fournisseurs de Cloud et les consommateurs est le principal obstacle à ce modèle.

#### **2.4.2 Cloud privé**

Un Cloud privé ou Cloud interne est un Data-Center appartenant à un fournisseur d'applications Cloud, dans lequel l'infrastructure et la plateforme sont administrées entièrement par le fournisseur de l'application. Cela élimine le besoin d'un modèle de confiance et offre plus de flexibilité. Les organisations peuvent mettre en œuvre leurs propres politiques en matière de confidentialité, de sécurité et de mécanismes d'accès. Cependant, cette option est coûteuse en termes de ressources, et la main d'œuvre nécessaire pour gérer ces ressources.

#### **2.4.3 Cloud hybride**

Un Cloud hybride est une combinaison du Cloud public et Cloud privé. Un Cloud hybride est moins cher qu'un Cloud privé; il élimine également le besoin d'un modèle de confiance. Cependant, avoir des Cloud publics et Cloud privés fonctionnant ensemble nécessite l'interopérabilité et la portabilité des applications et les données pour faciliter la communication entre les modèles.

#### **2.4.4 Cloud communautaire**

Un Cloud communautaire est utilisé par plusieurs organisations qui ont des besoins communs. Il peut être utilisé pour des applications génériques, mais qui ont des spécificités adaptées aux contraintes du groupe; c'est par exemple le cas du Cloud communautaire construit aux USA par la GSA<sup>6</sup> pour les organisations qui dépendent du gouvernement américain. Il peut aussi héberger une application métier très spécialisée, mais commune à de très nombreuses entreprises, qui décident de fédérer leurs efforts en construisant, ensemble, un Cloud pour l'héberger et la gérer. Amadeus<sup>7</sup> et CMed<sup>8</sup> sont deux des exemples de Cloud communautaires.

---

6. La GSA (General Services Administration) est un Cloud communautaire pour les organisations gouvernementales américaines.

7. Amadeus est un Cloud communautaire de l'industrie des voyages démarré en 2011.

8. CMed est un Cloud communautaire pour laboratoires pharmaceutiques démarré en 2010.

### 3 Enterprise Resource Planning (ERP)

Les ERP<sup>9</sup> sont des systèmes complets de solutions logicielles qui cherchent à intégrer l'ensemble des processus et des fonctions d'une entreprise dans l'ordre de présenter une vision globale de l'entreprise à partir d'une seule architecture informatique. Ce sont des grands logiciels d'applications intégrés qui ont été en usage depuis les années 1970. Les plus grandes organisations internationales ont adopté l'ERP auparavant, ainsi que les petites et moyennes entreprises (PME)<sup>10</sup> ont besoin de plus en plus de solutions ERP personnalisées pour améliorer leur concurrence [41].

#### 3.1 C'est quoi un ERP ?

Un ERP ou "Progiciel de Gestion Intégré (PGI)" est un logiciel qui permet de gérer l'ensemble des activités d'une entreprise en intégrant l'ensemble de ses fonctions telles que : la gestion des ressources humaines, la gestion comptable et financière, les relations client "Customer Relationship Management (CRM)", les achats, la gestion des stocks et inventaires, la gestion de la production, la distribution, l'approvisionnement et le e-commerce [1]. L'ERP est basé sur la construction des applications informatiques de l'entreprise sous forme de **modules indépendants**. Ces modules partagent **une base de données commune**, permettant la communication de données entre les applications [1].

Un ERP se caractérise par l'usage systématique d'un moteur de workflow<sup>11</sup>. Il permet d'automatiser un flux d'informations au sein d'une organisation de manière transparente. Il permet ainsi de propager les données au sein d'un SI entre tous ses modules requis [1].

Un ERP est un logiciel intégré orienté processus d'affaires. Il aide les gestionnaires à mieux comprendre la situation de leurs organisations.

Un ERP est destiné à la gestion globale des différents flux de l'entreprise aux niveaux : stratégique, tactique et opérationnel. Il met en commun, pour les diverses entités et fonctions, l'ensemble des données nécessaires à cette gestion dans une base de données unique [42].

#### Définition formelle :

*Un ERP est un système d'information composé de plusieurs applications partageant une même base de données, par le biais d'un moteur de workflow paramétrable et automatisé [1].*

9. Depuis le début des années 2000, le terme PGI (Progiciel de Gestion Intégré) est tombé en désuétude, l'acronyme ERP étant largement adopté dans le monde des entreprises.

10. En anglais Small and Medium Enterprise (SME)

11. Le Workflow se traduit par le « flux d'activités ». Il s'agit d'un processus qui permet d'automatiser la circulation des flux d'informations dans une entreprise.

### 3.2 Pourquoi un ERP ?

Les principales causes de la mise en place d'un ERP sont les suivantes [1] :

- *L'intégrité et l'unicité du SI* ; c'est-à-dire qu'un ERP permet une vue et une logique unique à travers sa base de données. Ceci se traduit par le fait qu'il peut exister plusieurs bases de données "physiques" mais celles-ci respectent la même structure. En d'autres termes, un ERP permet d'éviter la redondance d'information entre différents systèmes fonctionnels de l'entreprise.
- L'utilisateur a la possibilité de récupérer ou d'enregistrer des données de manière immédiate et transparente. Ainsi que les mises à jour dans la base de données sont effectuées en temps réel et propagées aux modules concernés.
- Un ERP est un outil multi-langue et multi-device <sup>12</sup>, il est donc adapté au marché mondial, en particulier aux multinationales.
- *Pas d'interface entre les modules* ; c'est-à-dire que les traitements sont synchronisés et les processus de gestion sont optimisés. De même, la maintenance corrective est simplifiée car celle-ci est assurée directement par l'éditeur et non plus par le service informatique de l'entreprise.
- Un ERP permet de contrôler les stocks, qui est un élément important pour la plupart des entreprises car les stocks coûtent cher.

Par conséquent, Une gestion optimale des processus d'affaires est de dépenser moins d'argent et d'éviter plus de risques par le biais des ERP. Un ERP permet aux entreprises une capitalisation d'affaires, un accès instantané aux informations et une exploitation efficace des ressources et ses rapports d'affaires. Il permet au chef d'entreprise de se concentrer sur l'analyse et la prise de décision sans préoccuper des détails techniques.

### 3.3 Caractéristiques d'un ERP

#### 3.3.1 Architecture client/serveur d'un ERP

La plupart des ERP utilisent l'architecture client/serveur pour leur déploiement (figure 1.4). La base de données et les modules de l'ERP sont installés sur le serveur. Il est accessible par tous les clients (postes) selon leurs privilèges. De plus, les ERP sont compatibles pack Office, en particulier pour Powerpoint et Excel. En effet, le premier étant utile pour personnaliser les bureaux ERP en fonction de l'entreprise et le second pour effectuer les imports/exports de données. Enfin, les ERP sont aussi compatibles avec des outils de reporting

---

12. Le multi-device représente un concept pour appuyer le fait que les utilisateurs usent de multiples "devices" ou appareils

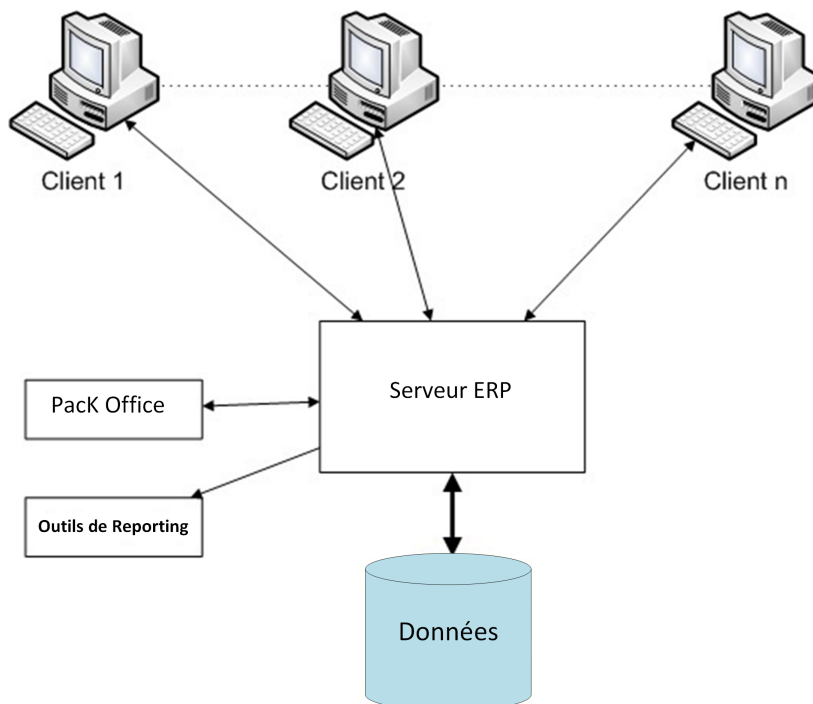


FIGURE 1.4 Architecture client/serveur d'un ERP [43]

( CrystalReport <sup>13</sup> en général). Le reporting étant utilisé en particulier pour le module de gestion de relations client (CRM).

### 3.3.2 Architecture modulaire d'un ERP

Un ERP est un ensemble de modules qui interagissent entre eux en partageant une base de données commune. Cette modularité est affirmée dans le sens où chaque module peut fonctionner seul mais également capable de fonctionner en relation avec les autres. Les applications modulaires telles que les ERP permettent d'être sûr de la compatibilité des modules entre eux, ils s'imbriquent comme des blocs et fonctionnent ensemble (pas de vérification de compatibilité). La figure 1.5 représente les modules principaux d'un ERP.

Les ERP sont basés sur une architecture modulaire centralisée autour d'un composant base de données. Toutefois, il faut étendre la conceptualisation de cette architecture pour s'entendre sur une architecture verticale spécifique à un métier particulier. Les entreprises qui ont besoins de fonctionnalités spécifiques sont obligés d'effectuer un grand travail de configuration et de spécialisation d'appropriation des modules [45].

13. Crystal report est un analyseur/interpréteur de données, il permet de générer des rapports à partir des données dans le cadre de l'informatique décisionnelle.

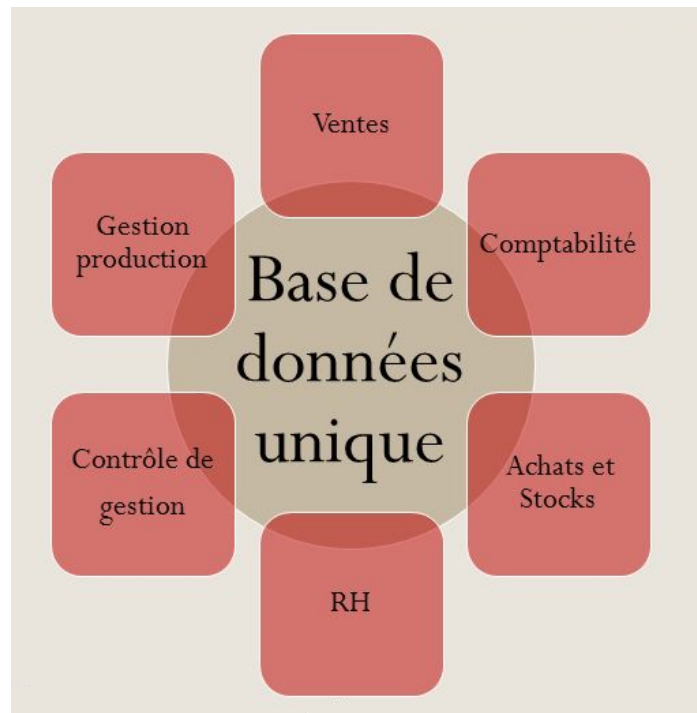


FIGURE 1.5 Architecture modulaire d'un ERP [44]

### 3.3.3 Fonctionnement d'un ERP

Dans un ERP, les opérations doivent être synchrones entre les entités de l'organisation. En d'autres termes, si un utilisateur d'une entité change une donnée, la donnée modifiée doit être immédiatement vue par un utilisateur d'une autre entité. Par exemple, dans une entreprise commerciale, dès qu'une facture d'achat est approuvée, il peut affecter immédiatement le coût du produit et les comptes des créditeurs. Du point de vue de l'ERP, la performance de votre organisation est le résultat d'exécution de plusieurs processus d'affaires. Chaque processus couvre un service de l'organisation et implique différentes ressources pour sa réalisation.

**Exemple** Dans une entreprise commerciale, un processus d'affaire de vente des produits peut être le résultat d'exécution de plusieurs processus : vente, comptabilité et achats. Chaque processus implique un ou plusieurs utilisateurs (voir Figure 1.6).

Un processus de vente est créé en fonction de l'ordre de vente. Une commande d'achat est créée en fonction de la facture de réquisition. L'achat est délivré à la suite d'un ordre de vente. Tous les liens de cette chaîne d'actions sont stockés dans un endroit logique (voir

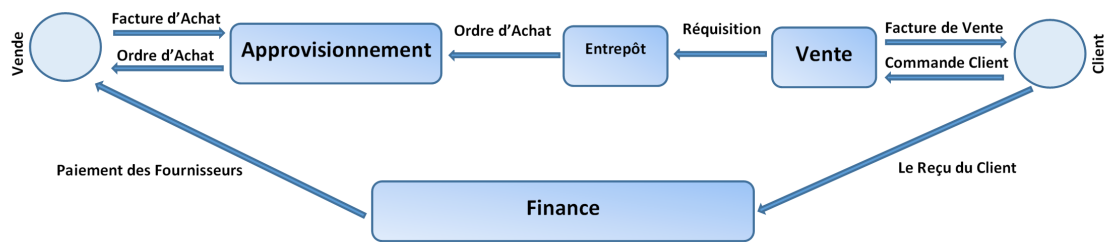


FIGURE 1.6 Processus d'affaire [46]

Figure 1.7). Nous avons une chaîne cohérente de documents qui nous permet de localiser précisément la raison/racine de toute activité de l'organisation à un moment donné.

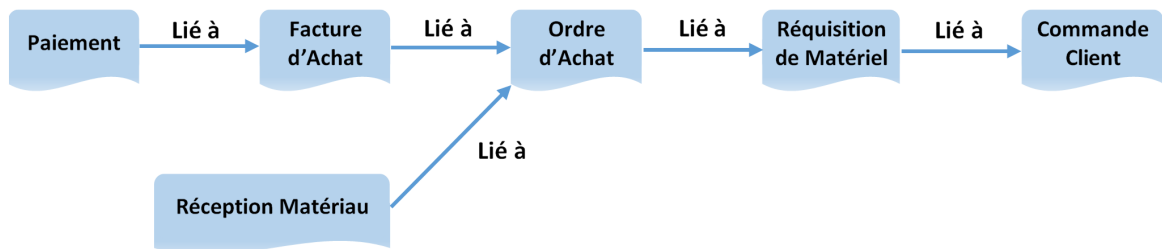


FIGURE 1.7 La chaîne d'activités d'un processus d'affaire [46]

### 3.4 Implémentation des systèmes ERP

La demande de la mise en œuvre de l'ERP dans les organisations augmente rapidement. La stratégie de la mise en œuvre varie en fonction de nombreux fournisseurs proposant des services d'hébergement différents. Tous les fournisseurs suivent une méthodologie de base similaire et passent par plusieurs phases principales qui sont : la préparation et la planification, la conception, le développement, les tests et le déploiement [47–49] afin de mettre en œuvre l'ERP dans un établissement. Chaque phase sera brièvement expliquée comme suit <sup>14</sup> :

- **Préparation et planification**

Cette étape présente la phase d'initiation pour définir le projet et les exigences. Elle comprend l'identification des problèmes, la spécification de la portée, l'ordonnancement du projet et la planification.

- **La conception**

Cette étape permet de définir l'architecture complète et les exigences détaillées. En

14. La description détaillée des phases d'implémentation est trop vaste et sort de l'objectif de notre travail.

outre, en soulignant la conception à développer par les phases suivantes. Des différentes configurations de prototypes sont spécifiées dans cette étape.

- **Le développement**

Les conceptions qui ont été précisées précédemment sont développées dans cette phase. Plusieurs fonctionnalités et configurations seront ajoutées pour finaliser le modèle requis pour la mise en ligne.

- **Les tests**

Dans la plupart des cas, cette phase chevauche la phase de développement. Elle est responsable des tests systématiques selon les lacunes pour satisfaire les besoins du client.

- **Le déploiement**

Dans cette phase, l'installation sera configurée sur les sites du client. Elle implique également d'autres tests du système et de soutien. Après cette étape, le système ERP est prêt à démarrer et l'utilisateur commence à l'utiliser.

Toutes ces phases ne présentent que les étapes de base qui doivent être divisées en sous-phases supportées par de nombreux outils et cadres. Cependant, le succès de la mise en œuvre du système ERP n'est pas garanti. Il y a entre 67% et 90% des systèmes qui ont échoué et environ 35% d'entre eux sont annulés [50]. De nombreuses études et recherches ont été menées sur les facteurs de succès de la mise en œuvre du système ERP. Mais, ils ne sont pas parvenus à un accord sur les principaux facteurs. Depuis, cela peut dépendre de la technologie elle-même et de la façon dont elle est mise en œuvre ou du point de vue de l'utilisateur final [47].

### 3.5 ERP sur le marché

Les fournisseurs d'ERP sont classés dans les catégories : "Catégorie 1, Catégorie 2 ou Catégorie 3" en fonction du type de clients qu'ils desservent. Les trois groupes sont très distincts, notamment, par rapport à la taille et la complexité de leurs solutions.

En règle générale, l'industrie classe un fournisseur ERP de catégorie 1 parmi les fournisseurs qui vendent beaucoup sur le marché de catégorie I1 ; un marché dont les entreprises ont des revenus annuels dépassant le milliard de dollars. Ces entreprises sont invariablement des multinationales présentes dans de nombreuses régions géographiques différentes. Naturellement, les produits ERP de catégorie 1 ont un coût de propriété élevé en raison de leur complexité et des coûts de mise en œuvre et de support. Bien qu'il y ait eu plusieurs fournisseurs de catégorie 1 plus tôt, les fusions et consolidations ont considérablement réduit

la liste. La liste des fournisseurs d'ERP de catégorie 1 est maintenant très réduite et ne comprend que deux entrées : SAP et Oracle [51].

Les fournisseurs de catégorie 2 vendent des produits ERP qui regroupent des entreprises de taille moyenne dont les revenus se situent entre 50 millions et 1 milliard de dollars. Les produits des fournisseurs de catégorie 2 sont spécifiquement conçus pour gérer ce marché et s'adresser à un ou plusieurs sites de déploiement. Naturellement, les solutions de catégorie 2 sont plus faciles à gérer et à prendre en charge, et coûtent également moins cher. Souvent, les solutions de catégorie 2 sont confinées à un secteur industriel spécifique. Ce groupe voit une concurrence considérable et il est composé d'environ 20 entreprises bien connues.

Les fournisseurs de solutions ERP de catégorie 3 ciblent les entreprises qui ont des revenus de 10 à 50 millions de dollars. Les solutions fournies par ces sociétés sont simples à mettre en œuvre et à prendre en charge, et ont donc un coût de propriété inférieur. De nombreux ERP de ce groupe sont des installations à emplacement unique pour une seule verticale. Bien qu'ils soient faciles à gérer et à déployer, le risque est qu'une entreprise puisse rapidement dépasser la solution et que, par conséquent, une forme de migration doit être gardée à l'esprit lorsqu'une petite entreprise à croissance rapide choisit une solution de catégorie 3.

On va citer dans cette section le top 10 des fournisseurs d'ERP.

**SAP** – Fondé en 1972 par cinq anciens ingénieurs d'IBM<sup>15</sup>, SAP est le leader incontesté du marché de l'ERP et le troisième éditeur de logiciels au monde. Sa version actuelle compte plus de 30 000 tables de bases de données relationnelles qui lui permettent de gérer des situations commerciales extrêmement complexes. Bien qu'il soit le numéro un incontesté dans l'espace de la catégorie 1, SAP a parfois été critiqué pour être trop complexe et difficile à gérer. SAP n'est pas assez convenable pour les petites et moyennes entreprises.

**Oracle** – Alors qu'Oracle était autrefois mieux connu pour sa base de données relationnelle, il a été pendant de nombreuses années la base de données de choix pour les applications SAP ERP. Cette situation de coopération existait depuis la fin des années 70. Toutefois, vers 2004, Oracle a commencé à envisager de créer ses propres solutions ERP et, parallèlement, SAP a commencé à proposer ses solutions ERP sur la plate-forme de base de données Microsoft SQL Server. Le premier produit Oracle ERP a été Oracle Financials qui a été lancé sur le marché dès 1989. Cependant, après 2004, Oracle a commencé à devenir un acteur important sur le marché des ERP et il est maintenant le numéro 2 bien établi sur le marché de catégorie 1.

---

15. IBM est la grande firme informatique "International Business Machines".

**Microsoft** – Microsoft Dynamics est principalement axé sur les clients de catégorie 2 dans l'espace ERP. Il fournit des solutions dans différents domaines d'activité, notamment dans le domaine de la gestion de la relation client. Un grand avantage des produits Microsoft est sa grande facilité d'utilisation. Cela vaut aussi pour ses produits ERP.

**Infor** – Infor Global Solutions est une société privée qui a connu une croissance rapide dans l'espace des fournisseurs de catégorie 2 depuis 2002. La société a pris une voie d'acquisition agressive vers la croissance et continue à suivre cette voie avec son acquisition d'ENXSUITE<sup>16</sup> en 2011. Infor a une présence mondiale pour correspondre à l'empreinte du top 3 et a des clients dans 194 pays. Infor propose des solutions dans quatorze domaines différents et a une très bonne présence dans le domaine de l'ERP.

**Epicor** – Commencé en 1984 et travaillant initialement avec DOS, Epicor a ensuite converti ses produits à Windows et a suivi une voie de fusion et d'acquisition pour acquérir des sociétés vendant des produits ERP et ensuite offrir leurs solutions comme un ensemble complet. Epicor est présent dans plus de 150 pays et compte plus de 20 000 clients de catégorie 2 et 3. Epicore aime appeler son ERP "la clé des possibilités pas encore imaginées".

**Lawson** – Acquis par Infor il y a quelques années, Lawson conserve une identité distincte, bien qu'elle affiche le logo Infor sur son site Web. Spécifiquement mentionnant qu'il est adapté aux petites et moyennes entreprises, Lawson est présent dans 68 pays et compte plus de 4 500 installations. Lawson s'adresse à un grand nombre de verticales et l'utilise comme son USP<sup>17</sup>. La simplicité de la solution est un autre domaine clé dans un marché connu pour sa complexité.

**QAD** – Le site Web de QAD montre une chaîne avec le logo du nuage formant l'un des liens afin que nous ayons une idée de ce qui est dans l'esprit de l'entreprise. L'application QAD Enterprise est conçue pour permettre aux utilisateurs ERP débutants de commencer à utiliser un ERP dans leur entreprise avec le moins de problèmes de migration possible. L'entreprise soutient et engage ses clients pour s'assurer que la rentabilité des investissements est obtenu rapidement.

**Sage** – est une société basée au Royaume-Uni et a débuté dans un emploi d'été en 1981 lorsque la première version d'un type de logiciel de comptabilité a été écrite. Cela a grandi dans des versions plus grandes jusqu'à ce qu'en fin de compte, en 1984, Sage

---

16. ENXSuite Inc fournit un système complet et intégré de gestion de la performance énergétique à la demande qui aide les entreprises à analyser et à gérer l'ensemble de leur empreinte énergétique à l'échelle de l'entreprise.

17. USP est l'acronyme pour Unique Selling Proposition ou argument clé de vente.

Software a été lancé en tant qu'entreprise et a obtenu un certain succès. À l'instar de nombreuses autres entreprises de l'espace ERP, Sage a connu plusieurs acquisitions et affirme que «les acquisitions font partie de son ADN». La pollinisation croisée de l'ADN semble avoir eu beaucoup de succès compte tenu du taux de croissance observé par Sage.

**IFS** – Fondé en 1983, IFS se concentre sur la construction de solutions ERP agiles qui utilisent l'architecture SOA. Cela implique une modification facile et une adaptation aux besoins des utilisateurs. IFS est le plus utile des quatre processus stratégiques de base - la gestion des services et des actifs, la fabrication, la chaîne d'approvisionnement et la gestion de projet. Il dispose d'une base d'utilisateurs de plus de 2 000 installations et de clients dans 50 pays. L'une des principales raisons de son succès est sa focalisation sur des verticales spécifiques.

**Consona Corp** – Consona tire son nom de la "consonance avec le client", elle est active dans les domaines de l'ERP, du CRM, de la gestion des connaissances et dans d'autres domaines connexes. La société est une société privée qui a grandi en acquérant un certain nombre de sociétés spécialisées. Elle propose des solutions ERP adaptées aux industries telles que la fabrication de circuits imprimés, la fabrication de fils et câbles métalliques, etc. Systématiquement, une solution ciblée est forcément meilleure qu'un ERP générique lorsqu'elle est utilisée dans l'une des domaines industriels spécialisés.

En 2016, les 10 premiers éditeurs de logiciels ERP représentaient près de 28,5% du marché mondial des applications ERP qui a augmenté de 1,4% pour atteindre près de 82,2 milliards de dollars de revenus de licences, de maintenance et d'abonnement. Le marché des applications ERP comprend les applications ERP "gestion financière" & ERP "services et opérations" [52]. La figure 1.8 montre le top 10 des fournisseurs de logiciels ERP et les parts de marché des applications ERP 2016 selon "Apps Run The World" en décembre 2017.

### 3.6 Les avantages et les inconvénients des systèmes ERP

L'adoption des ERP par les différentes entreprises est très utile pour mieux gérer le déroulement de leurs affaires. Néanmoins, les avantages multiples de ces ERP n'impliquent pas l'absence des inconvénients. Dans ce qui suit, nous allons citer les principaux avantages et inconvénients des ERP :

#### 3.6.1 Avantages

1. une visibilité complète sur tous les processus importants à travers les différents départements d'une organisation (en particulier pour les cadres supérieurs).

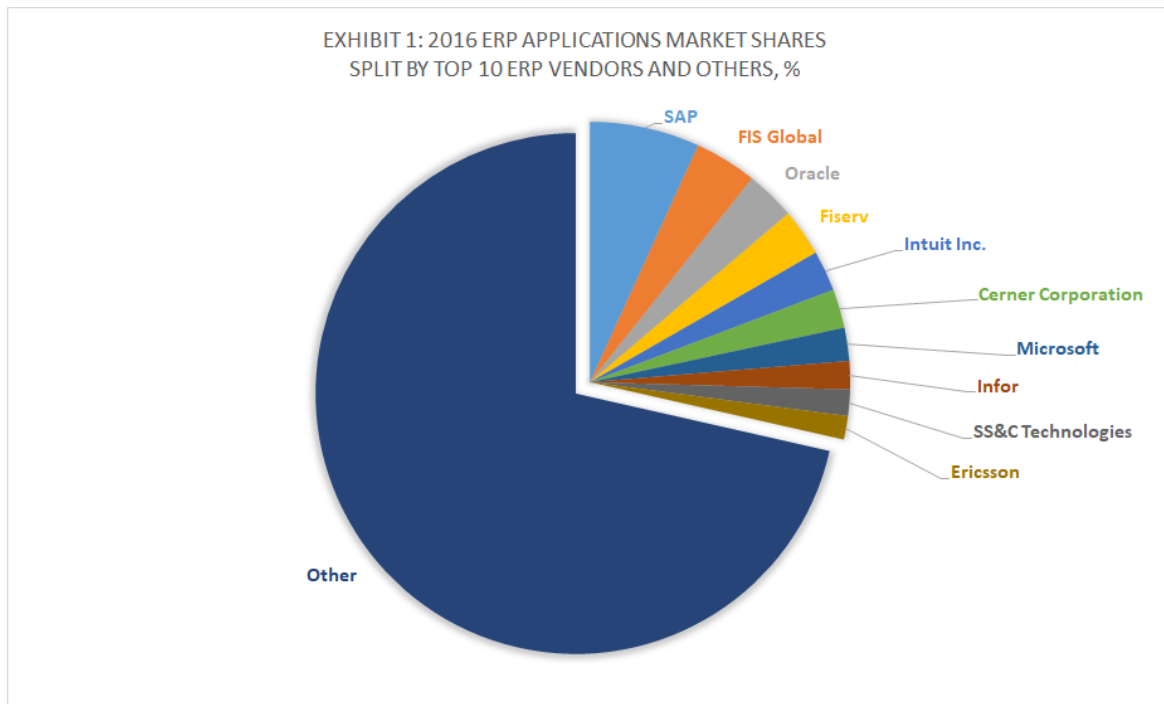


FIGURE 1.8 Top 10 des fournisseurs de logiciels ERP et des parts de marché des applications ERP 2016, Apps Run The World, décembre 2017.

2. workflow automatique et cohérent d'un département / fonction à l'autre, pour assurer une transition harmonique et une exécution plus rapide des processus. Cela garantit également que toutes les activités interdépartementales sont correctement suivies et qu'aucune d'entre elles n'est omise.
3. un système unifié et unique pour analyser les statistiques, statuts etc. en temps réel, à travers toutes les fonctions et les départements.
4. un seul logiciel (ERP) est utilisé dans tous les départements, ce qui implique que les départements individuels n'ont pas besoin d'acheter et de maintenir leurs propres systèmes logiciels.
5. certains fournisseurs ERP peuvent étendre leurs systèmes ERP pour fournir des fonctionnalités de BI<sup>18</sup>, qui peuvent donner des informations générales sur les processus métiers et identifier les domaines potentiels des problèmes et des améliorations.
6. l'architecture modulaire des systèmes ERP permet la mise en œuvre de quelques modules sélectionnés en basant sur les exigences de l'organisation. Si plus de modules sont mis en œuvre, l'intégration entre les différents départements est assurée automatiquement.

18. Business Intelligence : "aide à la décision"

7. un système de base de données implémenté dans le back-end <sup>19</sup> permet un stockage centralisé de toutes les données de l'entreprise.
8. les systèmes ERP sont plus sécurisés car des politiques de sécurité centralisées peuvent leur être appliquées. Toutes les transactions effectuées via les systèmes ERP peuvent être suivies.
9. Les systèmes ERP offrent une meilleure visibilité à l'échelle de l'entreprise et permettent ainsi une collaboration plus rapide entre tous les services.
10. les systèmes ERP sont particulièrement utiles pour une meilleure gestion d'entreprises globales dispersées.
11. un système ERP facilite l'audit en cas de dysfonctionnement, permettant d'identifier facilement les modules concernés ; il est facile de retrouver et d'analyser l'origine de chaque information.

### 3.6.2 Inconvénients

1. le coût du logiciel ERP, de la planification, de la personnalisation, de la configuration, des tests, de la mise en œuvre, etc. est trop élevé "investissement de 6 000 € à plus de 100 000 € par entreprise <sup>20</sup>".
2. les déploiements des ERP prennent beaucoup de temps - les projets peuvent prendre entre 1 et 3 ans (ou plus) pour être terminés et entièrement fonctionnels.
3. trop peu de personnalisation peut ne pas correspondre le système ERP au processus métier de l'entreprise. Ainsi que trop de personnalisation peut ralentir le projet et rendre difficile la mise à jour.
4. La participation des utilisateurs est très importante pour une mise en œuvre réussie des projets ERP et par conséquent, une formation exhaustive des utilisateurs et une interface utilisateur simple peuvent être critiques. Mais les systèmes ERP sont généralement difficiles à apprendre et à utiliser.
5. L'évaluation avant la mise en œuvre du système ERP est essentielle. Si cette étape n'est pas correctement effectuée et que des ressources techniques et commerciales expérimentées ne sont pas disponibles lors de l'évaluation, l'implémentation de l'ERP peut être échouée.

---

19. Un back-end est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-end qui est la partie visible à l'utilisateur

20. Le coût d'un ERP pour les grandes entreprises est à l'échelle de Millions de dollars.

### 3.7 Les architectures ERP

L'architecture du système ERP joue un rôle majeur pour déterminer son succès et sa durabilité pour l'organisation. Il existe aujourd'hui quatre grandes architectures connues pour les systèmes ERP [53–55] ; chacun a ses propres avantages et faiblesses.

#### 3.7.1 Architecture à trois niveaux

L'architecture à trois niveaux est une extension de l'architecture client / serveur<sup>21</sup> à deux niveaux. Elle est composée de trois couches : présentation, application et base de données. La figure 1.9 montre une comparaison des architectures à un, deux et trois niveaux. Comme on peut le voir dans l'architecture à trois niveaux, le client ne communique plus directement avec la base de données, car une couche responsable de l'exécution de la logique métier est introduite. Dans cette architecture, la couche de présentation est uniquement chargée de parcourir les données et de fournir une interface conviviale, ce qui permet à l'utilisateur d'avoir des machines moins puissantes. Cependant, la couche d'application est l'endroit où les données sont récupérées et transférées aux serveurs de base de données dans la couche de base de données. La couche application est également l'endroit où la logique et les règles métier sont implémentées.

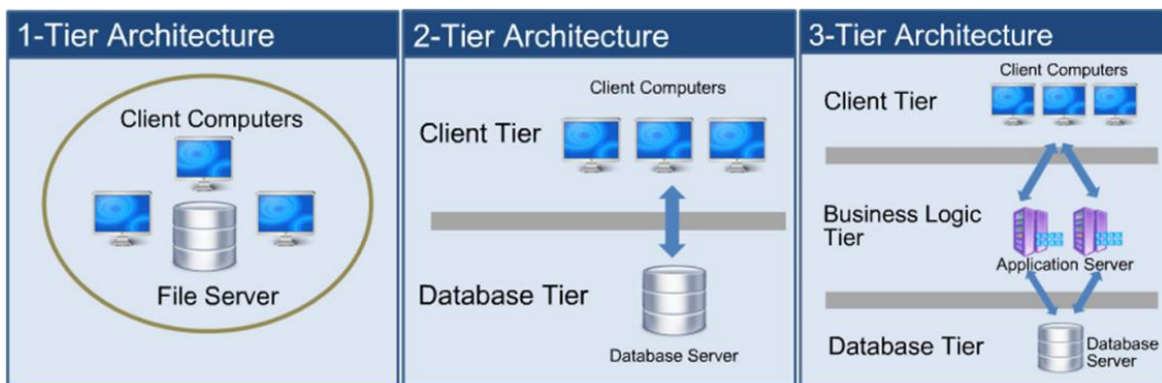


FIGURE 1.9 Les différentes architectures à niveaux [47]

#### 3.7.2 Architecture WEB

Cette architecture est basée sur le web permettant aux utilisateurs distants d'accéder au système ERP. La figure 1.10 montre que l'architecture Web a des couches d'application et de base de données identiques à l'architecture à trois niveaux. Cependant, la couche de

21. L'environnement client-serveur désigne un mode de communication à travers un réseau entre plusieurs programmes.

présentation est divisée en deux parties, les services Web et le navigateur Web pour soutenir la mobilité des dispositifs via Internet. De nombreuses organisations ne construisent pas leur ERP en tant qu'architecture basée sur le Web car l'accessibilité en dehors de l'organisation peut ne pas être une exigence au début. Mais avec la croissance de l'organisation, l'accès à distance devient une nécessité, dans laquelle une architecture Web est une solution. Une architecture Web est une architecture qui n'était pas à l'origine basée sur le Web mais avec de nouvelles demandes supplémentaires, des modifications sont apportées pour la rendre compatible avec le Web. Cela limitera sûrement les fonctionnalités et capacités à distance [47].

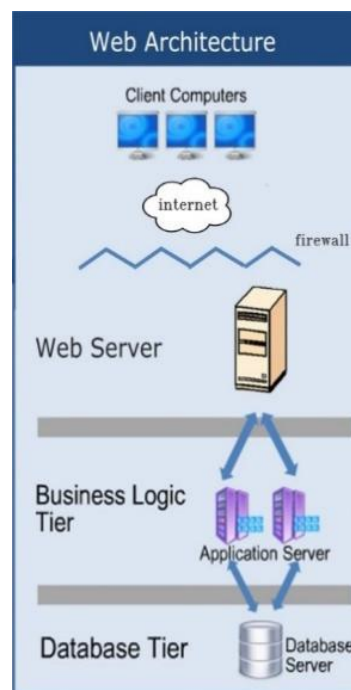


FIGURE 1.10 Architecture web [47]

### 3.7.3 Architecture orientée service

Contrairement aux architectures à trois niveaux et l'architecture Web, l'architecture orientée service SOA (Service Oriented Architecture) n'est pas basée sur une certaine technologie ou technique de mise en réseau. SOA est une approche utilisée pour créer une architecture basée sur l'utilisation de services, indépendamment des méthodologies de mise en réseau utilisées. Avec la croissance des organisations, le nombre de services à fournir augmente. Avec cela, un seul système ne serait pas en mesure de suivre le nombre de transactions. Par conséquent, un système pour chaque service est lancé. SOA permet

aux organisations de personnaliser et de modifier leur système ERP facilement. En plus, l'approche SOA encourage la réutilisation des services existants. La communication entre les différents systèmes pourrait être difficile si des interfaces régulières sont utilisées, car elles seraient très coûteuses à développer. Dans l'espace informatique d'une organisation, le langage XML est un moyen de standardiser les messages communiqués entre les systèmes, comme le montre la figure 1.11. XML<sup>22</sup> est un moyen de décrire des informations ou des données de manière cohérente. Avec XML, l'architecture orientée service devient probable.

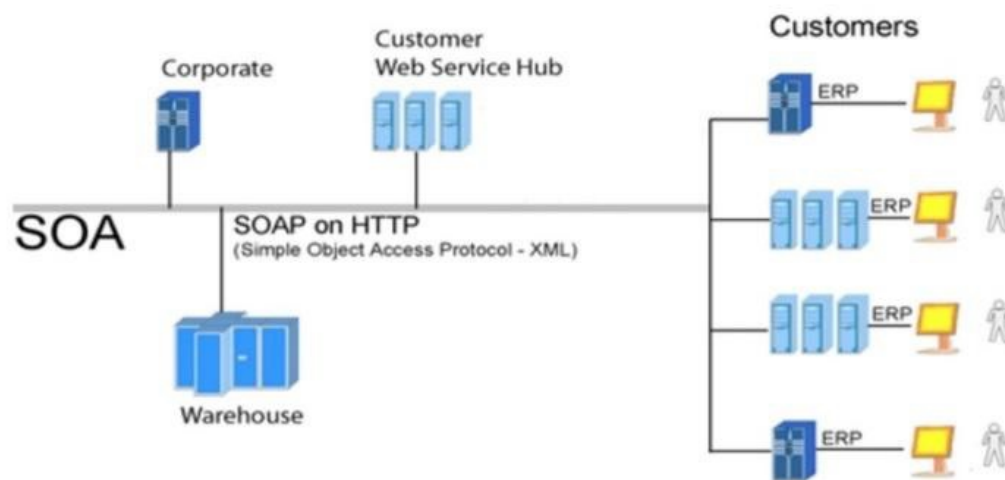


FIGURE 1.11 Architecture SOA [47]

### 3.7.4 Architecture Cloud

Les nouvelles technologies exigent une nouvelle façon de penser et le Cloud Computing n'est pas une exception. Au fur et à mesure que les connexions Internet deviennent plus rapides et plus fiables, la différence d'accès en termes de vitesse et de conservation entre l'établissement local et les données Web diminue considérablement. Grâce à la fiabilité de l'accès, cette nouvelle technologie s'est développée et fait son chemin dans la concurrence technologique. La figure 1.12 montre une abstraction de base de l'architecture de Cloud Computing [47].

Réaliser les différences entre les types d'architectures peut devenir confondu<sup>23</sup>. Il est fréquent de mélanger et de faire correspondre entre les différentes architectures. Il est délicat de choisir une architecture qui est plus appropriée, fiable et souhaitable pour les différentes

22. XML "Extensible Markup Language" est un langage utilisé pour le stockage et la transmission de données.

23. La confusion majeure est le mélange entre SOA et le Cloud Computing.

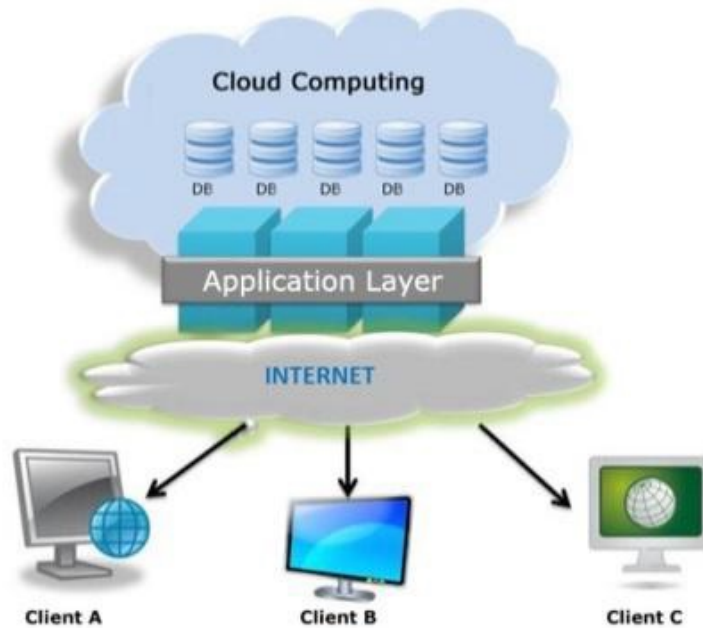


FIGURE 1.12 Architecture Cloud [47]

institutions. Le diagramme de Venn<sup>24</sup> de la figure 1.13 peut donner une vision plus claire des différentes architectures et de leurs limites.

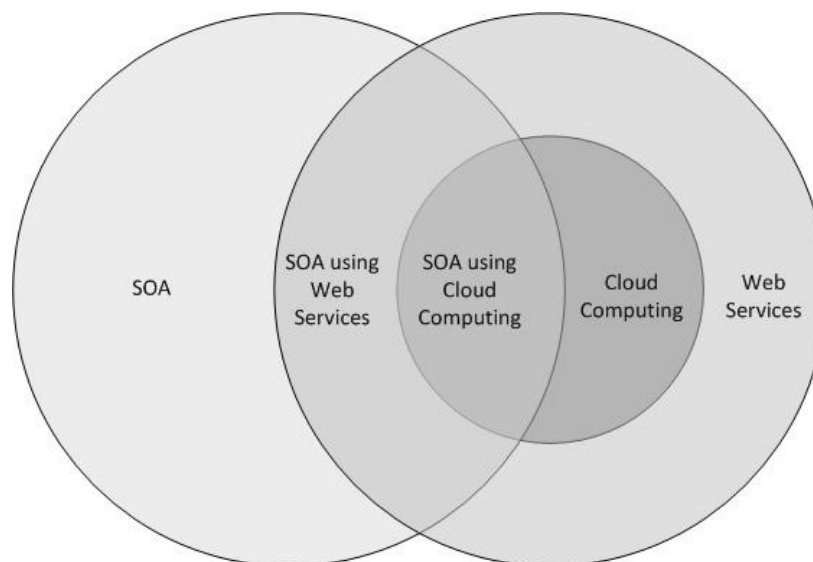


FIGURE 1.13 Les limites des différentes architectures ERP [47]

24. Un diagramme de Venn (également appelé diagramme logique) est un diagramme qui montre toutes les relations logiques possibles dans une collection finie de différents ensembles.

## 4 ERP et Cloud Computing

Dans le mode Cloud les logiciels sont installés et hébergés sur des serveurs distants dans le Cloud, plutôt que sur les terminaux ou les serveurs internes de l'organisation. Cette dernière ne paie pas une licence, mais un droit d'utilisation sous forme d'un abonnement mensuel ou annuel. Le prix de cet abonnement englobe tous les coûts du service (usage, stockage, sécurité et maintenance) et varie en fonction du nombre d'utilisateurs et des ressources consommées sur une période donnée. Plusieurs fournisseurs proposent des ERP en mode SaaS<sup>25</sup> tel que SAP, Oracle, Sage et Microsoft. Les principaux avantages de ces ERP sont :

- **Pas d'installation ni de mises à jour** : les logiciels SaaS sont pré-installés sur les serveurs dans le Cloud et prêts à l'emploi, ce qui diminue le temps de déploiement. L'entreprise n'a pas à se soucier des mises à jour qui sont réalisées par le prestataire.
- **Réduction de la consommation** : la consommation des périphériques est réduite grâce à la mutualisation des ressources entre les entreprises.
- **La sécurité des données** : la sécurité des données est à la charge du prestataire choisi, qui emploie des experts dont c'est le cœur de métier.
- **Accessibilité illimitée** : la délocalisation des logiciels sur les serveurs distants permet une utilisation nomade des services et documents. Il est possible d'y accéder n'importe où et n'importe quand. La seule condition est de disposer d'une connexion Internet.
- **Robustesse** : la mutualisation des ressources d'un Cloud public permet une grande robustesse et une haute disponibilité. Avec un taux de disponibilité élevé, il y a beaucoup moins d'interruptions de service pour les utilisateurs de l'ERP. La scalabilité est pratiquement sans limite, apportant ainsi des garanties d'évolutivité à l'entreprise.

On peut distinguer deux types de Cloud ERP : Le Cloud ERP hébergé et le Cloud ERP en mode SaaS.

### 4.1 Cloud ERP hébergé

Le Cloud ERP hébergé<sup>26</sup> est installé dans un centre de données ou un «centre d'hébergement» externe où des serveurs physiques ou virtualisés sont configurés. L'implémentation de la solution est semblable à celle de l'implémentation «sur site» ou dans vos bureaux.

Le paiement doit couvrir toutes les charges du fournisseur ERP tel que le paiement de l'implémentation du projet, l'approvisionnement initial du centre d'hébergement, le paiement

---

25. Les ERP en mode SaaS sont des ERP déployés dans le Cloud en tant qu'une seule entité de service. Dans notre recherche nous nous intéressons à composer plusieurs services Cloud pour obtenir un ERP.

26. "Le Cloud ERP hébergé" appelé en anglais *Hosted Cloud ERP*.

mensuel de l'utilisation de l'équipement du centre d'hébergement et la bande passante. Les frais continus à long terme incluraient les frais mensuels d'hébergement, les frais annuels de la maintenance de logiciels qui couvrent le débogage et les mises à jour, ainsi que toute assistance demandée.

### **Avantages**

- Le logiciel est possédé par l'entreprise cliente "une licence indéfinie". Le paiement est établi qu'une seule fois au-delà de la maintenance qui est généralement comprise entre 15% et 20% selon le fournisseur de L'ERP. Si la maintenance n'est pas payée, le logiciel continuera à fonctionner à la version actuelle.
- Les données sont stockées de manière fiable dans un centre sécurisé qui peut également offrir une redondance multi-site en cas d'exigence.
- La plupart des centres d'hébergement utilisent des serveurs «virtuels» tels que VMWare ou Microsoft Hyper-V qui permettent à synchroniser L'ERP hébergé "distant" avec le matériel physique local de l'entreprise en offrant une exécution plus rapide.

### **Inconvénient**

- Si la connexion Internet de l'entreprise tombe en panne, l'accès au système devient impossible (des lignes Internet redondantes vers l'entreprise peuvent aider à atténuer ce risque).
- L'intégration d'autres solutions au système ERP exige généralement la prise en charge et l'installation de ses solutions au centre d'hébergement, car il est difficile d'intégrer les applications locales avec les applications hébergées.
- La nécessité d'une large bande passante Internet. Surtout si l'application fonctionne avec des images numérisées et des photos dont leurs téléchargements peuvent saturer la bande passante.

## **4.2 Cloud ERP en mode SaaS**

Cloud ERP en mode SaaS<sup>27</sup> est un mode hybride d'un modèle financier et architectural. Comme la plupart des professionnels de l'industrie décrivent SaaS, l'application est concise et écrite à partir de zéro pour ce modèle. Les applications Cloud et hébergées peuvent être des applications Windows traditionnelles accessibles via une technologie telle que Citrix ou Terminal Services, ou une application basée sur un navigateur Web. Les applications SaaS

---

27. Même dans la littérature française, on utilise toujours l'acronyme anglais SaaS "Software as a Service" pour désigner "logiciel en tant que service".

sont vendues sur un modèle de location, généralement, X dollars par mois, par utilisateur. La plus grande différence avec les applications SaaS est qu'elles sont généralement multi-locataires. Cela signifie qu'une base de données partage plusieurs clients utilisateurs finaux et qu'ils sont partitionnés l'un par l'autre via un modèle de sécurité dans l'application, et non via des serveurs virtuels distincts.

### **Avantages**

- Les solutions SaaS peuvent être financièrement attrayantes. Il n'y a pas de coûts initiaux à utiliser. SAP Business By Design ou Epicor Express sont des exemples d'une véritable solution SaaS. Avec SAP, on peut remplir certaines informations de base et être dans un environnement de test en moins de 30 minutes.
- Les applications multi-locataires ont tendance à coûter moins cher dans la matière de l'entretien et l'exploitation et peuvent diminuer le coût mensuel pour les clients grâce à la base de données commune.
- Souvent, les services de la mise en œuvre et les mises à niveau futures sont regroupées dans les frais mensuels. Il peut être plus facile de dépenser 200 \$ par utilisateur par mois que 150 000 \$ pour une nouvelle solution ERP. Même si les 150 000 \$ sont effectivement moins chers sur 5 à 10 ans, les économies d'argent sont très attrayantes pour certaines entreprises.
- Dans le mode SaaS, il est probable que les applications ont été récemment développées, ce qui donne à beaucoup d'applications SaaS une apparence plus fraîche et une technologie plus moderne que les applications plus anciennes qui ont été déplacées vers le Cloud.
- L'intégration entre plusieurs applications SaaS (telles que ERP et CRM) peut être effectuée par les fournisseurs. Cela évite les entreprises de s'occuper des problèmes d'intégration et de configuration.

### **Inconvénients**

- Rien n'empêche un fournisseur SaaS d'augmenter les frais mensuels dans un an ou deux après avoir investi du temps et de l'argent dans la mise en œuvre du système. Avec SaaS, si vous ne payez pas, vous perdez l'accès à votre système, point final. Avec le service sur site, si vous arrêtez de payer la maintenance, votre logiciel continue de fonctionner au niveau de la version actuelle.
- Lorsque l'interface est uniquement Web, de nombreuses solutions sont plus lentes pour le transfert de données et manquent de la richesse d'une application Windows traditionnelle telle que l'analyse de clic droit, etc.

- Depuis que les fournisseurs SaaS surgissent rapidement et que beaucoup ne fonctionnent que grâce à Capital-Risque ou à d'autres fonds d'actions, leur survie à long terme est discutable à mesure que la consolidation se produit inévitablement. Si le fournisseur de logiciel SaaS tombe en panne, même s'il donne la possibilité d'obtenir les données, il pourrait prendre des mois pour ré-implémenter un nouveau système.

## 5 Sensibilité au Contexte

Une meilleure compréhension du contexte permettra aux concepteurs d'applications de choisir le contexte à utiliser dans leurs applications et de fournir des informations sur les types de données à prendre en charge et les abstractions et les mécanismes requis pour prendre en charge l'informatique contextuelle.

### 5.1 Définitions

Plusieurs définitions de contexte ont été décrites par différents auteurs. Quelques définitions ont simplement fourni des synonymes pour le contexte, se référant, par exemple, au contexte en tant qu'environnement ou situation. Bien que certains considèrent le contexte comme l'environnement de l'utilisateur, tandis que d'autres le considèrent comme l'environnement de l'application.

Par exemple, la définition de Schilit et *al.* se réfère au contexte en tant que lieu, aux identités des personnes et objets voisins, et aux changements de ces objets [56]. Ryan et *al.* définissent le contexte comme l'emplacement, l'environnement, l'identité et l'heure de l'utilisateur [57]. Adams et *al.* prétendent que les aspects importants du contexte sont : où vous êtes, avec qui vous êtes, et quelles ressources sont à proximité [58]. Ils définissent le contexte comme étant l'environnement d'exécution en constante évolution. Ils incluent les éléments suivants de l'environnement :

- Environnement informatique : processeurs disponibles, dispositifs accessibles pour l'entrée et l'affichage de l'utilisateur, la capacité du réseau, la connectivité et les coûts de l'informatique.
- Environnement de l'utilisateur : emplacement, collecte des personnes à proximité et situation sociale.
- Environnement physique : éclairage et niveau de bruit.

En se basant sur les différentes définitions dans la littérature, nous attribuons une définition laconique du contexte comme suit :

*Le contexte est toute information concernant les conditions et circonstances variables liées à l'utilisateur, l'application et l'environnement.*

## 5.2 Catégories du contexte

La sensibilité au contexte signifie la capacité de s'adapter aux nouvelles conditions. Cette adaptation ajuste le comportement du système en fonction de la situation (état du contexte). Cependant, ce qui est ajusté diffère d'un système à l'autre. Cette diversité se définit par la catégorie du contexte. Les principales catégories d'un contexte sont :

- Contexte physique : représente l'état du processus interne du système ; par exemple, réguler l'électro-consommation des appareils ménagers dans le but de maintenir la consommation globale du bâtiment dans certaines limites.
- Contexte utilisateur : est l'état et/ou la situation de l'utilisateur ; par exemple, traiter un patient sous surveillance distante d'une manière lorsque son état est normal et d'une autre manière en cas d'urgence.
- Contexte application : est le contexte du système qui a pour but d'offrir une sensibilité à la valeur lorsque le prestataire le demande ; par exemple, dans le cas du soutien des processus judiciaires, différents niveaux de transparence doivent être fournis aux différentes catégories de parties prenantes.

Cette catégorisation permet aux systèmes de réagir à un contexte donné avec une stratégie convenable.

## 5.3 Applications sensibles au contexte

La première définition des applications contextuelles donnée par Schilit et *al.* [58] a limité la définition des applications qui sont simplement informées du contexte aux applications qui s'adaptent au contexte (context-aware). De nombreux chercheurs définissent les applications contextuelles comme des applications qui modifient ou adaptent dynamiquement leur comportement en fonction du contexte de l'application et de l'utilisateur. Plus précisément, Ryan [59] les définit comme des applications qui surveillent les entrées provenant de capteurs environnementaux et permettent aux utilisateurs de choisir parmi une gamme de contextes physiques et logiques en fonction de leurs intérêts ou activités actuels. Brown et *al.* [60] définissent les applications sensibles au contexte (context-aware) en tant qu'applications qui fournissent automatiquement des informations et/ou prennent des mesures en fonction du contexte actuel de l'utilisateur détecté par les capteurs. Il prend également une vision étroite

de l'informatique contextuelle en affirmant que ces actions peuvent prendre la forme de présentation d'informations à l'utilisateur, d'exécution d'un programme en fonction du contexte ou de configuration d'une mise en page graphique en fonction du contexte. Enfin, Fickas et *al.* [61] définissent des applications orientées environnement comme des applications qui surveillent les changements dans l'environnement et adaptent leur fonctionnement selon des directives prédéfinies ou définies par l'utilisateur. En résumé, un système est sensible au contexte s'il utilise un contexte pour fournir des informations et/ou des services pertinents à l'utilisateur, lorsque la pertinence dépend de la tâche de l'utilisateur.

## 5.4 Contexte et ERP

Un ERP sensible au contexte est un ERP réagissant constamment aux différents changements du contexte. Le contexte d'un ERP peut être défini en plusieurs dimensions (utilisateur, application, environnement). Le contexte utilisateur (Entreprise) se présente comme besoins fonctionnels et non-fonctionnels évolutives de l'utilisateur à travers le temps. Le contexte de l'application se présente comme toute information concernant l'ERP lui-même, tel que ses modules constitutants, son type de déploiement et le mode de son utilisation. Le contexte de l'environnement contient les informations sur l'état du système, tel que l'emplacement de l'utilisateur et l'application, et le débit de la connexion entre l'utilisateur et l'ERP.

## 6 Conclusion

Ce premier chapitre est consacré à l'exposition du domaine de recherche de notre thèse avec ses différents axes. D'abord, nous avons présenté le Cloud Computing et son importance en décrivant ses caractéristiques, son modèle de services et ses modes de déploiement. Ensuite, nous avons illuminé l'utilité des ERP. En effet, nous avons décrit les objectifs, les caractéristiques, les architectures, les avantages et inconvénients des ERP. Nous avons également souligné les différentes manières d'utilisation des ERP dans le Cloud. Enfin, nous avons élaboré un aperçu sur le contexte et nous avons éclairci la notion des applications sensibles au contexte. Les trois axes abordés dans ce chapitre sont les principaux volets de notre recherche. Nous allons présenter dans les chapitres suivants d'autres axes attachés à notre travail de recherche telles que la composition des services Cloud et les ontologies.

Dans le chapitre suivant nous allons décrire l'état de l'art de la composition de services Cloud. Nous allons présenter les notions liées aux services Cloud et leurs coordinations. Nous allons également présenter les principaux travaux de recherche dans le domaine de la composition de services et les ERP dans le Cloud.

# Chapitre 2

## Composition de services Cloud

### Sommaire

---

|          |                                                                                          |           |
|----------|------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction . . . . .</b>                                                            | <b>40</b> |
| <b>2</b> | <b>Processus d'affaire . . . . .</b>                                                     | <b>40</b> |
| 2.1      | Définition . . . . .                                                                     | 41        |
| 2.2      | Exemple . . . . .                                                                        | 41        |
| 2.3      | Les exigences fonctionnelles et non-fonctionnelles . . . . .                             | 42        |
| <b>3</b> | <b>Composition de services Cloud . . . . .</b>                                           | <b>44</b> |
| 3.1      | Service Cloud . . . . .                                                                  | 44        |
| 3.2      | Les étapes de la composition des services Cloud . . . . .                                | 47        |
| 3.3      | Sélection des services Cloud . . . . .                                                   | 48        |
| 3.4      | Degré d'automatisation de la composition de services Cloud . . . . .                     | 51        |
| 3.5      | Approches de la composition des services Cloud . . . . .                                 | 52        |
| 3.6      | Aspects comportementaux des services composites . . . . .                                | 53        |
| 3.7      | Challenges de la composition des services Cloud . . . . .                                | 55        |
| <b>4</b> | <b>Principaux travaux dans le domaine de la composition des services Cloud . . . . .</b> | <b>56</b> |
| 4.1      | Les approches méta-heuristiques . . . . .                                                | 58        |
| 4.2      | Les ERP basés sur le WEB . . . . .                                                       | 61        |
| 4.3      | Les ERP à base de services . . . . .                                                     | 62        |
| 4.4      | Synthèse et Comparaison . . . . .                                                        | 62        |
| <b>5</b> | <b>Conclusion . . . . .</b>                                                              | <b>65</b> |

---

## 1 Introduction

Actuellement, l'automatisation de la gestion des entreprises est un facteur primordial pour le bon déroulement de leurs affaires et l'augmentation de leurs gains. Pour ce faire, ces entreprises adoptent des logiciels "voir des ERP" selon la complexité de leurs activités. D'autre part, l'évolution des activités de ces entreprises est un fait inévitable due à leurs défis concurrentiels. L'adaptation des logiciels en main aux évolutions éventuelles des activités de l'entreprise demande plus de temps et d'argent, ce qui peut diminuer les gains de l'entreprise. Par conséquent, le paradigme de composition des services Cloud est intervenu pour pallier ces soucis. En effet, il permet l'intégration rapide de nouvelles fonctionnalités et la réduction des coûts grâce à un modèle d'application hébergé [4]. En outre, plusieurs fournisseurs de services de Cloud proposent un large éventail de services hétérogènes avec différentes QoS (temps d'exécution, prix, disponibilité, fiabilité, sécurité, etc.). Il est dans l'intérêt des développeurs et utilisateurs de tirer parti de ces services pour créer un processus d'affaire qui répond aux besoins fonctionnels et non fonctionnels des entreprises.

Le processus de composition des services Cloud cherche à identifier, sélectionner, coopérer et coordonner plusieurs services Cloud disponibles afin de générer un service composite qui répond à une requête utilisateur spécifique. Ce processus peut produire tout une application en réalisant une composition hiérarchique des services Cloud existants. Plusieurs auteurs ont concentré leurs travaux de recherche dans cet axe pour améliorer ce processus et beaucoup d'entre eux se sont focalisés sur l'optimisation de la composition des services en matière de QoS. Les auteurs constatent que le processus de composition est une tâche hautement complexe, et les solutions proposées ne manquent pas d'inconvénients.

Les sections suivantes sont consacrées à la description du paradigme de composition des services Cloud. Nous définissons d'abord, le processus d'affaire qui représente le résultat de la composition des services Cloud. Ensuite nous présentons la définition du service Cloud, les étapes, les types et les approches de la composition des services Cloud. Enfin, nous étudions les principaux travaux de recherche du domaine de la composition des services présentés dans la littérature.

## 2 Processus d'affaire

Le processus d'affaire<sup>1</sup> est l'instrument clé pour organiser les activités d'une entreprise et d'améliorer la compréhension de leurs interrelations [62].

---

1. *Processus d'affaire* est appelé également "processus métier" ou "Processus d'entreprise". En anglais, on utilise l'expression "Business Process" pour désigner le *processus d'affaire*.

Les activités de processus d'affaire peuvent être effectuées par les employés de l'entreprise manuellement ou à l'aide de systèmes d'information. Ces activités peuvent être traitées automatiquement par les systèmes d'information sans aucune implication humaine.

Le processus d'affaire est un concept important pour faciliter la collaboration efficace entre les personnes physiques et les ressources de l'entreprise [62].

Dans de nombreuses entreprises, il existe une distance entre les aspects commerciaux organisationnels et la technologie de l'information mise en place. La réduction de cet écart est importante car dans les marchés dynamiques actuels, les entreprises sont constamment obligées de fournir de meilleurs produits à leurs clients. En effet, les produits qui ont du succès aujourd'hui pourraient ne pas l'avoir demain. Si un concurrent fournit un produit moins cher, mieux conçu ou plus facile à l'utilisation, la part du marché du premier produit sera probablement diminuée.

Au niveau organisationnel, les processus d'affaires sont essentiels à la compréhension du fonctionnement de l'entreprise, ils jouent également un rôle important dans la conception et la réalisation des systèmes d'informations flexibles. Ces systèmes d'information fournissent la base pour la création rapide de nouvelles fonctionnalités et pour adapter les fonctionnalités existantes pour répondre aux nouvelles exigences du marché.

La gestion des processus d'affaires est influencée par les concepts et les technologies de différents domaines de l'administration des affaires et de l'informatique. La nouvelle façon d'organiser les entreprises, basée sur les processus d'affaire, a été proposée dans les années 1990 [62].

## 2.1 Définition

Un processus d'affaire désigne un ensemble d'activités exécutées en coordination dans un environnement organisationnel et technique. Ces activités réalisent conjointement un objectif d'affaires. Chaque processus d'affaire est adopté par une seule organisation, mais il peut interagir avec les processus d'affaires adoptés par d'autres organisations. Un processus d'affaire peut être découpé en processus plus simples, jusqu'à en arriver aux activités élémentaires [63].

## 2.2 Exemple

La figure 2.1 montre un exemple de processus d'affaire "Commande produit". Ce processus a pour objectif de livrer et facturer au client le produit commandé en respectant les délais. Il faut noter qu'un modèle de processus d'affaire décrit en général le métier, et non le système informatique. Certaines actions décrites sont exécutées manuellement, sans interaction avec

un composant ou une application logicielle (par exemple, l'action "Livrer produit" se réalise sans utilisation d'un élément logiciel). Un processus d'affaire est transverse, il s'appuie en général sur plusieurs structures et applications d'une organisation, voire de plusieurs organisations. En UML, les processus d'affaires sont représentés à l'aide du diagramme d'activité. Cependant, les praticiens préfèrent souvent BPMN (Business Process Modeling Notation) [64] qui est jugé plus facile à appréhender.

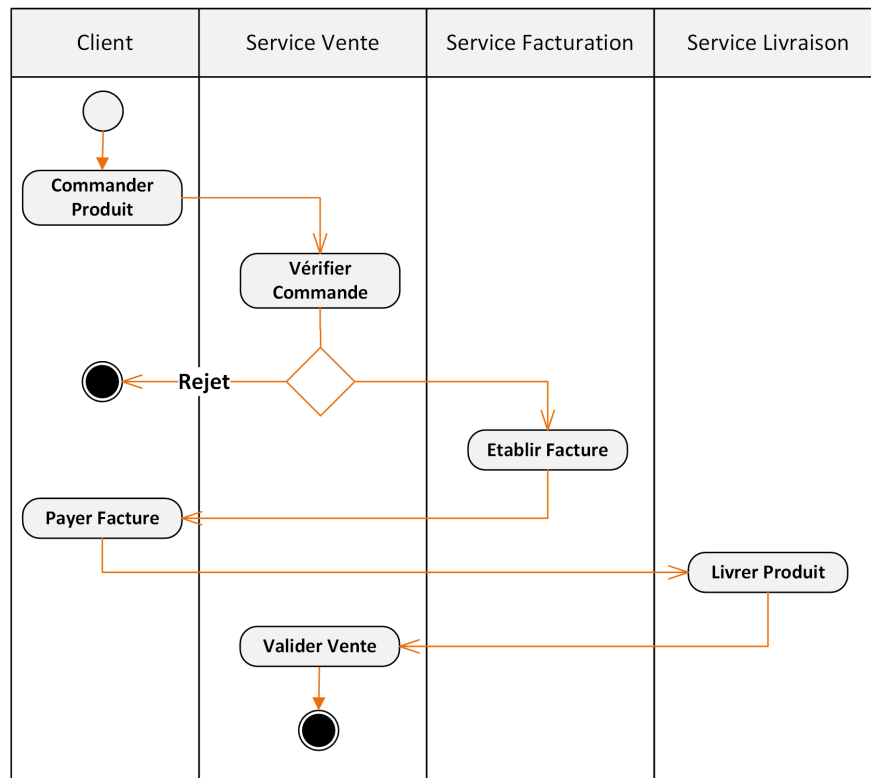


FIGURE 2.1 Processus d'affaire "Commande produit"

### 2.3 Les exigences fonctionnelles et non-fonctionnelles

Les exigences doivent être soigneusement sélectionnées afin de s'assurer qu'elles ont un sens dans le contexte du résultat final du projet et qu'elles sont transmises à tous les membres de l'équipe qui y travaillent. Passer à côté d'une exigence ou en faire une mauvaise application pourrait entraîner un désastre pour un projet.

Les exigences sont généralement de deux types : fonctionnelles et non-fonctionnelles. La différence entre eux est assez simple, néanmoins, nous allons définir les deux types d'exigences et fournir des exemples de chacun pour souligner plus concrètement la différence fondamentale entre eux.

- **Exigences fonctionnelles :** Une exigence fonctionnelle est *toute exigence qui spécifie ce que le système devrait faire*. En d'autres termes, une exigence fonctionnelle décrira un comportement particulier de la fonction du système lorsque certaines conditions sont remplies, par exemple : "Envoyer un e-mail lorsqu'un nouveau client s'inscrit" ou "Ouvrir un nouveau compte". Les exigences fonctionnelles typiques incluent [65] :
  - Règles d'affaires
  - Corrections de transactions, ajustements et annulations
  - Fonctions administratives
  - Authentification
  - Niveaux d'autorisation
  - Suivi d'audit
  - Interfaces externes
  - Conditions de certification
  - Conditions de rapport
  - Données historiques
  - Exigences légales ou réglementaires
- **Exigences non-fonctionnelles :** Une exigence non-fonctionnelle est *toute exigence qui spécifie comment le système remplit une certaine fonction*. En d'autres termes, une exigence non fonctionnelle décrira comment un système doit se comporter et quelles sont les limites de sa fonctionnalité. Les exigences non fonctionnelles spécifient généralement les attributs ou les caractéristiques de qualité du système, par exemple : «Les données modifiées dans une base de données doivent être mises à jour pour tous les utilisateurs qui y accèdent dans un délai de 2 secondes». Les exigences non-fonctionnelles les plus connues sont [65] :
  - Performance (temps de réponse, débit, etc.)
  - Scalabilité
  - Capacité
  - Disponibilité
  - Fiabilité
  - Maintenabilité
  - Sécurité
  - Utilisabilité
  - Interopérabilité

### 3 Composition de services Cloud

La composition des services Cloud est un processus complexe adoptée dernièrement afin de réutiliser et combiner les services Cloud atomiques pour obtenir un service composite répondant à une requête particulière. Les services atomiques sont déployés dans les différents serveurs Cloud, et ils sont publiés dans des annuaires spécialisés (voir figure 2.2). Avant la coordination des services atomiques, le processus de composition passe par l'étape de "découverte et sélection de services" qui se charge d'identifier et de sélectionner les services atomiques appropriés à partir des différents fournisseurs Cloud.

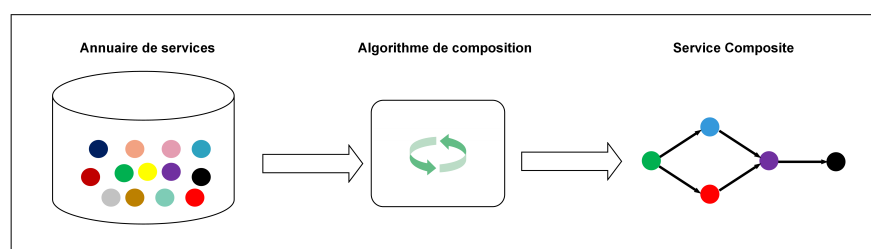


FIGURE 2.2 Composition de services

#### 3.1 Service Cloud

Un service Cloud est un protocole d'interface informatique de la famille des technologies Web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit d'un ensemble de fonctionnalités exposées par des fournisseurs Cloud. Le service Cloud peut être décrit, publié, localisé, et agrégé dynamiquement avec d'autres services afin de construire des systèmes distribués, interopérables et évolutifs [66].

Les services Cloud sont publiés dans un annuaire avec toutes leurs informations nécessaires à leurs utilisations. Cet annuaire est un registre mettant à la disposition de l'utilisateur un ensemble d'informations spécifiant les propriétés des services Cloud accessibles à travers l'Internet. Il indique principalement le fonctionnement du service, les informations de l'accessibilité pour assurer un accès structuré entre le client du service et son fournisseur.

En plus des propriétés fonctionnelles, un service Cloud possède d'autres types de propriétés, telles que les qualités de service (QoS) qui font de ce composant logiciel le service le plus approprié à un client donné qu'à un autre. Une multitude de services Cloud sont fournis par différents fournisseurs et qui proposent des fonctionnalités similaires mais répondent différemment aux besoins et préférences spécifiques des clients.

### 3.1.1 Propriétés des services Cloud

Les propriétés des services Cloud sont divisées souvent en deux classes : *propriétés fonctionnelles* et *non-fonctionnelles*. D'autres classes ont été séparées de la classe non-fonctionnelle telles que les propriétés contextuelles, les propriétés orientées données, les propriétés techniques et les propriétés comportementales.

- **Propriétés fonctionnelles**

Une propriété fonctionnelle est une propriété relative à la fonctionnalité du service qu'elle décrit [67]. Elle constitue la spécification de l'offre du service Cloud lorsqu'il est invoqué par ses clients. Cette spécification comporte en général la catégorie métier du service Cloud, les opérations, les entrées/sorties ou encore une pré-condition à vérifier au préalable avant son exécution. Dans le cas d'un service Cloud de paiement électronique par exemple, " payer en ligne par carte bancaire ", " vérifier la validité du numéro de la carte bancaire " et " paiement électronique " sont trois propriétés fonctionnelles qui renseignent respectivement sur l'opération qu'il exécute, la pré-condition à vérifier au préalable avant son exécution et sa catégorie métier.

- **Propriétés non-fonctionnelles**

Les propriétés non-fonctionnelles décrivent l'aspect non-fonctionnel du service Cloud. Ces propriétés sont souvent désignées par des contraintes, des attributs ou des paramètres du service Cloud qui décrivent les informations non-fonctionnelles du service Cloud. Elles constituent en fait des paramètres qui régissent son comportement ou son utilisation telles que les QoS (la fiabilité, la performance, la sécurité, le prix, etc.) et d'autres propriétés primordiales pour l'utilisation correcte du service Cloud tel que le protocole de communication avec le service Cloud et le type des entrées/sorties [68]. Dans le cas d'un service Cloud de paiement électronique, la description des propriétés non-fonctionnelles liées à la sécurité est primordiale. La signature numérique, le type de cryptage et le niveau de confiance sont par exemple trois propriétés non-fonctionnelles qui peuvent être indiquées pour renseigner sur le niveau de sécurité garanti par le service Cloud fourni. Ainsi, le protocole de communication (p.ex. SOAP, REST, etc.) et le schéma de données<sup>2</sup> sont des propriétés non-fonctionnelles du service Cloud.

---

2. La connaissance de la structure de données est indispensable pour faire le mapping entre le client et le service Cloud.

### 3.1.2 Approches principales de description des services Cloud

Les propriétés des services Cloud peuvent être décrites à deux niveaux : syntaxique et/ou sémantique. Les premières spécifications de services Web [69] sont décrites seulement à un niveau syntaxique et elles ont été basées que sur leur aspect fonctionnel. Cependant, le besoin d'automatisation des processus de découverte, de sélection et de composition de services suite à leur augmentation exponentielle sur le Web, a très rapidement révélé les limitations de ces travaux. En effet, la plupart des services publiés sur Internet se trouvent décrits par des fichiers de type WSDL (Web Services Description Language). Néanmoins, ce langage qui constitue un standard W3C (World Wide Web Consortium) s'est avéré incapable de fournir une description suffisamment étoffée pour les consommateurs des services Cloud. Il se contente de décrire au niveau syntaxique des aspects élémentaires des services Cloud. Ainsi, les standards W3C, SAWSDL (Semantic Annotations for WSDL and XML Schema) et WS-Policy, ont été proposés dans la littérature pour couvrir cette limitation. Ces approches introduisent d'autres propriétés de services Cloud et prennent également en considération le niveau de description sémantique en enrichissant les fichiers WSDL. D'autres travaux et modèles sémantiques [70, 29, 71–74] ont essayé de pallier ces limites en introduisant l'aspect non-fonctionnel et en proposant des descriptions de niveau sémantique. En se basant sur les bases et les caractéristiques des approches qui restent les plus utilisées dans ce cadre tels que les standards WSDL, SAWSDL et WSPolicy, ainsi que OWL-S (OWL Web Ontology Language for Services) [29] et WSMO (Web Service Modeling Ontology) [70], on peut distinguer deux classes. La première classe est basée sur l'utilisation de langages syntaxiques à base de XML pour la description des services Cloud et la deuxième classe décrit ces services par des modèles sémantiques.

1. **Approches à base de langages syntaxiques :** La classe des approches à base de langages syntaxiques utilise des structures syntaxiques de type XML pour décrire les propriétés des services Cloud. Elle est constituée du langage WSDL ainsi que des extensions de ce langage tels que SAWSDL et les WS\*-spécifications (WS-Policy). Ils ont été proposés par la communauté de chercheurs du consortium W3C pour munir le langage ancêtre WSDL d'une couverture plus élevée des différents aspects de description de services Web tout en prenant compte de leur sémantique au moyen du langage SAWSDL. Ainsi, le fait qu'ils soient standardisés, leur structure basée sur le langage XML en fait des langages aisément extensibles pour s'adapter à de nouveaux et éventuels besoins en matière de description des services Cloud.
2. **Approches à base de modèles sémantiques :** La classe des approches basées sur l'usage des modèles sémantiques adopte une démarche sémantique pour décrire les ser-

vices Cloud. Afin de fournir une description sémantique de ces services interprétables automatiquement, elle exploite la notion d'ontologie qui a été adoptée par plusieurs chercheurs en Web sémantique en tant qu'approche prometteuse pour ce type d'applications. Cette classe comprend les deux frameworks OWL-S (voir figure 2.3) et WSMO qui représentent les modèles sémantiques de description des services Web les plus référencés. Ces frameworks proposent une description sémantique des services couvrant plusieurs aspects, mais ils ne se limitent pas uniquement à la description de ces services. Ils proposent aussi des mécanismes de découverte et de composition automatique des services Web.

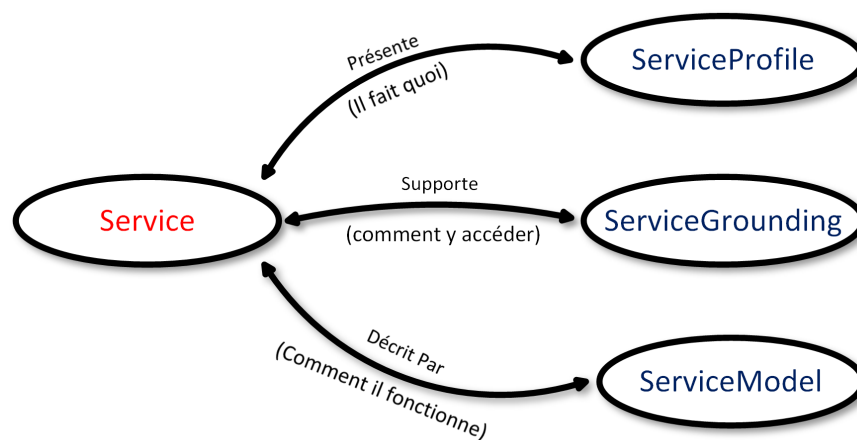


FIGURE 2.3 OWL-S : Semantic Markup for Web Services

Dans le chapitre 3 nous allons présenter une nouvelle ontologie pour gérer un grand nombre de services Cloud hétérogènes regroupés sémantiquement en fonction de leur catégorie de service et leur description fonctionnelle et contextuelle. Cette ontologie facilite le processus de découverte de services Cloud pertinents, l'interopérabilité des services et met en évidence les contraintes sémantiques des utilisateurs.

### 3.2 Les étapes de la composition des services Cloud

Le mécanisme de composition de services Cloud doit suivre les étapes suivantes de la spécification à l'exécution afin d'obtenir un service composite adéquat répondant au besoin du client.

1. **la découverte et la sélection des services** : dans cette étape, le mécanisme de composition permet d'identifier et localiser les services compatibles à partir des annuaires disponibles. Ensuite, il se charge de la sélection des services qui répondent adéquate-

ment aux besoins du client. Cette étape est primordiale pour la composition, dont nous allons la détailler après.

2. **la médiation entre services** : l'étape précédente engendre un ensemble de services qui peuvent répondre à la requête du client. Cette étape n'est pas souvent suffisante pour l'assemblage des services engendrés. En conséquence, une autre étape qui est la médiation doit être intervenue pour établir les interactions entre les services. Cette étape prend en considération les concepts sémantiques des services et le contexte du système afin de répondre précisément aux requêtes du client.
3. **le déploiement et l'invocation des services** : cette étape s'occupe du déploiement des services sur les plateformes du Cloud pour permettre aux clients de les invoquer et les exécuter. Le déploiement est généralement à la charge des fournisseurs de services Cloud et non pas aux invocateurs de services.

La composition des services Cloud est un processus qui sert à offrir au client un service composite qui répond à ses besoins. Dans la majorité des cas, le processus de composition engendre plusieurs solutions qui répondent à la requête du client. En effet, le mécanisme doit choisir la solution optimale parmi toutes les solutions possibles. L'optimisation de la composition de services est basée sur les besoins fonctionnels et non-fonctionnels (QoS) du client.

### 3.3 Sélection des services Cloud

La découverte des services Cloud est la tâche d'identification des services Cloud qui répondent à une requête particulière. C'est une phase primordiale dans le processus de composition des services Cloud. Elle consiste à découvrir les services Cloud appropriés pour les coopérer afin de former un service composite qui répond à une requête complexe. Thoma et *al.* [75] la définissent comme étant le processus qui prend en entrée une requête d'utilisateur et renvoie une liste de ressources ou de services qui peuvent éventuellement remplir le besoin décrit. Dans ce qui suit, on va détailler les trois étapes de la découverte des services Cloud.

#### 3.3.1 Spécification de la requête utilisateur

La première chose que doit faire un usager de service Cloud est la spécification de la requête selon ses besoins et exigences sans avoir besoin de connaître les spécifications des services Cloud. Plusieurs approches de représentation des requêtes de services Cloud ont été présentées dans la littérature telles que les approches textuelles [76, 77], les approches

basées sur la logique [78, 79], les approches graphique [80] et les approches sémantiques [81, 82].

- **Les approches textuelles** utilisent le langage naturel pour spécifier une requête, ce qui nécessite un traitement supplémentaire qui consiste à formaliser et interpréter la requête avant de pouvoir la traiter automatiquement.
- **Les approches basées sur la logique** spécifient les requêtes formellement sous forme de prédicats afin de les traiter en décomposant les requêtes complexes en sous-requêtes simples.
- **Les approches graphiques** décrivent les requêtes sous forme de nœuds et d'arcs. Les entrées du service composite sont représentées par les nœuds de départ tandis que les nœuds d'arrivées représentent les sorties du service composite.
- **Les approches fondées sur la sémantique** utilisent l'ontologie et les règles sémantiques pour exprimer une requête de l'utilisateur.

Généralement, la requête utilisateur est spécifiée par ses exigences fonctionnelles et non-fonctionnelles. L'opération de découverte est effectuée en appariant les propriétés fonctionnelles des services publiés avec la description fonctionnelle de la requête. Les aspects non-fonctionnels sont utilisés dans l'étape de sélection des services découverts.

### 3.3.2 Localisation des services Cloud

La localisation des services Cloud satisfaisant une requête donnée est une phase essentielle dans l'étape de découverte. On distingue deux catégories d'approches de localisation :

- Les approches basées sur la recherche à partir des registres centralisés à travers des interfaces de description des services Cloud stockés dans un annuaire<sup>3</sup> central tel que l'UDDI [83] ou WSIndex [84]. Dans cette catégorie, c'est les fournisseurs des services qui sont chargés volontairement d'enregistrer et publier leurs services dans les annuaires.
- Les approches basées sur la recherche à partir des référentiels distribués "décentralisés", supposant souvent que la description du service Cloud est présentée dans le site du fournisseur de ce service. Dans cette catégorie, la requête utilisateur est exécutée généralement par un middleware d'agents ou des robots d'extraction des descriptions des services Cloud [85–87]. Cependant, la gestion des informations dans un environnement distribué est une tâche complexe à cause de la description hétérogène des services Cloud par les différents fournisseurs.

---

3. On utilise généralement l'acronyme "Repository" pour désigner les dépôts et les annuaires.

### 3.3.3 Appariement des requêtes et des services Cloud

Après la localisation des services Cloud, une opération est effectuée sur les services localisés qui s'appelle l'appariement "Matching". Cette opération a pour objectif de faire la correspondance entre la requête utilisateur et la description des services localisés. Elle consiste à identifier les similarités et les rapprochements entre les besoins décrits de la requête et les propriétés de la description du service Cloud ciblé. L'appariement est appliqué en priorité sur les propriétés fonctionnelles pour bien répondre aux besoins fonctionnels de l'utilisateur indépendamment des contraintes et préférences non-fonctionnelles ou des QoS. Les modes d'appariement les plus utilisés sont classés en deux catégories principales : l'appariement syntaxique et l'appariement sémantique dont les principes sont expliqués ci-dessous.

- **Appariement syntaxique**

Les techniques d'appariement syntaxique des services Cloud consiste à déterminer la correspondance syntaxique entre la spécification de la requête utilisateur et la description syntaxique du service Cloud. Dans ce mode, la spécification et la description des propriétés sont exprimées syntaxiquement, et l'identification des similarités se base sur une analyse lexicale et syntaxique. Parmi les mécanismes d'appariement syntaxique des services proposés dans la littérature, on cite l'appariement de mots-clés [88] et l'appariement d'interfaces WSDL [89]. Le premier mécanisme consiste à faire la comparaison entre les mots clé de la spécification de la requête utilisateur avec ceux de la description du service Cloud enregistrée dans l'annuaire des services. Le deuxième effectue la comparaison entre les balises du fichier WSDL décrites dans le service abstrait qui présente la requête utilisateur avec ceux des services publiés.

- **Appariement sémantique**

Contrairement aux techniques syntaxiques, les mécanismes sémantiques effectuent la correspondance entre les propriétés fonctionnelles décrites de la requête utilisateur avec celles des services Cloud en se basant sur leur sens<sup>4</sup> pour la découverte des services Cloud. On va décrire deux catégories de méthodes d'appariement sémantiques qui sont : les méthodes à base d'analyse sémantique latente (LSA : Latent Semantic Analysis) [90] et les méthodes basées sur le calcul de similitudes sémantiques [91].

— Les méthodes basées sur l'analyse sémantique latente analysent les documents WSDL des services Cloud publiés afin d'extraire la description sémantique de leurs fonctionnalités. Elles utilisent généralement des robots de collection et

---

4. On utilise le terme sémantique pour désigner le sens des données.

d'extraction d'information<sup>5</sup> pour localiser, dans le Cloud, les fichiers WSDL appropriés [92]. L'analyse sémantique latente consiste principalement à comparer les mots utilisés pour les éléments de description (opérations, paramètres, etc.) dans les fichiers WSDL avec ceux utilisés pour la spécification de la requête utilisateur en se référant à des dictionnaires pour analyser les mots et les expressions ayant le même sens.

- Les méthodes de calcul de similitudes sémantiques s'articulent sur le calcul des distances sémantiques entre les propriétés fonctionnelles (entrées, sorties, opérations, etc.) de la requête utilisateur et des services Cloud publiés afin d'évaluer la correspondance. Ces méthodes utilisent généralement les ontologies qui permettent de fournir des descriptions sémantiques exprimant le sens des termes utilisés dans un domaine et leurs liens sémantiques [93]. Le calcul de la distance sémantique entre deux nœuds ontologiques est basé sur la profondeur des nœuds dans la structure hiérarchique de l'ontologie et le nombre des nœuds qui les séparent.

D'autres approches d'appariement sémantique utilisent des ontologies optant le contexte [94, 95] qui permettent d'intégrer des informations concernant le contexte de l'utilisateur et de spécifier les contraintes de contexte au niveau de la requête dans le processus d'appariement.

### 3.4 Degré d'automatisation de la composition de services Cloud

Le degré d'automatisation doit être pris en considération pour déterminer la nature du processus de composition des services Cloud. Selon le critère d'automatisation, la composition des services Cloud peut être classée en trois catégories : manuelle, semi-automatique et automatique [96].

#### 3.4.1 Composition manuelle

La composition manuelle est la méthode traditionnelle où l'utilisateur des services doit programmer manuellement toutes les étapes du processus de composition afin d'avoir le service composite dont il a besoin sans utiliser les outils dédiés existants. L'inconvénient de cette méthode est qu'elle exige des compétences et des efforts considérables de la part de l'utilisateur qui n'est pas toujours un développeur. Non seulement, cette tâche est visiblement difficile, elle est aussi sensible aux erreurs dues à la nature dynamique et évolutive des services Cloud. En effet, les changements et l'évolution des services Cloud atomiques influence le

---

5. On appelle la fonctionnalité de ces robots : "Information Retrieval"

service composite utilisé et peut le mener à la défaillance. Ce qui oblige l'utilisateur à mettre à jour la composition du service chaque fois un service atomique change de caractéristiques.

### 3.4.2 Composition automatique

La composition automatique est un mécanisme qui réalise tout le processus de composition automatiquement, sans aucune intervention de l'utilisateur. Le processus de composition automatique est un processus générique fourni aux différents utilisateurs pour la recherche et la composition des services appropriés et convenables à leurs besoins spécifiques. Ce processus est générique et flexible, ce qui permet la prise en considération des contraintes et des préférences personnalisées de chaque client. Cependant, l'automatisation du processus de composition des services augmente l'échelle des solutions possibles et complique énormément l'interprétation des services Cloud existants. En effet, ce type de composition est une tâche d'une grande complexité à cause de la diversité des propriétés des services Cloud à explorer pour découvrir et sélectionner ceux qui sont appropriés et compatibles avec le schéma de composition. Cette complexité peut encore augmenter en raison de l'accroissement rapide des services disponibles dans le Cloud et de la difficulté d'interprétation sémantique correcte des propriétés de ces services pour répondre aux besoins du client.

### 3.4.3 Composition semi-automatique

Les inconvénients du processus de composition automatique des services Cloud mènent certains spécialistes dans ce domaine à considérer que ce type de composition comme trop ambitieux et irréaliste. Par conséquent, ils proposent un troisième type qui est la composition semi-automatique ou interactive. Dans ce genre de composition, l'utilisateur dispose d'un certain contrôle et peut gérer le processus de composition considéré, même s'il n'a pas de connaissances approfondies en programmation. En effet, ils existent des outils graphiques qui sert d'assistance à l'utilisateur pour concevoir, modéliser et implémenter des services Cloud composites, sélectionner les services Cloud appropriés selon la base de règles sémantiques proposée, spécifier ses préférences et intervenir à chaque étape du processus de composition.

## 3.5 Approches de la composition des services Cloud

Pour répondre à un besoin complexe du client, les services Cloud disponibles peuvent être combinés et coordonnés afin d'obtenir un service composite. Cette coordination peut être réalisée de façon statique ou dynamique selon l'environnement du procédé.

### 3.5.1 Composition statique

Dans l'approche statique, la composition des services Cloud est effectuée à la phase de la conception du service composite. Les services Cloud sont préalablement identifiés, sélectionnés et combinés entre eux pour être déployés irrémissiblement. Ce genre de composition est plus convenable aux environnements stables où les services identifiés au préalable par l'utilisateur ne sont pas sujets à changement à court terme. Par conséquent, la modification ou la substitution d'un service composant atomique exige des modifications au niveau de la conception du service composite.

### 3.5.2 Composition dynamique

Dans l'approche dynamique, l'opération de composition des services Cloud est déclenchée par la requête de l'utilisateur. Cette opération permet de découvrir, sélectionner et assembler dynamiquement les services à partir des annuaires de services Cloud, au moment de l'exécution de la requête en respectant les contraintes de l'utilisateur. Le point fort de l'approche dynamique par rapport à l'approche statique est la flexibilité et l'adaptation aux changeants de l'utilisateur. En effet, cette approche permet d'engendrer des schémas de compositions adaptables à chaque requête à partir des services disponibles au cours de la composition. Cependant, la réalisation de la composition dynamique dans un environnement où le nombre de services fournis évolue en permanence, est une tâche considérablement complexe.

## 3.6 Aspects comportementaux des services composites

Il existe deux aspects comportementaux qui servent à décrire la façon de la constitution d'un service composite. Le premier aspect appelé chorégraphie décrit le comportement interne et le deuxième appelé l'orchestration décrit le comportement externe. L'information sur le comportement du service composite est primordiale pour l'interaction correcte entre ses services atomiques qui le composent.

### 3.6.1 Orchestration

L'orchestration représente le comportement externe d'un service composite, c'est-à-dire la manière de l'invocation et l'agrégation des services atomiques afin d'offrir un service composite qui répond à la requête complexe de l'utilisateur [97]. En effet, le moteur d'orchestration permet de définir l'enchaînement des services selon un schéma déterminé, ensuite il s'occupe

de l'invocation des services afin d'être exécutés dans leurs serveurs Cloud propriétaires (voir figure 2.4).

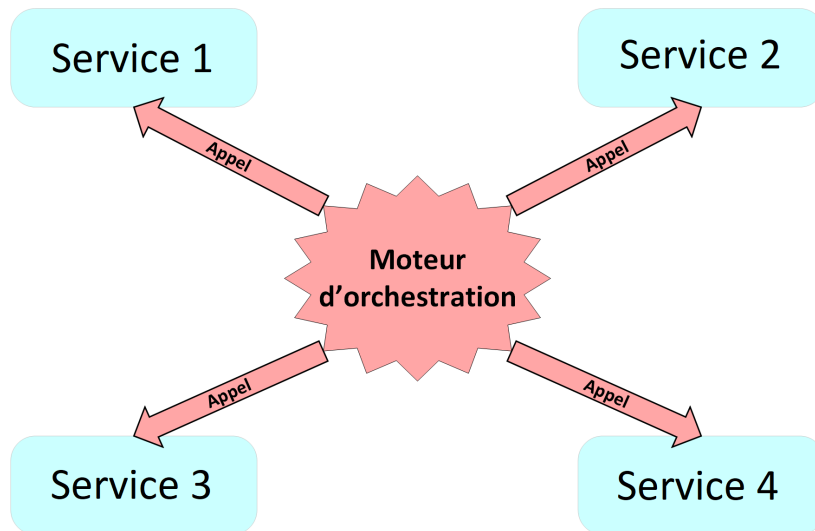


FIGURE 2.4 L'orchestration des services Cloud

### 3.6.2 Chorégraphie

La chorégraphie est l'aspect interne du service Cloud, elle décrit les différentes règles d'interactions exigées au client pour bénéficier des fonctionnalités du service Cloud. En effet, c'est un moyen d'atteindre un objectif commun en utilisant un ensemble de services Cloud en tant que composants par l'intermédiaire des flux de contrôle [98]. Elle consiste à concevoir une coordination décentralisée des services Cloud, qui communiquent entre eux à l'aide d'échange de messages (voir figure 2.5). Cette approche soulève de nombreux problèmes, notamment pour le traitement des erreurs et l'évolution dynamique des services.

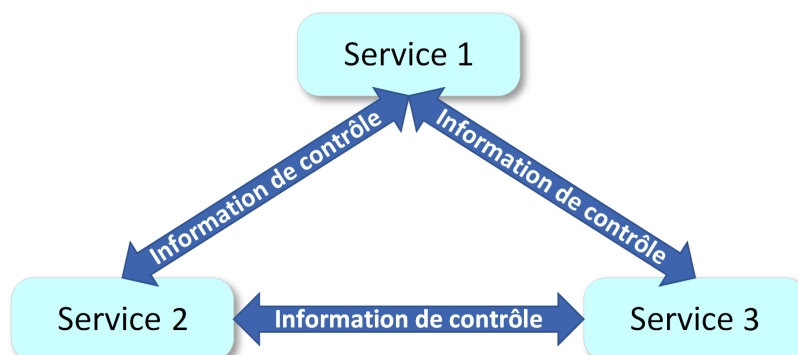


FIGURE 2.5 La chorégraphie des services Cloud

Dans la section suivante, nous allons décrire les principaux défis posés dans le domaine de la composition des services Cloud.

### **3.7 Challenges de la composition des services Cloud**

#### **3.7.1 Respect des besoins fonctionnels de l'utilisateur**

Pour satisfaire les besoins fonctionnels d'un utilisateur, une correspondance doit être effectuée entre la spécification de la requête utilisateur et les spécifications fonctionnelles des services Cloud disponibles. Cette tâche passe par les étapes de découverte et sélection qui permettent d'obtenir les services les plus appropriés à la requête utilisateur. Néanmoins, plusieurs difficultés s'imposent par chaque étape de composition tels que l'hétérogénéité des services Cloud disponibles avec leurs différentes représentations syntaxiques et les changements continus de ces services. En outre, plusieurs combinaisons possibles des services Cloud sélectionnés peuvent satisfaire la requête utilisateur, ce qui mène le processus de composition à effectuer la combinaison optimale. L'utilisation des concepts sémantiques aident considérablement à traiter le problème de la représentation syntaxique des différents services hétérogènes et leur composition. Cependant, le problème reste toujours complexe et pousse les chercheurs à améliorer la correspondance et l'adaptation aux changements établis sur les services Cloud.

#### **3.7.2 Optimisation des critères de QoS**

L'optimisation de la composition des services Cloud cherche à coordonner plusieurs services Cloud pour obtenir un service composite qui répond à une requête utilisateur en respectant ses besoins fonctionnels et non-fonctionnels (QoS). La sélection des services pour la composition optimale en termes de QoS est une tâche complexe. Étant donné que plusieurs QoS contradictoires doivent être optimisés simultanément (par ex, un minimum de prix et de temps d'exécution), les algorithmes d'optimisation visent à trouver le service composite optimal avec le meilleur compromis de QoS (trade-off).

#### **3.7.3 Augmentation de la flexibilité : "Sensibilité au contexte"**

Les utilisateurs des services Cloud cherchent à utiliser le minimum de services répondants à leurs besoins. Ces besoins ne sont pas constants et peuvent être changé à n'importe quel moment selon le contexte imposé. Par conséquent, la solution fournie à l'utilisateur doit être assez flexible pour réagir convenablement aux changements du contexte. La composition des services en fonction du contexte est un aspect central pour y parvenir dont les services

sont découverts et composés en fonction du contexte pour répondre aux besoins des clients. Les environnements contextuels sont intrinsèquement dynamiques et hétérogènes, ce qui rend la composition des services en fonction du contexte une tâche complexe. Les défis de la recherche dans le domaine de la composition des services sont nombreux et certains ont déjà été réalisés. Nous allons présenter dans les chapitres suivants l'approche que nous avons proposée pour la conception des applications ERP sensibles au contexte et la façon de les modéliser.

### 3.7.4 Dynamicité

Dans un environnement dynamique, le processus de l'optimisation de la composition de services Cloud est effectué à la volé au moment de l'exécution de chaque requête utilisateur en respectant ses contraintes de QoS. Les changements de QoS en raison de la dynamicité de l'environnement Cloud et l'accès concurrent aux services impliquent des dégradations qualitatives au niveau de certains QoS, notamment, la disponibilité et la fiabilité. Le processus de composition dynamique doit répondre aux attentes des utilisateurs en réagissant à tous les changements attribués dans le contexte de l'utilisateur, le contexte Cloud et le contexte de l'environnement.

## 4 Principaux travaux dans le domaine de la composition des services Cloud

Plusieurs travaux ont été proposés pour la sélection et la composition des services. Ils sont généralement regroupés en trois catégories principales : les approches basées sur les contraintes [10, 11], les approches basées sur la classification sémantique des services [12–14] et les approches méta-heuristiques [15–28]. Avec les approches basées sur les contraintes, Rosenberg et *al.* [11] ont proposé une approche semi-automatique appelée CaaS (Composition as a Service) combinée avec un langage de domaine spécifique VCL (Vienna Composition Language). Ce langage détermine les contraintes hiérarchiques et les paramètres de QoS pour le processus de composition des services Cloud. Ce langage est assez riche, mais il n'utilise pas les descriptions sémantiques de services Cloud hétérogènes et il ne prend pas en considération le critère de violation des contraintes qui permet une sélection efficace et une satisfaction aux clients du service. Cela implique une incompatibilité due à l'hétérogénéité et augmente ainsi la complexité du problème de décision. Une approche qui se concentre sur le problème de la composition dynamique des services sous contraintes de temps et QoS dans le Cloud Computing mobile est proposée par Deng et *al.* [10]. Ils visaient à construire

un service composite qui ne satisfait pas seulement les contraintes de temps et de qualité de service, mais traite également la composition des services dans un environnement mobile et incertain. Ils ont proposé une évolution différentielle pour l'algorithme de composition de services dirigée par les contraintes (DE4CDSC)<sup>6</sup>. Ils ont également utilisé un processus de filtrage pour réduire l'espace de recherche, ensuite ils ont adopté un algorithme différentiel évolutionnaire pour former un service combiné en maximisant sa probabilité de succès après l'examen de la mobilité des fournisseurs de services Cloud.

Jun Li et *al.* [99] ont proposé un algorithme heuristique pour la sélection efficace et fiable de l'ensemble de services. L'algorithme proposé comprend trois étapes. D'abord, une sélection basée sur la méthode de confiance est utilisée pour filtrer les services de composants non fiables. La deuxième étape consiste à construire les services associés pour réduire l'espace de recherche dans le processus de composition des services. A la dernière étape, une approche heuristique pour l'optimisation globale est utilisée pour obtenir une solution proche de la solution optimale. Cela réduit considérablement l'espace de recherche formé principalement par des services fiables, et améliore ainsi le temps d'exécution du processus de la composition des services.

Alti et *al.* [14] ont proposé une approche de composition sémantique automatique basée sur l'utilisation d'ontologies. Ils ont décrit une ontologie qui permet la génération automatique pour l'assemblage des services hétérogènes de qualité. Ils ont également effectué une composition dynamique de services basée sur une description sémantique (nom, catégorie, rôle), caractéristiques (coût / qualité) des services d'adaptation et les paramètres de contexte (préférences du client, dispositif et environnement). De plus, cette approche permet d'établir des relations d'équivalence entre les services. Ce travail est exploité automatiquement et sémantiquement pour générer le processus global de composition des services virtuels. La solution de qualité et le temps d'exécution de cette approche dépend fortement de la conception de la fonction de coût<sup>7</sup>, ce qui nécessite une certaine mesure. Cela dépend aussi de la conception des opérateurs de sélection efficaces pour offrir des services plus précis et plus pertinents. Cependant, ce travail ne tient pas compte de la croissance d'une variété de fournisseurs de services hétérogènes, alors que les clients ont besoin de services Cloud efficaces et de haute qualité.

Les approches basées sur les contraintes et les approches sémantiques génèrent un service composite réalisable qui répond à certaines contraintes et préférences, mais ne fournit pas le service composite dont les QoS soient optimales<sup>8</sup>.

---

6. DE4CDSC : est l'abréviation de l'expression "Differential Evolutionary for Constraint Driven Service Composition Algorithm".

7. La fonction de coût est appelée généralement "fitness function".

8. Ce problème est défini dans la littérature par : "Optimal QoS-aware Cloud service composition".

## 4.1 Les approches méta-heuristiques

Le problème de la composition optimale des services Cloud compatible QoS est connu en tant que NP-difficile [100] dont on ne peut pas résoudre ce problème avec des méthodes exactes dans un temps polynomial. Divers travaux utilisant des techniques méta-heuristiques ont été proposés pour résoudre ce problème. Les méta-heuristiques sont des méthodes stochastiques servant à trouver une solution proche de l'optimale. On distingue deux approches d'optimisation :

### 4.1.1 Les approches fondées sur la scalarisation

Un problème multi-objectif peut être résolu en le réduisant à un problème mono-objectif en utilisant la technique de scalarisation. Le but de cette technique est d'associer une valeur de fitness (coût, score ou utilité) à chaque solution (c'est-à-dire un service composite) selon certaines fonctions d'utilité pour sélectionner la meilleure solution parmi les solutions possibles. La technique de scalarisation la plus utilisée est la somme pondérée (SAW)<sup>9</sup>.

Il existe plusieurs travaux qui utilisent les méthodes méta-heuristiques basées sur la scalarisation. Par exemple, Pop et *al.* [28] ont présenté une méthode basée sur la recherche tabou pour sélectionner la solution optimale ou quasi-optimale dans la composition d'un service Cloud sémantique. La méthode proposée est appliquée à une structure graphique de planification améliorée qui code toutes les solutions de composition qui satisfont une requête utilisateur. Les critères de sélection de la solution optimale incluent les attributs QoS et la similarité sémantique entre les services impliqués dans la composition. La méthode proposée a été évaluée sur des scénarios du domaine de planification de voyage.

Sasikaladevi et *al.* [18] ont utilisé une méthodologie systématique pour sélectionner les services requis basés sur leurs caractéristiques non-fonctionnelles ou sur la qualité de service (QoS). Ils ont développé un algorithme génétique pour maximiser le caractère non-fonctionnel "fiabilité"<sup>10</sup> du service Cloud.

Ye et *al.* [20] ont proposé un modèle extensible basé sur les critères QoS pour calculer les valeurs de QoS des services Cloud. Ils ont également attribué un algorithme génétique pour la composition des services Cloud. L'approche proposée fournit des solutions optimales pour des scénarios à petite échelle. Cependant, cette approche manque de stratégie pour gérer efficacement un grand nombre de services Cloud et de fournir des services composites appropriés qui respectent à la fois la QoS et les contraintes contextuelles.

---

9. SAW est l'acronyme de "Simple Additive Weighting".

10. Le terme "fiabilité" désigne en anglais la QoS "Reliability".

Karimi et *al.* [19] ont utilisé un algorithme génétique pour réaliser une optimisation globale en ce qui concerne l'accord de niveau de service (SLA). De plus, le regroupement de services a été utilisé pour réduire l'espace de recherche du problème et des règles d'association ont été utilisées pour un service composite en fonction de leur historique afin d'améliorer l'efficacité de la composition des services. Les expériences menées ont reconnu l'efficacité supérieure de la méthode proposée par rapport à des travaux connexes similaires.

Long et *al.* [24] ont proposé un algorithme d'optimisation par l'essaimage de particules (PSO : Particule Swarm Optimization) sensible à l'environnement pour améliorer la vitesse de convergence. Cet algorithme diminue l'instabilité résultante de la recherche aléatoire et augmente la vitesse de l'algorithme par la méthode de rappel de la population optimisée et l'augmentation du champ visuel.

Jula et *al.* [25] ont élaboré un nouvel algorithme hybride basé sur l'attraction gravitationnelle (en tant que recherche locale) et l'algorithme de la compétition impérialiste (ICA)<sup>11</sup> pour l'optimisation du temps de réponse et les frais d'exécution simultanément. L'élégibilité de QoS des solutions générées ont été calculées par une équation basée sur la normalisation du temps de réponse et des frais d'exécution.

Yu et *al.* [27] ont présenté un algorithme glouton appelé Greedy-WSC et un algorithme basé sur l'optimisation par les colonies de fourmis appelé ACO-WSC<sup>12</sup>, qui tente de sélectionner des combinaisons de Cloud qui sont réalisables en utilisant un nombre minimum de Cloud. Leurs résultats expérimentaux montrent que la méthode d'optimisation de colonies de fourmis proposée peut trouver efficacement des combinaisons de services avec un nombre minimal de Clouds. L'algorithme n'a pas besoin d'explorer tous les services Clouds pour avoir une solution réalisable ce qui mène à réduire le temps d'exploration et de composition des services Cloud. Néanmoins, il ne fournit pas une solution optimale car il n'interroge pas tous les fournisseurs du Cloud qui peuvent avoir des services attribuant d'autres meilleures compositions.

Le principal inconvénient commun des méthodes de scalarisation est que la solution optimale est basée sur la fonction fitness qui exprime les QoS avec une valeur globale et unique. Donc, l'optimisation de la fonction fitness peut pénaliser quelques QoS à travers d'autres. Pour bien expliquer l'inconvénient mentionné, on va présenter un exemple dans ce qui suit.

Supposons par exemple qu'on cherche à optimiser la fonction fitness ci-dessous :

---

11. ICA : Imperialist Competitive Algorithm.

12. ACO-WSC : Ant Colony Optimization - Web Service Composition.

$Fitness = 0.5 * Q_1 + 0.5 * Q_2$  tel que  $Q_1$  et  $Q_2$  correspondent aux QoSs *temps de réponse* et *sécurité* respectivement.

Prenant maintenant les deux solutions  $S_1$  et  $S_2$  suivantes :

- $Fitness(S_1) = 0.5$  avec  $Q_1 = 0.6$  et  $Q_2 = 0.4$ .
- $Fitness(S_2) = 0.53$  avec  $Q_1 = 0.95$  et  $Q_2 = 0.1$ .

La valeur de la fonction fitness de  $S_2$  (0.53) est supérieure à celle de  $S_1$  (0.5). Cependant,  $Q_1$  de la solution  $S_2$  égale à 0.1, dont on peut juger que c'est une valeur inacceptable en tant que valeur de sécurité et on préfère la solution  $S_2$  qui à une fonction de fitness inférieur à celle de  $S_1$ , mais avec une valeur de  $Q_2$  acceptable (0.4). On déduit que la solution avec une valeur de fitness optimale n'est pas la solution idéale pour certains QoS et peut être même une solution inacceptable.

#### 4.1.2 Approches basées sur le Pareto (approches multi-objectif)

Les approches d'optimisation multi-objectif sont souvent utilisées lorsque deux objectifs conflictuels ou plus doivent être considérés simultanément afin de négocier<sup>13</sup> un ensemble de solutions Pareto-optimales. Nous citons dans ce cadre quelques algorithmes d'optimisation multi-objectif tels que NSGA-II [16], SPEA2 [21], E3-MOGA [22] et MOPSO [101].

Wada et al. [22] ont proposé un framework d'optimisation basé sur un algorithme génétique multi-objectif appelé E3. Il résout de manière heuristique le problème de composition de services Web compatible-QoS dans un délai raisonnable. Cet algorithme produit un ensemble Pareto de solutions de qualité équivalente pour satisfaire plusieurs SLA<sup>14</sup>.

Li et al. [21] ont proposé l'utilisation d'algorithmes évolutifs multi-objectif (MOEA : Multi-Objective Evolutionary Algorithm) pour l'optimisation de la composition de services en fonction de QoS. L'algorithme est bon pour traiter les problèmes d'optimisation combinatoire multi-objectif. Les résultats expérimentaux révèlent que SPEA2 est capable d'approcher le front de Pareto-optimal avec une distribution bien étalée.

Rezaie et al. [101] ont présenté un modèle de composition de service Web basé sur l'optimisation d'essaimage de particules multi-objectif (MOPSO : Multi-Objective Particle Swarm Optimization), en se bénéficiant des principaux avantages de la PSO standard, à savoir la simplicité et la rapidité.

Deb et al. [102] ont proposé un algorithme multi-objectif appelé (NSGA-II : Non-dominated Sorting Genetic Algorithm) avec une complexité d'ordre  $O(MN^2)$ . Cet algorithme est basé sur le tri des solutions non-dominées et la distance d'encombrement<sup>15</sup>. Un opérateur

13. On utilise le terme "trade-off" pour désigner la négociation entre les différents objectifs à optimiser.

14. SLA est l'acronyme de l'expression "Service Level Agreement".

15. La distance d'encombrement est désignée dans la littérature anglaise par "Crowding distance".

de sélection est utilisé pour créer un pool d'accouplement en combinant les populations de parents avec celles des descendants en choisissant les meilleures solutions. Le critère "distance d'encombrement" est utilisé pour obtenir plus de variété de solutions au niveau du front du Pareto. Yao et *al.* [16] ont proposé l'approche NSGA-II pour la composition de services Web compatible-QoS.

Les approches Pareto-optimales génèrent un ensemble de solutions, mais éventuellement les clients n'ont besoin que d'une seule solution qui devrait être sélectionnée automatiquement parmi l'ensemble des solutions résultantes.

Dans notre travail, nous avons adopté un approche Pareto-optimal (NSGA-II) auquel nous avons ajouté une nouvelle relation de dominance Pareto basée sur une fonction de pénalité mesurant le degré de violation de contraintes pour obtenir un ensemble de processus d'affaire ERP qui respectent les contraintes du client. Ensuite nous avons intégré les préoccupations liées aux attributs de QoS par la pondération additive simple (SAW<sup>16</sup>) pour déterminer le processus d'affaire optimal qui répond aux préférences contextuelles du client.

## 4.2 Les ERP basés sur le WEB

Actuellement, le principal avantage concurrentiel de chaque entreprise réside dans sa capacité de prescrire, normaliser et adapter ses activités et sa collaboration avec les clients, les fournisseurs, les partenaires et les concurrents. Récemment, la plupart des fournisseurs ERP reconnaissent l'importance du problème de l'interopérabilité et ont développé des modules de chaîne / logistique sur Internet pour faciliter l'intégration avec des systèmes back-end<sup>17</sup> sur un ensemble de bases de données, d'infrastructures informatiques et d'applications [103].

Gollakota [104] a rapporté une entreprise qui crée des kiosques par un ordinateur et un accès internet, et exploite un portail web répondant aux demandes de l'industrie agricole. Le portail fournit des informations relatives à l'agriculture générale, culture, vétérinaire et à accès au système ERP de l'entreprise.

Tarantilis et *al.* [4] ont développé un système ERP basé sur le Web pour traiter les problèmes de l'entreprise et gérer les processus d'affaires réels, allant d'une simple procédure de bureautique à une planification complexe de la chaîne d'approvisionnement.

Zhang et *al.* [105] ont développé l'exploration de l'innovation des services informatiques dans les grappes industrielles textiles au point de vue du système de service. Ils ont soutenu que le service des producteurs informatisés pourrait être utilisé pour assurer la mise à niveau structurelle des grappes industrielles textiles.

---

16. SAW : Simple Additive Weighting.

17. Back-end appelé arrière-plan est un terme désignant une couche d'une application devant produire un résultat. A l'opposé le front-end est la partie visible de l'application.

Mital et *al.* [106] ont développé un framework d'intégration pour identifier les caractéristiques du choix du SaaS dans le contexte spécifique de l'approvisionnement en ligne de l'ERP en mode SaaS.

### 4.3 Les ERP à base de services

Dans cette section nous allons citer deux principaux travaux qui ont entamé la composition de services pour obtenir un ERP. Li et *al.* [3] ont proposé une approche basée sur SOA et la technologie des composants pour l'obtention d'un ERP end-to-end. L'approche proposée pourrait aider à la réalisation d'une intégration sémantique dynamique axée sur les objectifs commerciaux des services Web. Il permet l'adaptation aux demandes en constante évolution du marché, l'accélération du processus d'achèvement des systèmes d'information et peut ensuite aider les entreprises à réduire leurs coûts et améliorer la capacité de la concurrence.

Chen et *al.* [17] ont défini une plate-forme CloudERP sur laquelle l'entreprise cliente pouvait personnaliser un système ERP spécifique correspondant à ses besoins particuliers. Le CloudERP fournit aux entreprises clientes un service ERP complet provenant de plusieurs fournisseurs. Ils ont également élaboré la composition des services Web et proposé une solution Web pour l'automatisation du processus de personnalisation du service ERP. La méthode de composition de service proposée est basée sur l'algorithme génétique incorporant la connaissance des services Web extraits de la plate-forme de service Web avec la théorie des ensembles bruts<sup>18</sup>. Cette approche a le même objectif que notre travail dans le principe de fournir un ERP complet aux clients grâce à la composition des services Cloud. Cependant, il n'utilise pas les concepts sémantiques et il n'est pas sensible aux changements du contexte.

### 4.4 Synthèse et Comparaison

La composition des services Cloud est un processus adopté pour bénéficier de la réutilisation des modules existants et éviter le développement de nouveaux modules qui coûtent considérablement en matière de temps et d'argent. Les travaux de recherches existants dans ce cadre se focalisent spécialement sur l'optimisation de la composition de services en termes de QoS. Les différentes approches présentées dans ce domaine sont diverses et chacun d'entre eux a ses caractéristiques et répond à certains critères spécifiques.

Les travaux basés sur les contraintes [10, 11] utilisent un processus de filtrage selon des contraintes pour réduire l'espace d'exploration de services Cloud. Cependant, l'appariement est basé sur la syntaxe et n'utilise pas l'aspect sémantique. Ce qui peut empêcher un ensemble

---

18. La théorie des ensembles bruts est connue sous le nom de "Rough set theory".

de services Cloud avec une certaine similarité d'entrer en concurrence lors du processus de découverte.

Les travaux basés sur la sémantique [12–14] sont plus fiables lors de la découverte et la sélection car l'appariement se fait en basant sur le sens des caractéristiques des services Cloud et non sur les termes syntaxiques, ce qui peut étendre l'ensemble de services Cloud concurrents.

Néanmoins, les approches basées sur les contraintes et les approches sémantiques cherchent à composer des services Cloud pour obtenir un service composite qui répond à une requête utilisateur sans prendre en considération l'optimisation de la composition en termes de QoS.

Les approches méta-heuristiques se focalisent sur l'optimisation des QoS du service composite engendré. Ces approches utilisent deux méthodes d'optimisation : les méthodes de scalarisation et les méthodes multi-objectif.

Les méthodes de scalarisation [17–20, 24, 25, 27, 28] transforment le problème d'optimisation multi-objectif en un problème mono-objectif en utilisant des techniques de scalarisation, tel que la méthode SAW, pour attribuer une fonction fitness à chaque service composite. La solution générée par ces méthodes est un service composite ayant la meilleure valeur fitness. Les méthodes de scalarisation génèrent une solution dont la fonction fitness est optimale, mais peut dans beaucoup de cas pénaliser quelques QoS.

Les méthodes multi-objectif [16, 21, 22, 101] génèrent un ensemble de solutions (Pareto-optimal) qui représente les solutions ayant un meilleur compromis de QoS. Cependant, ces méthodes obligent les utilisateurs de sélectionner manuellement une solution parmi l'ensemble de solutions Pareto-optimal engendrées.

Le tableau 2.1 montre une étude comparative entre les différentes approches en basant sur certains critères d'évaluation.

TABLE 2.1 Comparaison des approches de composition de services

| Approches                |       | Automatique | Dynamique/Statique | Contexte | QoS | Sémantique | Violation de contraintes |
|--------------------------|-------|-------------|--------------------|----------|-----|------------|--------------------------|
| Basé sur les contraintes | [10]  | Oui         | Dynamique          | Oui      | Oui | Non        | Non                      |
|                          | [11]  | Semi-auto   | statique           | Non      | Oui | Non        | Non                      |
|                          |       |             |                    |          |     |            |                          |
| Sémantiques              | [12]  | Oui         | Statique           | Non      | Non | Oui        | Non                      |
|                          | [13]  | Oui         | Statique           | Non      | Oui | Oui        | Oui                      |
|                          | [14]  | Oui         | Dynamique          | Oui      | Oui | Oui        | Non                      |
| Scalarisation            | [17]  | Oui         | Statique           | Non      | Oui | Oui        | Non                      |
|                          | [18]  | Oui         | Dynamique          | Non      | Oui | Non        | Non                      |
|                          | [19]  | Oui         | Dynamique          | Non      | Oui | Oui        | Non                      |
|                          | [20]  | Oui         | Statique           | Non      | Oui | Non        | Non                      |
|                          | [24]  | Oui         | Statique           | Non      | Oui | Non        | Non                      |
|                          | [25]  | Oui         | Statique           | Non      | Oui | Non        | Non                      |
|                          | [27]  | Oui         | Dynamique          | Non      | Oui | Non        | Non                      |
|                          | [28]  | Oui         | Statique           | Non      | Oui | Oui        | Non                      |
| Multi-objectif           | [16]  | Semi-auto   | Dynamique          | Non      | Oui | Non        | Non                      |
|                          | [21]  | Semi-auto   | Statique           | Non      | Oui | Non        | Non                      |
|                          | [22]  | Semi-auto   | Statique           | Non      | Oui | Non        | Non                      |
|                          | [101] | Semi-auto   | Statique           | Non      | Oui | Non        | Non                      |

Méta-heuristiques

## 5 Conclusion

Le présent chapitre est consacré à l'état de l'art de la composition des services Cloud en passant par la définition des éléments essentiels dans ce domaine. Nous avons constaté que l'objectif de la composition est d'engendrer un service composite (qui peut être un processus d'affaire) à partir des services Cloud publiés dans des annuaires et proposés par les différents fournisseurs Cloud. Contrairement aux approches traditionnelles de développement des applications qui sont obligé de développer tous les modules de l'application, les approches de composition de services essaient d'utiliser les services existants disponibles dans le Cloud et de les faire coordonner afin d'obtenir l'application souhaitée. Nous avons notamment présenté les principaux travaux dans l'optimisation de la composition des services Cloud en les classifiant dans trois catégories (les approches basées sur les contraintes, les approches sémantiques et les approches méta-heuristiques). En effet, nous avons détaillé quelques exemples de travaux de chaque catégorie et nous avons montré leurs caractéristiques principales.

Après avoir présenté l'état de l'art de notre domaine de recherche, nous allons présenter dans le chapitre suivant une nouvelle ontologie qui permet d'utiliser l'aspect sémantique dans le processus de la composition des services Cloud afin d'assurer son efficacité. Cette ontologie est la colonne vertébrale de nos propositions et contributions.

**Deuxième partie**

**Contributions**

# Chapitre 3

## Modèle de description des applications ERP à base d'ontologie

### Sommaire

---

|          |                                                                |           |
|----------|----------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction . . . . .</b>                                  | <b>68</b> |
| <b>2</b> | <b>Ontologie proposée . . . . .</b>                            | <b>69</b> |
| <b>3</b> | <b>Modèle sémantique du service . . . . .</b>                  | <b>70</b> |
| 3.1      | Service virtuel . . . . .                                      | 71        |
| 3.2      | Service concret . . . . .                                      | 71        |
| <b>4</b> | <b>Modèle sémantique de QoS à deux dimensions . . . . .</b>    | <b>73</b> |
| 4.1      | Modèle de QoS-Utilité . . . . .                                | 73        |
| 4.2      | Modèle de QoS-LongueVie . . . . .                              | 74        |
| 4.3      | Modèle de QoS . . . . .                                        | 75        |
| <b>5</b> | <b>Modèle sémantique du contexte . . . . .</b>                 | <b>76</b> |
| 5.1      | Le contexte du service Cloud . . . . .                         | 76        |
| 5.2      | Le contexte du Client . . . . .                                | 77        |
| 5.3      | Le contexte de l'environnement . . . . .                       | 80        |
| <b>6</b> | <b>Modèle sémantique du processus d'affaire . . . . .</b>      | <b>80</b> |
| 6.1      | Processus d'affaire virtuel . . . . .                          | 81        |
| 6.2      | Processus d'affaire concret . . . . .                          | 81        |
| <b>7</b> | <b>L'interrogation et l'inférence . . . . .</b>                | <b>82</b> |
| <b>8</b> | <b>Exemple de PA selon l'ontologie CxQSCloudSERP . . . . .</b> | <b>84</b> |
| <b>9</b> | <b>Conclusion . . . . .</b>                                    | <b>87</b> |

---

## 1 Introduction

Les services Cloud distribués par les différents fournisseurs de Cloud augmentent de plus en plus en matière d'hétérogénéité (les types de données, la variété des plateformes et les différents mécanismes de communication). De plus, comme les besoins des clients sont exprimés par différents langages et différentes expressions selon leurs contextes, des ontologies contextuelles basées sur la sémantique sont nécessaires pour mieux répondre à ces besoins.

D'autre part, le processus de gestion de qualité des systèmes ERP dépendant des besoins spécifiques des clients et leurs préférences est une tâche complexe. Cette complexité est clairement visible lorsqu'un service est fourni par deux ou plusieurs fournisseurs Cloud. Une décision devrait être prise pour sélectionner et composer les services Cloud les plus appropriés en fonction des besoins contextuels avec la meilleure qualité de service. L'ontologie a acquis une signification plus large dans l'interopérabilité sémantique entre les domaines d'informations hétérogènes. Elle est particulièrement adaptée aux systèmes ERP en prenant un rôle important pour fournir des informations sémantiques de la qualité et du contexte, essentielles à la description des processus d'affaires. Dans ce cadre, l'ontologie peut fournir un modèle conceptuel partagé pour améliorer la sémantique des processus d'affaires et permettre au client de localiser le processus d'affaire à travers son sens indépendamment du contexte du service évolué et du fournisseur de Cloud.

L'ontologie des services Web sémantique "OWL-S" présentée dans le chapitre précédent offre des capacités appropriées, notamment, la découverte, la sélection et la composition automatique des services. Cependant, l'ontologie OWL-S ne traite pas la manière de prendre en compte les besoins non-fonctionnels (les contraintes et les préférences de QoS) et comment les représenter sémantiquement. En outre, elle ne prend pas vraiment en considération les informations contextuelles (le contexte des services Cloud, le contexte du client et l'environnement) dans la sélection et la composition des services Cloud pour ERP.

La qualité sémantique et la description contextuelle du processus d'affaire sont deux aspects importants pour gérer un grand nombre de services Cloud hétérogènes, contrôler la qualité des services Cloud au niveau sémantique et adapter le système ERP aux besoins personnalisés des clients.

Dans le présent chapitre, nous décrivons une nouvelle ontologie différente des ontologies proposées dans la littérature. En effet, nous proposons un système intelligent pour la gestion efficace des processus d'affaires basé sur la sémantique des services Cloud pour obtenir un ERP sensible au contexte. Le système facilite les annotations sur les processus d'affaires, les qualités des services Cloud et les besoins non-fonctionnels des clients en attribuant des métadonnées sémantiques aux propriétés de la qualité et du contexte. Dans ce système, un

mécanisme de raisonnement est appliqué sur les services Cloud annotés sémantiquement pour générer automatiquement des processus d'affaires virtuels, découvrir des services Cloud pertinents et répondre aux besoins particuliers des différents clients.

## 2 Ontologie proposée

L'objectif principal de notre ontologie est (1) de faire correspondre les besoins des clients avec la spécification des services virtuels au niveau sémantique, (2) d'identifier le concept de processus d'affaire virtuel en tant que combinaison de services virtuels (tâches) pour les différentes applications ERP [107] et (3) d'envisager les informations de contexte et les paramètres de QoS. En outre, cette ontologie accélère le processus d'identification des services d'affaires pertinents du Cloud et améliore l'efficacité et la pertinence du processus de développement d'ERP. La classification sémantique est utile pour une prise de décision intelligente afin d'optimiser le processus de composition des services Cloud selon les besoins des clients. Ces besoins diffèrent considérablement par la qualité et le contexte en termes de contraintes et de préférences. Le but de la classification des services Cloud est d'améliorer l'efficacité du processus de composition en les hiérarchisant selon un ou plusieurs critères tels que la fonctionnalité, le rôle du service, la QoS, la qualité de sortie et les paramètres de contexte de service [108]. Si le processus d'affaire sémantique est correctement divisé en services d'affaires appropriés, le processus d'affaire sémantique peut être spécifié et analysé plus facilement. Au niveau sémantique, le processus d'affaire peut être mesuré avec des critères, appelé  $QoS_U$  (Quality of Service Utility) et  $QoS_L$  (Quality of Service Long-Life). Le client cherche une qualité de service basée sur sa signification, indépendamment de la langue utilisée, du service contextuel évolué et du fournisseur de services Cloud. L'ontologie adoptée vise à mieux capturer la sensibilité du contexte et à contrôler la qualité des services au niveau sémantique. Le processus d'affaire virtuel est déduit à partir des besoins fonctionnels du client. Le processus d'affaire concret est composé de services Cloud et il est optimisé selon les contraintes et les préférences de QoS du Client. La Figure 3.1 illustre la structure de notre ontologie baptisée "CxQCloudSERP". Elle contient cinq concepts clés : Processus d'affaire, le Contexte, le Service virtuel, le Service Cloud<sup>1</sup> et les QoS ( $QoS_U$ ,  $QoS_L$ ). Elle est utilisée pour :

- ✓ La représentation sémantique des informations du service (rôle du service, catégorie de service, les ressources requises, les paramètres de qualité et la sémantique des entrées/sorties).

---

1. A l'opposé du service virtuel, le service Cloud désigne le service concret hébergé dans le Cloud.

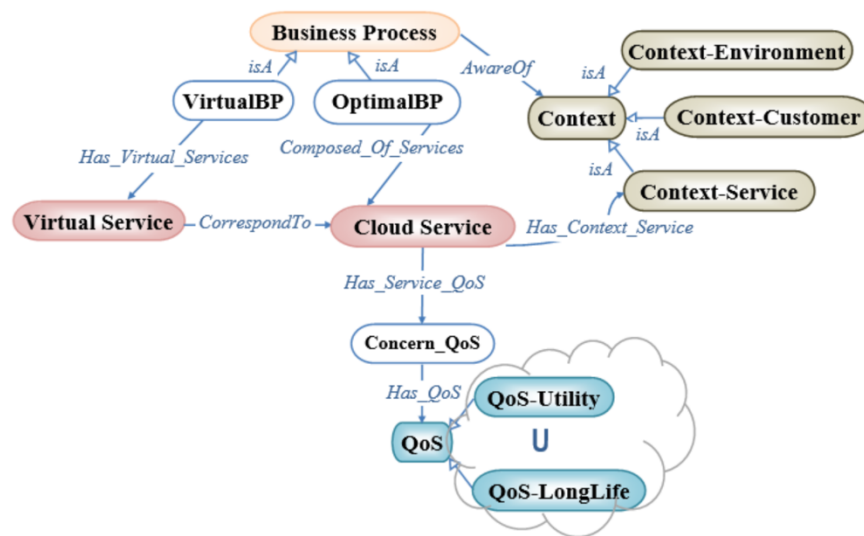


FIGURE 3.1 Un aperçu général sur l'ontologie CxQSCloudSERP

- ✓ La description sémantique de tout type de service d'affaire de n'importe quel fournisseur de services Cloud en basant sur la compatibilité des QoS.
- ✓ L'unification des propriétés et des contraintes des qualités de service pour faciliter leur gestion et leur partage.
- ✓ Classification des services Cloud selon leur catégorie, leur QoS et leur contexte.
- ✓ L'attribution d'une correspondance sémantique entre les valeurs quantitatives de QoS et les termes qualitatifs. Par exemple : le temps de réponse "moins rapide" ou "lent" correspond à la valeur quantitative de QoS appartenant à l'intervalle [0.1, 0.3].
- ✓ La déduction automatique des processus d'affaires virtuels qui répondent aux besoins fonctionnels des clients grâce à des règles d'inférence basées sur les informations sémantiques de service (catégorie, rôle, QoS, etc.).
- ✓ La déduction automatique de la qualité de service correspondant aux spécifications sémantiques des préférences et contraintes du client.
- ✓ La séparation des besoins fonctionnels et non-fonctionnels pour contrôler et maintenir la qualité du processus d'affaire.

### 3 Modèle sémantique du service

Dans notre modèle ontologique, nous considérons deux types de services : le service virtuel et le service concret.

### 3.1 Service virtuel

Un service virtuel est un service abstrait associé à un ensemble de services Cloud ayant les mêmes fonctionnalités. Chaque service virtuel est défini par un identifiant unique, un nom, des entrées/sorties, des opérations, un rôle et une catégorie [108] (voir figure 3.2).

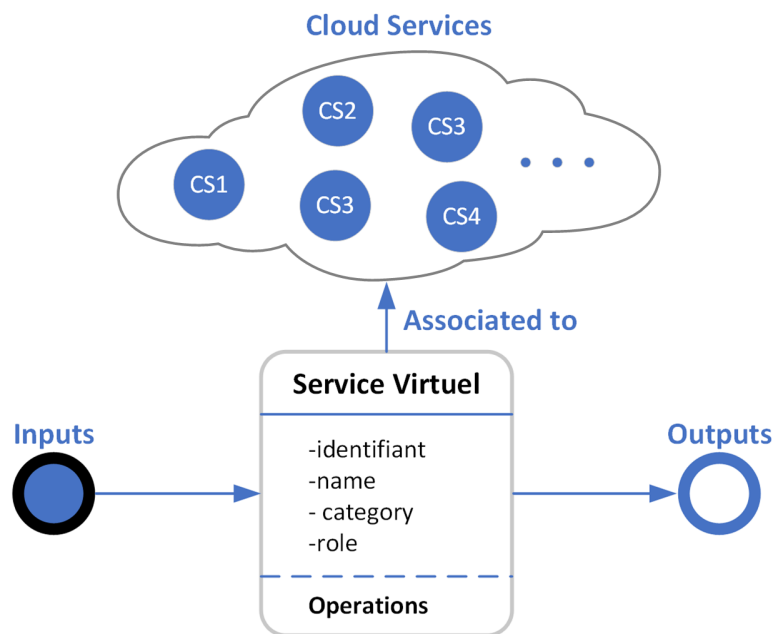


FIGURE 3.2 Représentation d'un service virtuel

### 3.2 Service concret

Un service concret (ou service Cloud) est une instance concrète du service virtuel (voir figure 3.3). Il est caractérisé par un identifiant unique, un nom, une version, une URI<sup>2</sup>, ses QoS, son contexte et son modèle du service (SaaS, PaaS et IaaS)<sup>3</sup>. Chaque service Cloud est fourni par un fournisseur Cloud (Cloud provider). Chaque fournisseur est caractérisé par son emplacement<sup>4</sup> et son mode de déploiement (privé, public et hybride) [107, 108].

L'ontologie proposée sélectionne un service dans un environnement dynamique en confrontant un contexte requis (client) avec un contexte fourni (service Cloud). Chaque

2. URI est l'abréviation de la chaîne de caractère : "Uniform Resource Identifier".

3. SaaS : logiciel en tant que service, PaaS : plate-forme en tant que service et IaaS : infrastructure en tant que service.

4. Dans la littérature anglaise, on utilise le terme "Location" pour désigner l'emplacement du fournisseur Cloud.

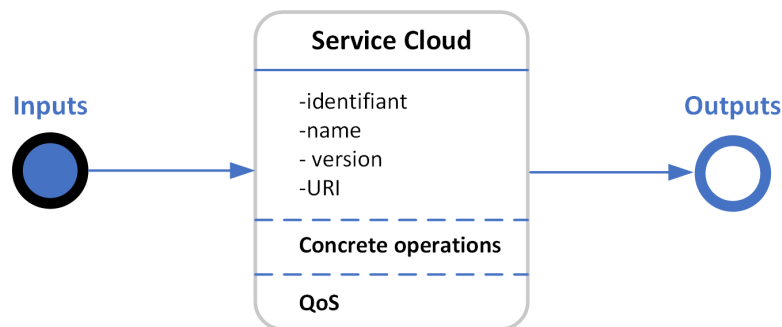


FIGURE 3.3 Représentation d'un service concret (service Cloud)

service Cloud est décrit par un ensemble de propriétés de QoS et de contexte. Les services Cloud sont organisés en hiérarchies de classes relatives à la catégorie métier de l'application (ventes, achats, etc.). La figure 3.4 montre le modèle sémantique de service et toutes ses caractéristiques.

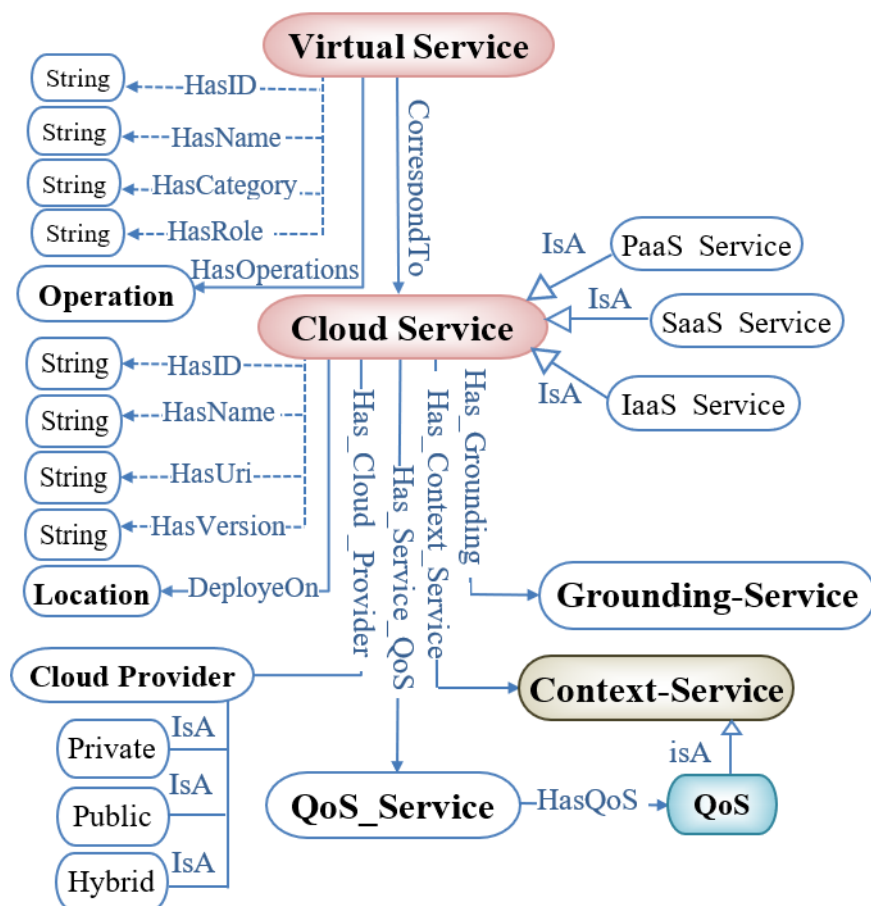


FIGURE 3.4 Modèle sémantique du service

## 4 Modèle sémantique de QoS à deux dimensions

Afin d'obtenir des valeurs de QoS fiables, des benchmarks sont souvent utilisés pour évaluer les QoS des différents services Cloud. Le rôle de ces benchmarks est de mesurer la QoS de chaque service publié dans le Cloud en utilisant des tests fréquents à chaque période déterminée par la stratégie du benchmark. Dans le but de garantir un processus d'affaire ERP de qualité à long terme pour le client, le modèle QoS de l'ontologie proposée a été conçu sémantiquement en tant que modèle bidimensionnel : QoS-Utility ( $QoS_U$ ) et QoS-LongLife ( $QoS_L$ ) (voir figure 3.5). Une interface graphique est fournie au client pour spécifier ses contraintes et ses préférences en termes de QoS.

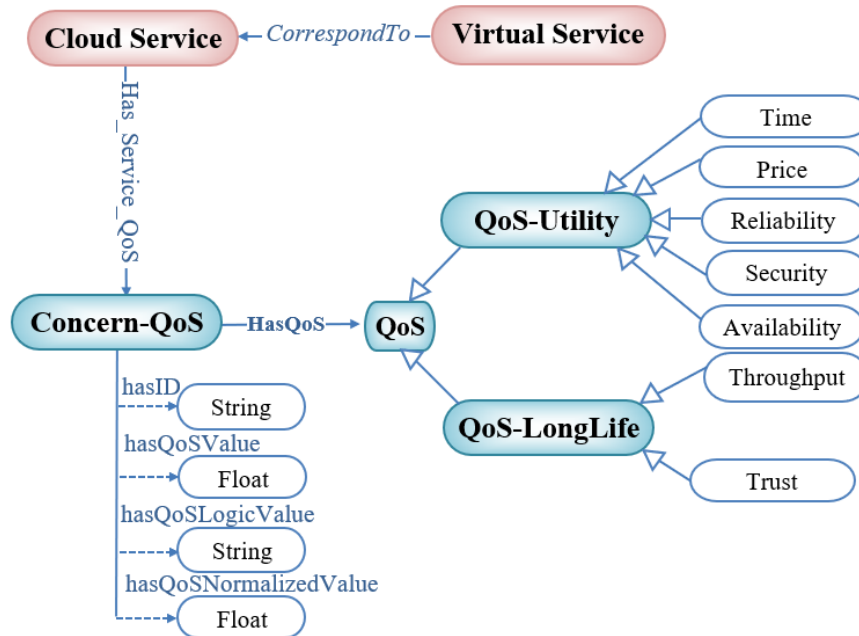


FIGURE 3.5 Modèle sémantique de QoS

### 4.1 Modèle de QoS-Utilité

Le client peut définir ses besoins non-fonctionnels en tant que contraintes et préférences de QoS pour un processus d'affaire ERP donné. L'approche proposée recherche un nombre fini de services Cloud pour optimiser le processus d'affaire ERP composé qui respecte les contraintes du client et répond à ses préférences en matière de QoS d'utilité. Notre modèle de QoS d'utilité contient les attributs de QoS les plus utilisés : *prix*, *temps d'exécution*, *disponibilité*, *fiabilité* et *sécurité*. Le Tableau 3.1 illustre la description des attributs de QoS-Utilité.

TABLE 3.1 Description du modèle des critères de QoS-Utilité

| Attributs de QoS         | Description                                                                                                                                                    |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Prix</b>              | Il mesure les frais d'invocation d'un service Cloud pour chaque requête.                                                                                       |
| <b>Temps d'exécution</b> | C'est le temps nécessaire pour traiter et compléter une demande de service Cloud.                                                                              |
| <b>Disponibilité</b>     | C'est l'aspect qualitatif de savoir si le service Cloud est présent ou prêt pour une utilisation immédiate.                                                    |
| <b>Fiabilité</b>         | Cela représente le degré de capacité à maintenir le service et sa qualité.                                                                                     |
| <b>Sécurité</b>          | C'est le degré de sécurité d'un service Cloud. Le critère de sécurité vise à garantir les objectifs suivants : confidentialité, authentification et intégrité. |

## 4.2 Modèle de QoS-LongueVie

Le maintien et l'amélioration de l'ERP tout au long du cycle de vie doit prendre en compte l'impact profond des expériences des clients de services Cloud pertinents. En effet, on doit définir la manière d'utilisation du processus d'affaire ERP à long terme en fonction du contexte du client. Nous proposons un nouveau modèle de QoS "longue vie" qui combine les ressources et les attributs du contexte client (capacités réseau, consommation d'énergie, confiance, accessibilité, évolutivité) afin d'assurer la persistance du processus d'affaire ERP. Khurana et al. ont proposé de nombreuses mesures contextuelles [109], dont deux sont utilisées dans notre travail : le débit de service (throughput) et la confiance de service (trust). Des débits plus élevés et des valeurs de confiance plus élevées représentent de bons résultats de performance et une longue durée de vie des processus d'affaire pertinents. Le tableau 3.2 illustre notre modèle de critères de QoS-LongueVie.

TABLE 3.2 Description du modèle des critères de QoS-LongueVie

| Attributs de QoS | Description                                                                                                              |
|------------------|--------------------------------------------------------------------------------------------------------------------------|
| <b>Confiance</b> | Il s'agit du niveau de garantie d'un service Cloud donné dans un niveau de confiance spécifié.                           |
| <b>Débit</b>     | Il mesure le nombre de demandes de service Cloud exécutées avec succès dans une période et un emplacement client donnés. |

### 4.3 Modèle de QoS

Le modèle de QoS-Utilité et le modèle de QoS-LongueVie ont plusieurs propriétés communes, ce qui conduit à définir le modèle QoS comme suit :

$$QoS = QoS_U \cup QoS_L.$$

La QoS d'un service Cloud candidat  $CS_i$  est définie comme un vecteur d'attributs  $QoS(CS_i) = \langle q_{i,1}, q_{i,2}, \dots, q_{i,n} \rangle$ , où  $n$  représente le nombre d'attributs de QoS requis par le client et  $q_{i,j}$  représente la valeur du  $j^{eme}$  attribut de QoS ( $1 \leq j \leq n$ ) du service Cloud  $CS_i$ . Les attributs de QoS sont divisés en deux classes : les attributs positifs et les attributs négatifs. Les valeurs des attributs positifs (disponibilité, fiabilité, sécurité, confiance et débit) doivent être maximisées et les valeurs des attributs négatifs (prix et temps d'exécution) doivent être minimisées. Pour traiter les différentes valeurs de QoS, une phase de mise à l'échelle peut être effectuée pour normaliser les valeurs des attributs de QoS. La mise à l'échelle est une phase de prétraitement pour normaliser les valeurs de QoS associées aux attributs positifs et négatifs en les transformant à des valeurs comprises entre 0 et 1 par les formules suivantes :

- Normalisation des attributs de QoS positifs (dénnoté par  $QoS^+$ )

$$q'_{(i,j)} = \begin{cases} \frac{q_{(i,j)} - q_j^{min}}{q_j^{max} - q_j^{min}} & \text{If } q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{Otherwise} \end{cases} \quad 1 \leq j \leq n \quad (3.1)$$

- Normalisation des attributs de QoS négatifs (dénnoté par  $QoS^-$ )

$$q'_{(i,j)} = \begin{cases} \frac{q_j^{max} - q_{(i,j)}}{q_j^{max} - q_j^{min}} & \text{If } q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{Otherwise} \end{cases} \quad 1 \leq j \leq n \quad (3.2)$$

tel que :

- $q'_{(i,j)}$  est la valeur normalisée du  $j^{eme}$  attribut de QoS du service Cloud  $CS_i$ .
- $q_{(i,j)}$  est la valeur courante du  $j^{eme}$  attribut de QoS du service Cloud  $CS_i$ .
- $q_j^{min}$  est la valeur minimale du  $j^{eme}$  QoS.
- $q_j^{max}$  est la valeur maximale du  $j^{eme}$  QoS.

## 5 Modèle sémantique du contexte

Comme le processus d'affaire est défini par le client (l'entreprise) et le service Cloud de deux points de vue différents, il prend en compte deux classes de contexte : le client et le service Cloud. Le contexte de service Cloud représente les services Cloud et leurs paramètres de QoS, alors que le contexte client représente les besoins fonctionnels et non-fonctionnels requis par le client. En outre, le contexte est lié à l'environnement qui doit être pris en compte pour garantir une bonne mise en œuvre et une bonne performance de la gestion de l'entreprise dans des conditions différentes. Notre ontologie utilise ces informations en confrontant le contexte requis du client avec celui du service Cloud fourni pour la sélection et la composition des services Cloud dans un environnement dynamique. La classe du contexte est divisée en trois sous-classes principales, comme illustré dans la figure 3.6.

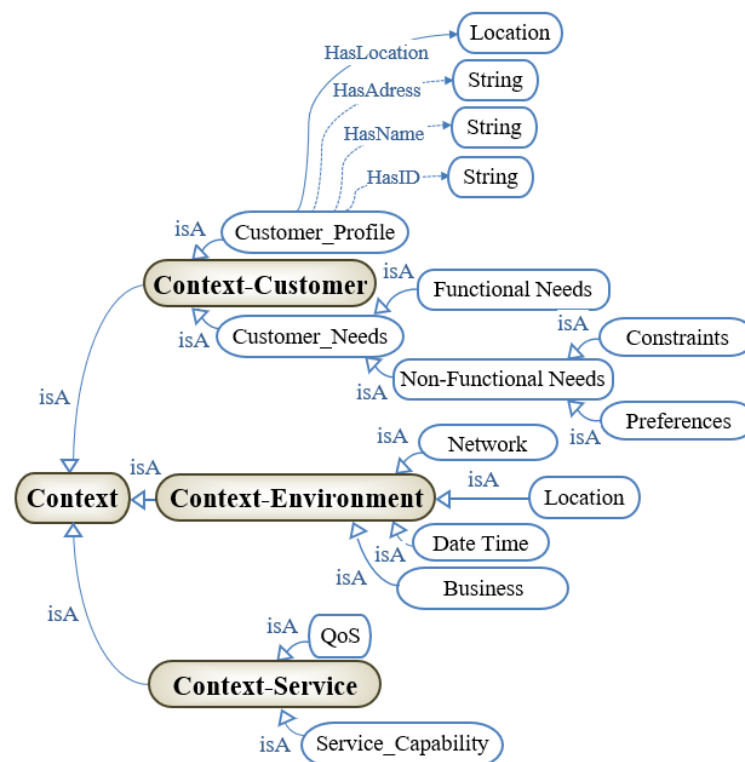


FIGURE 3.6 Modèle sémantique du contexte

### 5.1 Le contexte du service Cloud

Le contexte du service Cloud est défini par les capacités et les QoS du service. Les capacités du service Cloud sont constituées de paramètres sémantiques du service tels que :

son rôle, son type, ses opérations et ses entrées/sorties. Les critères de QoS incluent certaines métriques évaluant la qualité du service Cloud. Il est important de distinguer deux services Cloud fournissant les mêmes fonctionnalités avec différentes valeurs d'attributs de QoS.

## 5.2 Le contexte du Client

1. **Le profil du client** : décrit les activités de l'entreprise (client) et ses informations essentielles (nom, rôle, identifiant, adresse, adresse e-mail, etc.).
2. **Les besoins fonctionnels du Client** : décrivent les différentes fonctionnalités correspondantes aux activités du client. Le processus d'affaire fourni doit satisfaire les besoins exigés par le client.
3. **Les besoins non-fonctionnels du Client** : décrivent la façon et la qualité des fonctionnalités à obtenir par le client. Ces besoins peuvent être exprimés par des contraintes et des préférences.
  - a) **Les contraintes du client** : décrit les limites et les restrictions exigées par le client concernant les propriétés non-fonctionnelles (QoS). L'objectif de l'approche proposée est de respecter les contraintes qualitatives des clients sans aucune violation lors de la sélection du service composite optimal. Les contraintes du client sont exprimées par des clauses logiques. Par exemple, un client veut obtenir un service composite avec une sécurité haute et un prix moins cher. Cette contrainte est exprimée par la clause : (la sécurité du service = "High" AND Prix du service = "High"). En effet, une interface utilisateur graphique est fournie pour permettre aux clients de sélectionner des valeurs de contraintes sémantiques (*very low*, *low*, *medium*, *high*, *very high*) pour chaque critère de QoS. Ces valeurs sémantiques sont reliées à nb intervalles voisins  $I_1, I_2, \dots, I_{nb}$  tels que  $I_i \in [0, 1]$ ,  $\cup_{i=1}^{nb} I_i = [0, 1]$  et  $\cap_{i=1}^{nb} I_i = \emptyset$  où nb est le nombre de valeurs de contraintes sémantiques.

Dans notre modèle, nous avons divisé les contraintes sémantiques en trois valeurs suivantes.

- **High** correspond à une valeur de QoS normalisée appartenant à l'intervalle  $[qmin_i^{High}, qmax_i^{High}]$ .
- **Medium** correspond à une valeur de QoS normalisée appartenant à l'intervalle  $[qmin_i^{Medium}, qmax_i^{Medium}]$ .
- **Low** correspond à une valeur de QoS normalisée appartenant à l'intervalle  $[qmin_i^{Low}, qmax_i^{Low}]$ .

TABLE 3.3 Correspondance entre les spécifications linguistiques du client et les valeurs sémantiques

|                      | <b>High</b> | <b>Medium</b> | <b>Low</b> |
|----------------------|-------------|---------------|------------|
| <b>Prix</b>          | Cheap       | Medium        | Expensive  |
| <b>Temps</b>         | Fast        | Medium        | Slow       |
| <b>Confiance</b>     |             |               |            |
| <b>Disponibilité</b> |             |               |            |
| <b>Fiabilité</b>     | High        | Medium        | Low        |
| <b>Débit</b>         |             |               |            |
| <b>Sécurité</b>      |             |               |            |

Le client peut spécifier ses contraintes par des expressions en langage courant (temps rapide, prix cher, etc), donc nous transformons chaque spécification de contrainte client en valeur sémantique correspondante. Le tableau 3.3 montre la correspondance entre la spécification des contraintes client et les valeurs de contraintes sémantiques.

La  $i^{eme}$  contrainte client explicite (notée  $C_i$ ) de l'attribut  $q_i$  est convertie en valeur scalaire qui représente la limite inférieure de l'intervalle correspondant (équation 3.3) :

$$C_i = qmin_i^{valeur\_smantique} \quad (3.3)$$

- b) Les préférences du client :** Les préférences du client sensibles au contexte peuvent être utiles pour sélectionner un seul service composite concret avec des QoS optimisées. Pour cela, le client doit indiquer un niveau de préférence pour chaque attribut de QoS à l'aide d'une interface graphique. Chaque client a ses préférences spécifiques. Par exemple, un client considère plus d'importance à la sécurité d'un processus d'affaire par rapport à son temps d'exécution, tandis qu'un autre accorde une plus grande importance au temps d'exécution par rapport à la sécurité. Ceci est très intéressant dans le cas d'avoir deux services composites fournissant les mêmes fonctionnalités métiers avec des qualités différentes. Les préférences de QoS sont ordonnées par le client sous forme d'une liste, de la préférence la plus importante jusqu'à la moins importante. Cet ordre affecte un nombre à chaque attribut  $q_i$  de QoS, appelé *explicitPriority*. Ensuite, un poids est attribué à chaque  $q_i$  en fonction de son importance spécifiée par le client. Le poids

de chaque attribut  $q_i$  est calculé par l'équation 3.4 en adoptant le taux exponentiel au lieu du taux linéaire pour mieux valoriser les priorités des critères de QoS.

$$w_i = \frac{e^{P_i}}{\sum_{j=1}^n e^{P_j}} \quad (3.4)$$

$P_i$  est la priorité calculée de la QoS  $q_i$  définit par l'équation suivante :

$$P_i = \text{explicitPriority}_i + \text{constraintRank}_i \times n \quad (3.5)$$

tel que :

- $n$  est le nombre d'attributs de QoS.
- $\text{explicitPriority}$  est un nombre varie entre 1 à  $n$  alloué à chaque préférence de QoS selon l'ordre spécifié par le client. (c'est-à-dire de la préférence du dernier élément de la liste jusqu'à la préférence du premier élément). Si le client ne spécifie aucune préférence pour un attribut de QoS donnée, alors son  $\text{explicitPriority}$  est nulle.
- $\text{constraintRank}$  est un nombre entier qui indique la valeur de l'importance de la contrainte QoS (0 pour "none", 1 pour "low", 2 pour "medium" et 3 pour "high").

**Lemme**

Soit  $QoS_i$  et  $QoS_j$  deux attributs de QoS.  $C_i$  et  $C_j$  sont leurs contraintes respectivement et  $P_i$  et  $P_j$  sont leurs priorités. Si la contrainte d'une QoS  $C_i$  est supérieure à une autre  $C_j$ , alors la priorité  $P_i$  est forcément supérieure à  $P_j$ . Cette propriété est exprimée formellement par l'équation suivante :

$$\forall i, j : C_i > C_j \Rightarrow P_i > P_j$$

**Preuve :**

Selon l'équation 3.5, on a :

$$P_i = \text{explicitPriority}_i + \text{constraintRank}_i \times n \text{ et}$$

$$P_j = \text{explicitPriority}_j + \text{constraintRank}_j \times n$$

ce qui implique que :

$$P_i - P_j = \text{explicitPriority}_i - \text{explicitPriority}_j + (\text{constraintRank}_i - \text{constraintRank}_j) \times n$$

D'autre part on a :  $\text{constraintRank}_i - \text{constraintRank}_j \geq 1 \Rightarrow (\text{constraintRank}_i - \text{constraintRank}_j) \times n \geq n$

En plus,  $explicitPriority_i - explicitPriority_j \geq -n$  puisque  $1 \leq explicitPriority_i \leq n$  et  $1 \leq explicitPriority_j \leq n$

Donc  $P_i - P_j > 0 \Rightarrow P_i > P_j$

### 5.3 Le contexte de l'environnement

Il représente les informations disponibles caractérisant les connexions réseau, l'heure, la date, l'emplacement et les contraintes du domaine métier. Ces informations sont nécessaires pour qu'un processus d'affaire fonctionne correctement et soit persistant aux changements de l'environnement. Dans le domaine des affaires, nous distinguons le mode B2C (Business to Consumer) et le mode B2B (Business to Business). Le mode B2B correspond aux techniques utilisées dans le cadre d'une activité commerciale entre deux entreprises, alors que le mode B2C correspond aux techniques destinées au grand public dont elles sont utilisées dans le cadre d'une activité commerciale entre une entreprise et les clients.

## 6 Modèle sémantique du processus d'affaire

Un processus d'affaire (ou un service composite) décrit les différents types de processus d'affaire tels que les ventes, les achats, les opportunités et les commandes. La figure 3.7 montre le modèle sémantique de processus d'affaire avec ses propriétés et les entités associées. On distingue deux types de processus d'affaires : le processus d'affaire virtuel et le processus d'affaire concret.

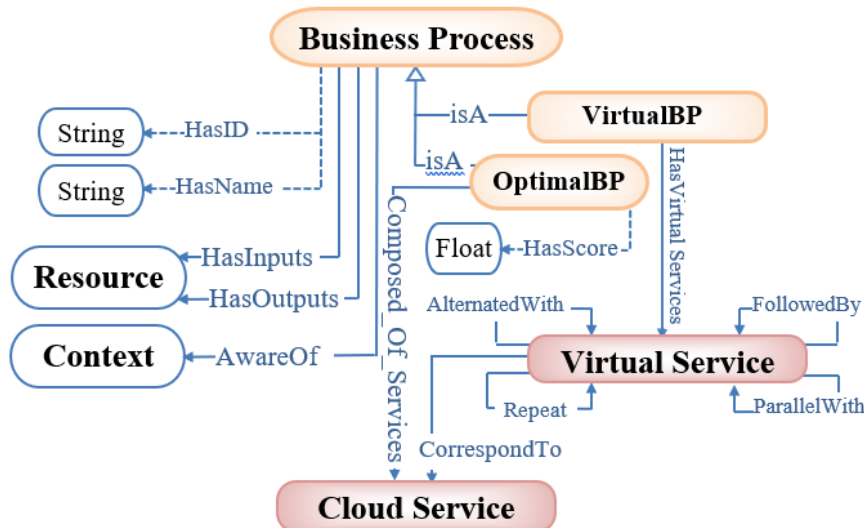


FIGURE 3.7 Le modèle sémantique du processus d'affaire

## 6.1 Processus d'affaire virtuel

Un processus d'affaire virtuel est composé d'un ensemble de services virtuels exécutés par un groupe d'utilisateurs qui participent au processus de gestion des ressources d'entrée et de sortie. L'ensemble de services virtuels peut être structurée par quatre modèles. Le premier modèle est le modèle "FollowedBy" qui représente l'exécution séquentielle d'un ensemble de services. Le deuxième est le modèle "Repeat" qui permet d'exécuter en boucle un service plusieurs fois. Le troisième modèle est le modèle "ParallelWith" qui signifie l'exécution de plusieurs services simultanément. Le dernier modèle est le modèle "AlternateWith" qui exécute une branche de services selon une condition donnée. La figure 3.8 montre les différents modèles de composition de services Cloud.

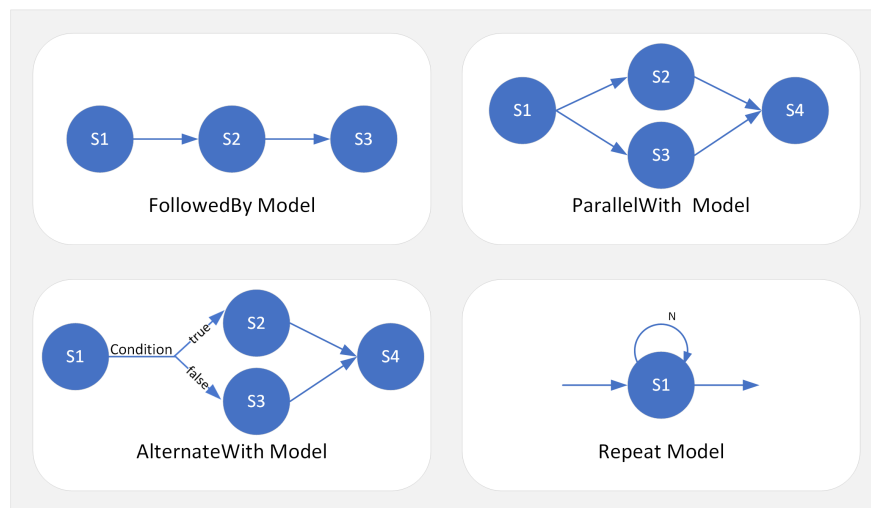


FIGURE 3.8 Les modèles de composition

## 6.2 Processus d'affaire concret

Un processus d'affaire concret ou service composite Concret (CCS : Concrete Composite Service) est l'ensemble des services concrets sélectionnés et composés en utilisant une stratégie de composition telle que la correspondance entre les sorties d'un service et les entrées du service suivant. L'évaluation de la qualité du CCS est définie par une évaluation des critères  $QoS_U$  et  $QoS_L$ . La première évaluation dépend des propriétés du service, à savoir le temps d'exécution, les ressources disponibles, la portabilité, la fiabilité et la flexibilité. La seconde dépend de l'environnement et du contexte du client. Pour prendre en charge la composition dynamique de services Cloud personnalisés avec des fournisseurs de Cloud hétérogènes, la qualité de service est configurable pour chaque client à l'aide des fonctions

d'évaluations agrégées. Les valeurs de QoS normalisées de CCS pour les attributs négatifs et positifs sont calculées par les équations suivantes :

- Normalisation des attributs agrégés de QoS positifs ( $q_i \in QoS^+$ )

$$q'_j(CCS) = \begin{cases} \frac{agg(q_j) - agg(q_j^{min})}{agg(q_j^{max}) - agg(q_j^{min})} & \text{If } agg(q_j^{max}) \neq agg(q_j^{min}) \\ 1 & \text{Otherwise} \end{cases} \quad (3.6)$$

- Normalisation des attributs agrégés de QoS négatifs ( $q_i \in QoS^-$ )

$$q'_j(CCS) = \begin{cases} \frac{agg(q_j^{max}) - agg(q_j)}{agg(q_j^{max}) - agg(q_j^{min})} & \text{If } agg(q_j^{max}) \neq agg(q_j^{min}) \\ 1 & \text{Otherwise} \end{cases} \quad (3.7)$$

où  $agg(q_j^{min})$  et  $agg(q_j^{max})$  représentent le minimum et le maximum des valeurs agrégées du  $j^{eme}$  QoS du CCS respectivement. Et  $agg(q_j)$  dénote la valeur agrégée du critère QoS du CCS. Le Tableau 3.4 illustre les fonctions d'agrégation QoS du CCS.

TABLE 3.4 Les fonctions d'agrégations de QoS

| Les attributs de QoS | Séquentiel    | Parallèle             | choix                 | Boucle       |
|----------------------|---------------|-----------------------|-----------------------|--------------|
| $QoS_U$              | Temps         | $\sum_{j=1}^n Time_j$ | $Max_{j=1}^n Time_j$  | $Time_j * k$ |
|                      | Disponibilité | $\prod_{j=1}^n Av_j$  | $\prod_{j=1}^n Av_j$  | $(Av_j)^k$   |
|                      | Fiabilité     | $\prod_{j=1}^n Re_j$  | $\prod_{j=1}^n Re_j$  | $(Re_j)^k$   |
|                      | Sécurité      | $Min_{j=1}^n Sec_j$   | $Min_{j=1}^n Sec_j$   | $Sec_j$      |
|                      | Prix          | $\sum_{j=1}^n Cost_j$ | $Max_{j=1}^n Cost_j$  | $Cost_j * k$ |
| $QoS_L$              | Confiance     | $Min_{j=1}^n Trust_j$ | $Min_{j=1}^n Trust_j$ | $Trust_j$    |
|                      | Débit         | $Min_{j=1}^n Thr_j$   | $Min_{j=1}^n Thr_j$   | $Thr_j$      |

## 7 L'interrogation et l'inférence

La logique de description (DL<sup>5</sup>) [110] est un langage de représentation des connaissances d'une manière formelle et structurée. Elle permet la description des concepts sémantiques pour décrire un domaine par des prédicats du premier ordre. Elle est utilisée comme extension

5. DL : Description Logic ; appelée en français "Logique de description ou "Logique descriptive."

des langages qui ne possèdent pas de sémantique formelle basée sur la logique telles que les ontologies. La plupart des DL divisent la connaissance en deux parties :

- **TBox** : permet de décrire les informations terminologiques en définissant les notions basiques ou dérivées et de la façon dont elles sont reliées entre elles. Ces informations sont génériques ou globales, vraies dans tous les modèles et pour tous les individus.
- **ABox** : contient les informations qui caractérisent les individus. Ces informations sont exprimées par des assertions spécifiques ou locales, vraies pour certains individus particuliers.

L'objectif principal de notre ontologie est la gestion efficace du processus de composition des services Cloud ERP. En effet, elle permet aux clients de chercher, sélectionner et de composer des BPs. L'interaction entre l'ontologie et le système d'optimisation de la composition des services Cloud est effectué par des requêtes SPARQL (SPARQL Protocol and RDF Query Language) [111] et des Règles SWRL (Semantic Web Rule Language) [112] au cours du processus d'optimisation.

Le tableau 3.5 représente une requête SPARQL qui permet au système d'extraire la liste des services Cloud concrets correspondants à un service Virtuel donné.

TABLE 3.5 Requête d'interrogation des services Cloud en SPARQL.

---

```

PREFIX cloudERPOnto : <http://www.owlontologies.com/EO.owl#>
PREFIX rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?sid ?sname ?slocation ?qos_service ?qos_value
WHERE {
    ?sc cloudERPOnto :HasCloudServiceID ?sid .
    ?sc cloudERPOnto :HasCloudServiceLocation ?slocation .
    ?sc cloudERPOnto :HasCloudServiceName ?sname .
    ?sv cloudERPOnto :HasConcretService ?sc .
    ?qos_service cloudERPOnto :QoS_Concern_Service ?sc .
    ?qos_service cloudERPOnto :HasQoSValue ?qos_value .
    ?sv cloudERPOnto :TaskHasID \"\" + task_id + \"\" .
} ;

```

---

Le tableau 3.6 décrit une règle SWRL qui permet d'inférer les services virtuels appropriés selon les besoins fonctionnels spécifiés par le client. Les besoins sont spécifiés à l'aide d'une interface graphique et, ils sont transformés en expressions logiques comme expliqué dans la section 5.2.

Le tableau 3.7 présente des exemples de TBox et ABox de notre ontologie.

TABLE 3.6 Règle d'inférence des services virtuels d'un BP en SWRL.

---

```

Virtualized_Busniess_Process( ?bp ) ∧ Has_Virtual_Services( ?bp, ?vs ) ∧
HasStep( ?vs, "current" ) ∧ FollowedBy( ?vs, ?vs1 ) ∧ VSHasInput( ?vs1, ?in1 ) ∧
ParameterSemantic( ?in1, ?s1 ) ∧ FunctionalNeeds( ?needs ) ∧
ExpressedBy( ?needs, ?cexp ) ∧ BusinessType( ?cexp, "Sale" ) ∧
ExpressionComposedOf( ?cexp, ?sexp ) ∧ HasParameter( ?sexp, ?param_in ) ∧
ParameterSemantic( ?param_in, ?s ) ∧ swrlb :equal( ?s, ?s1 ) →
Has_Virtual_Services( ?bp, ?vs1 ) ∧ HasStep( ?vs, "passe" ) ∧ HasStep( ?vs1, "current" )

```

---

## 8 Exemple de PA selon l'ontologie CxQSCloudSERP

La figure 3.9 montre un exemple de modèle concret basé sur l'ontologie CxQSCloudSERP représenté par OWL. Il s'agit d'un processus de vente en ligne "Online Sale Process".

L'exemple présenté montre l'application de l'ontologie CxQSCloudSERP. L'attribut ID (ligne 1) représente un identifiant unique de processus généré par le composant annotateur pour faire référence à un processus d'affaire sémantique. Le processus d'affaire est composé de trois services virtuels : le service de connexion du client, le service de détails du produit et le service d'achat en ligne (ligne 1-8). Les services virtuels sont exécutés séquentiellement. Le système doit d'abord vérifier le nom et le mot de passe du client, ensuite il doit sélectionner les détails des produits et enfin se charge du paiement en ligne. Les services virtuels sont décrits par la classe VirtualService et ils sont liés entre eux par les relations d'enchaînement (FollowedBy). Chaque service virtuel décrit par des informations sémantiques. Par exemple, le service virtuel "service de connexion client" décrit les informations suivantes : catégorie de service (vente) et le rôle de service Cloud (détails du client) sur lesquels les paramètres seront attachés (paramètre\_in = "nom du client", mot de passe = "chaîne de caractère", paramètre\_out = "booléen", paramètres d'action = "vérifier les détails du client") (ligne 10-20). Chaque service virtuel est associé à un ensemble de services Cloud concrets. Chaque service concret a deux QoS d'utilité (ligne 21 - 25). La première QoS d'utilité (ligne 26 - 30) représente le prix du service Cloud (prix = "100 \$", prix normalisé = "0,78" et valeur sémantique = "High"). La deuxième QoS représente le temps d'exécution du service Cloud (temps = "10", temps normalisé = "0,75" et valeur sémantique = "High") (ligne 32 - 37). La QoS Longue-vie (ligne 38 - 43) représente le débit du service Cloud (débit = "4000", débit normalisé = "0,82" et valeur sémantique = "High").

TABLE 3.7 Exemple de DL de notre ontologie.

| <b>TBox</b>                                                   | <b>ABox</b>                                                                                                                                                            |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $BP \sqsubseteq Thing$                                        | $BP(Online\_Sale\_BP)$                                                                                                                                                 |
| $\geq 1HasID \sqsubseteq String$                              | $hasID(Online\_Sale\_BP, ID1)$                                                                                                                                         |
| $\geq 1HasName \sqsubseteq String$                            | $hasName(Online\_Sale\_BP, Sale\_BP)$                                                                                                                                  |
| $\geq 1HasInputs \sqsubseteq Resource$                        | $hasInputs(Online\_Sale\_BP, I1)$                                                                                                                                      |
| $\geq 1HasOutputs \sqsubseteq Resource$                       | $hasOutputs(Online\_Sale\_BP, O1)$                                                                                                                                     |
| $\geq 1HasVirtualService \sqsubseteq$<br>$VirtualService$     | $HasVirtualService(Online\_Sale\_BP, CustomerLogin)$<br>$HasVirtualService(Online\_Sale\_BP, ProductDetails)$<br>$HasVirtualService(Online\_Sale\_BP, OnlinePurchase)$ |
| $VirtualService \sqsubseteq Thing$                            | $VirtualService(CustomerLoginService)$                                                                                                                                 |
| $\geq 1HasName \sqsubseteq String$                            | $HasName(CustomerLoginService, LoginCustomer)$                                                                                                                         |
| $\geq 1HasOperations \sqsubseteq Operation$                   | $HasOperations(CustomerLoginService, Op1)$                                                                                                                             |
| $\geq 1HasRole \sqsubseteq String$                            | $Role(CustomerLoginService, Login)$                                                                                                                                    |
| $\geq 1HasCategory \sqsubseteq String$                        | $Category(CustomerLoginService, Authentification)$                                                                                                                     |
| $\geq 1CorrespondTo \sqsubseteq$<br>$CloudService$            | $CorrespondTo(CustomerLoginService, SaaS_1)$<br>$CorrespondTo(CustomerLoginService, SaaS_2)$                                                                           |
| $CloudService \sqsubseteq Thing$                              | $CloudService(SaaS_1)$                                                                                                                                                 |
| $\geq 1HasID \sqsubseteq String$                              | $Has\_Service\_Qos(SaaS_1, QoS_1)$                                                                                                                                     |
| $\geq 1Has\_Context\_Service \sqsubseteq$<br>$ContextService$ | $Has\_Service\_Qos(SaaS_1, QoS_2)$<br>$Has\_Service\_Qos(SaaS_1, QoS_3)$                                                                                               |
| $\geq 1Has\_Service\_QoS \sqsubseteq$<br>$QoS\_Service$       |                                                                                                                                                                        |
| $QoS \equiv QoS_{Utility} \sqcup QoS_{LongLife}$              | $QoS(QoS_1)$                                                                                                                                                           |
| $QoS \sqsubseteq Thing$                                       | $HasID(QoS_1, QoS_1_1)$                                                                                                                                                |
| $\geq 1HasID \sqsubseteq String$                              | $HasName(QoS_1, Price)$                                                                                                                                                |
| $\geq 1HasName \sqsubseteq String$                            | $HasQoSValue(QoS_1, 100)$                                                                                                                                              |
| $\geq 1HasQoSValue \sqsubseteq Float$                         | $HasQoSLogicValue(QoS_1, High)$                                                                                                                                        |
| $\geq 1HasQoSLogicValue \sqsubseteq String$                   | $HasQoSNormalizedValue(QoS_1, 0.78)$                                                                                                                                   |
| $\geq 1HasQoSNormalizedValue \sqsubseteq$<br>$String$         |                                                                                                                                                                        |

---

```

(01) <CxQSCloudSERP: Busniess_Process rdf:ID="Online_Sale_BP">
(02)   < CxQSCloudSERP:hasID>BP001</ CxQSCloudSERP:hasID>
(03)   < CxQSCloudSERP:hasNom> Online_Sale_BP </CxQSCloudSERP:hasNom>
(04)   < CxQSCloudSERP:hasVirtualService rdf:resource =#CustomerLoginService />
(05)     FollowedBy =#ProductDeatilsService AlternateWith =#NewCustomer />
(06)   < CxQSCloudSERP:hasVirtualService rdf:resource =#ProductDeatilsService/>
(07)     FollowedBy =#OnlinePurchaseService Repeat = #ProductDeatilsService/>
(08)   < CxQSCloudSERP:hasVirtualService rdf:resource =#OnlinePurchaseService />
(09) </CxQSCloudSERP:Busniess_Process>
(10) < CxQSCloudSERP:VirtualService rdf:ID="CustomerLoginService">
(11)   < CxQSCloudSERP: HasCategory>Sale</CxQSCloudSERP: HasCategory>
(12)   < CxQSCloudSERP: HasRole> Customer Details</CxQSCloudSERP: HasRole>
(13)   < CxQSCloudSERP: hasOperation rdf:ID="OP_1" rdf:Name="CustomerLoginOp">
(14)     < CxQSCloudSERP: hasParemeters> CustomerName, Password, LoggedInOut
(15)     </CxQSCloudSERP: hasParemeters>
(16)   < /CxQSCloudSERP: hasOperation>
(17)   < CxQSCloudSERP: CorrespondTo rdf:resource=#SaaS_CustomerLogService_1>
(18)   < CxQSCloudSERP: CorrespondTo rdf:resource=#SaaS_CustomerLogService_2>
(19)   ....
(20) </CxQSCloudSERP:VirtualService >
(21) < CxQSCloudSERP: CloudService rdf:ID="SaaS_CustomerLogService_1" >
(22)   < CxQSCloudSERP: hasQoS rdf:resource =#QoS_1_1 >
(23)   < CxQSCloudSERP: hasQoS rdf:resource =#QoS_1_2 >
(23)   < CxQSCloudSERP: hasQoS rdf:resource =#QoS_1_3 >
(24) </ CxQSCloudSERP: CloudService " >
(25)   ....
(26) < CxQSCloudSERP:QoS rdf:ID="QoS_1_1">
(27)   < CxQSCloudSERP:hasName>Price</CxQSCloudSERP:hasName>
(28)   < CxQSCloudSERP:hasQoSValue> 100</ CxQSCloudSERP: hasQoSValue >
(29)   < CxQSCloudSERP: hasQoSLogicValue>High</ CxQSCloudSERP: hasQoSLogicValue >
(30)   < CxQSCloudSERP: hasQoSNormValue > 0.78</CxQSCloudSERP: hasQoSNormValue >
(31)   < /CxQSCloudSERP:QoS>
(32) < CxQSCloudSERP:QoS rdf:ID="QoS_1_2">
(33)   < CxQSCloudSERP:hasName>Time</CxQSCloudSERP:hasName>
(34)   < CxQSCloudSERP:hasQoSValue> 10 </ CxQSCloudSERP: hasQoSValue >
(35)   < CxQSCloudSERP: hasQoSLogicValue> High </CxQSCloudSERP: hasQoSLogicValue >
(36)   < CxQSCloudSERP: hasQoSNormValue >0.75 </CxQSCloudSERP: hasQoSNormValue >
(37)   < /CxQSCloudSERP:QoS>
(38) < CxQSCloudSERP:QoS rdf:ID="QoS_1_3">
(39)   < CxQSCloudSERP:hasName>Throughout</CxQSCloudSERP:hasName>
(40)   < CxQSCloudSERP:hasQoSValue> 4000 </ CxQSCloudSERP: hasQoSValue >
(41)   < CxQSCloudSERP: hasQoSLogicValue> High </ CxQSCloudSERP: hasQoSLogicValue >
(42)   < CxQSCloudSERP: hasQoSNormValue >0.82 </ CxQSCloudSERP: hasQoSNormValue>
(43)   < /CxQSCloudSERP:QoS>

```

---

FIGURE 3.9 Processus de vente en ligne représenté par CxQSCloudSERP

## 9 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle ontologie "CxQSCloudSERP" (Context-aware Quality of Semantic Cloud Service ERP) pour la description et la gestion efficace des services Cloud afin d'obtenir un processus d'affaire ERP pertinent sensible au contexte. Cette ontologie est constituée de cinq entités principales : Processus d'affaire, Contexte, Service virtuel, Service Cloud et QoS. Le processus d'affaire peut être un processus concret ou un processus virtuel. Un processus d'affaire concret est représenté par une composition de services concrets (services Cloud), alors qu'un processus d'affaire virtuel est composé de services virtuels et chaque service virtuel est un service abstrait associé à un ensemble de services Cloud ayant les mêmes fonctionnalités. Nous avons divisé le contexte en trois catégories : Contexte Client, Contexte Service et Contexte d'environnement. Le système doit prendre en considération ses différentes catégories pour mieux réagir aux changements des différents contextes. Concernant les exigences des clients, nous avons modélisé les besoins fonctionnels et les besoins non-fonctionnels (QoS) séparément. Ainsi, nous avons constaté que les besoins non-fonctionnels doivent être exprimés sous formes de contraintes et de préférences. Les contraintes représentent les limites que le processus d'affaire doit respecter, tandis que les préférences représentent une priorité pour chaque QoS. Cette priorité sert à calculer le poids de chaque QoS en utilisant une nouvelle formule exponentielle. Nous avons aussi proposé un nouveau modèle de QoS bidimensionnel dont la première dimension est focalisée sur les QoS utilité (*prix, temps d'exécution, disponibilité, fiabilité et sécurité*) et la deuxième dimension contient les QoS longue-vie (*confiance et débit*). Dans ce modèle, des valeurs sémantiques ont été attribuées à chaque critère de QoS.

Dans le chapitre suivant nous allons présenter une approche sémantique pour l'optimisation de la composition des services Cloud afin d'obtenir un processus d'affaire ERP sensible au contexte, optimal en termes de QoS. Nous allons baser sur notre ontologie qui joue le rôle de noyau de notre approche.

# Chapitre 4

## Nouvelle approche dédiée aux ERP sensibles au contexte à base d'ontologie

### Sommaire

---

|          |                                                                                                              |            |
|----------|--------------------------------------------------------------------------------------------------------------|------------|
| <b>1</b> | <b>Introduction . . . . .</b>                                                                                | <b>89</b>  |
| <b>2</b> | <b>Motivation : étude de cas et exigences . . . . .</b>                                                      | <b>90</b>  |
| 2.1      | Etude de cas . . . . .                                                                                       | 90         |
| 2.2      | Exigences . . . . .                                                                                          | 91         |
| <b>3</b> | <b>Architecture de composition d'ERP sensible au contexte . . . . .</b>                                      | <b>91</b>  |
| <b>4</b> | <b>Modèle fonctionnel . . . . .</b>                                                                          | <b>93</b>  |
| <b>5</b> | <b>Algorithme de composition et d'optimisation sensible au contexte basé<br/>sur la sémantique . . . . .</b> | <b>95</b>  |
| <b>6</b> | <b>Application de l'algorithme proposé à l'étude de cas . . . . .</b>                                        | <b>102</b> |
| 6.1      | Génération d'un processus d'affaire virtuel (VBP) optimal . . . . .                                          | 102        |
| 6.2      | Génération des solutions Pareto-optimal . . . . .                                                            | 102        |
| 6.3      | Sélection de la solution finale . . . . .                                                                    | 107        |
| <b>7</b> | <b>Nouvelle extension : CxQSCloudERP à base d'agents . . . . .</b>                                           | <b>107</b> |
| 7.1      | Architecture et principe de fonctionnement . . . . .                                                         | 111        |
| 7.2      | Description de l'algorithme SCwA-CCS-2S . . . . .                                                            | 114        |
| 7.3      | Application de l'algorithme SCwA-CCS-2S à l'étude de cas . . . . .                                           | 117        |
| <b>8</b> | <b>Conclusion . . . . .</b>                                                                                  | <b>119</b> |

---

## 1 Introduction

Les PME recherchent des solutions flexibles et efficaces pour automatiser convenablement leurs activités. En effet, elles souhaitent avoir un ERP personnalisé adapté aux besoins du client et aux changements d'environnement. La personnalisation est imposée en raison des activités spécifiques des différentes entreprises, ainsi que leurs évolutions éventuelles. En plus des besoins fonctionnels spécifiques des PME, plusieurs entreprises exigent des besoins non-fonctionnels, notamment les QoS. Pour répondre aux préoccupations de ces entreprises, la composition des services existants dans le Cloud est une solution efficace pour la génération d'ERP personnalisé, évolutive et flexible.

Dans le chapitre précédent, nous avons présenté l'ontologie CxQSCloudSERP qui joue le rôle du noyau de notre approche. Cette ontologie est utilisée pour décrire la sémantique des divers éléments de l'ERP personnalisé dans le Cloud avec ses différentes catégories de contexte. Elle permet de guider le processus de composition de services Cloud afin d'obtenir un processus d'affaire ERP qui satisfait les besoins du client dans un contexte spécifique. Le contexte est divisé en trois catégories : le client, le Cloud et l'environnement. Le contexte client à son tour est divisé en deux catégories verticales qui sont : les changements des besoins fonctionnels en raison des évolutions des activités de l'entreprise et les changements éventuels des contraintes et des préférences de QoS. L'ERP à offrir au client doit être adapté aux changements de tout type de contexte.

Dans ce présent chapitre nous proposons une approche qui vise à offrir aux entreprises un ERP personnalisé sensible au contexte en basant sur notre ontologie. Cette approche consiste en deux fonctionnalités principales. La première fonctionnalité génère automatiquement un processus d'affaire en tant que combinaison de services virtuels via les fonctions de raisonnement de notre ontologie. La deuxième offre au client un CCS (Composite Concrete Service) optimal en termes de QoS qui respecte ses contraintes et préférences contextuelles. L'approche met en évidence les besoins fonctionnels des clients d'une part, et facilite la recherche et la génération de meilleurs CCSs afin de satisfaire leurs contraintes et préférences non-fonctionnelles d'autre part. Le système permet au client de spécifier ses besoins fonctionnels avec ses contraintes et préférences via une interface utilisateur. Il génère d'abord un processus d'affaire virtuel optimal qui satisfait les besoins fonctionnels des clients grâce à l'application des règles de raisonnement à base d'ontologie. Ensuite, il applique un algorithme à deux phases pour optimiser les QoS des processus d'affaires engendrés. La première phase basée sur l'algorithme NSGA-II permet d'obtenir un ensemble de solutions Pareto-optimal qui respectent les contraintes du client. La deuxième est utilisée pour sélectionner la solution optimale finale qui doit être offerte au client.

Nous présentons également une nouvelle extension de notre approche qui utilise l'aspect multi-agent pour améliorer les performances de notre système.

## 2 Motivation : étude de cas et exigences

Le principal inconvénient des systèmes ERP est leur coût élevé et la difficulté de la personnalisation de ces ERP en cas de changement d'activités des entreprises. En réalité, les PME n'ont pas besoin d'un ERP complet avec toutes ses fonctionnalités génériques qui ne seront pas toutes utilisées. Ce qui exige une configuration spécifique à chaque entreprise, ainsi qu'une reconfiguration chaque fois l'entreprise évolue en activités. Les grands fournisseurs de systèmes ERP, tels que SAP, Oracle, Sage et Microsoft, positionnent leurs offres ERP dans le modèle SaaS [2]. Néanmoins, ces systèmes ne sont pas suffisamment flexibles pour supporter les évolutions dynamiques des besoins des entreprises avec leurs contraintes et préférences contextuelles. En effet, les PME recherchent un ERP sur mesure selon leurs activités et qui peut être adaptés facilement à leurs évolutions. Avec la prolifération des services Cloud, les développeurs s'intéressent à la réutilisation de ces services en penchant vers le paradigme de composition de service Cloud. Ce paradigme s'articule sur la collaboration des services existants dans le Cloud pour obtenir un processus d'affaire. L'avantage principal de ce paradigme est que les services sont payés à la demande, ainsi que la grande flexibilité en raison de l'atomisation de ces services qui peuvent être remplacés rapidement. Nous avons adopté ce paradigme en le combinant avec l'ontologie présentée dans le chapitre précédent pour générer un ERP sensible au contexte. Avant de décrire notre approche, nous allons présenter une étude de cas afin d'éclaircir l'optique de nos motivations et objectives.

### 2.1 Etude de cas

Nous présentons dans cette étude de cas un processus d'affaire B2B (Business to Business). Considérons qu'une entreprise (client) a besoin du processus d'affaire de "commande de produits en ligne". L'entreprise doit spécifier ses besoins fonctionnels en tant que type d'activité, les entrées/sorties du processus et autres informations supplémentaires. Elle peut également spécifier des besoins non-fonctionnels qu'ils s'agissent des QoS du processus d'affaire requis. Cette spécification s'exprime sous forme de contraintes et préférences non-fonctionnelles. Notre approche vise à fournir à l'entreprise un processus d'affaire qui satisfait tous ses besoins fonctionnels tout en optimisant ses besoins non-fonctionnels.

## 2.2 Exigences

Le passage à un ERP Cloud personnalisé et flexible doit être planifié avec soin en fonction des besoins actuels avec une vision à long terme qui prend en considération les changements du contexte. La conception d'un tel système doit respecter les exigences suivantes :

- **Gestion des besoins** : Le client spécifie visuellement ses besoins via une interface utilisateur. Les besoins fonctionnels sont traduits en règles sémantiques, tandis que les besoins non-fonctionnels sont liés à des valeurs quantitatives. Les besoins non-fonctionnels sont spécifiés sous forme de contraintes et préférences contextuelles.
- **Gestion des services** : Chaque service concret disponible dans le registre des services est associé à un service virtuel selon sa catégorie, son rôle, ses fonctionnalités et ses QoS. Un service virtuel est associé à l'ensemble des services concrets ayant les mêmes fonctionnalités.
- **Gestion de contexte** : Le système doit être flexible en réagissant continuellement aux changements du contexte. En effet, le registre des services est mis à jour à chaque changement d'un service Cloud (apparition, disparition ou modification). Ainsi qu'à chaque évolution du contexte d'environnement tel que le "débit" et à chaque changement de contexte du client en termes de contraintes et préférences, le système régénère le processus d'affaire ERP qui s'adapte à ces changements.
- **Optimisation des QoS** : Le processus d'affaire ERP généré par le système doit être optimal en termes de QoS. Par ailleurs, les QoS sont spécifiées par le client sous forme de contraintes et préférences. Vu que les attributs de QoS sont généralement conflictuels, le système doit trouver le meilleur compromis de QoS en respectant les contraintes et les préférences du client.
- **Composition automatique des services** : Le système est chargé de prendre en entrées les besoins fonctionnels et non-fonctionnels du client afin d'engendrer un processus d'affaire ERP personnalisé en utilisant la composition automatique de services Cloud.
- **Scalabilité et performance** : Les services existants dans le Cloud sont trop nombreux et hétérogènes. Le système est basé sur une ontologie pour traiter l'interopérabilité sémantique entre les descriptions hétérogènes des services Cloud pour avoir une représentation et une classification sémantique commune.

## 3 Architecture de composition d'ERP sensible au contexte

Dans cette section, nous proposons notre architecture de personnalisation et d'adaptation de l'ERP avec l'exploitation de notre modèle ontologique. L'ontologie facilite l'interopé-

tabilité sémantique des services Cloud, gère les qualités de services Cloud et renvoie le service composite optimal au client à partir des registres des services concrets. L'architecture utilise un moteur d'inférence et un algorithme de composition sémantique et dynamique pour optimiser le processus d'affaire ERP sensible au contexte [108]. L'architecture générale de notre approche est constituée de trois composants principaux comme illustrée dans la figure 4.1 :

- **L'entreprise cliente** : est l'entreprise qui a besoin d'un ERP personnalisé.
- **Proxy de l'ERP Cloud** : comprend un planificateur de service composite virtuel ERP (ERP-VCS planner), un moteur à deux phases pour la composition des services concrets (CCS-2S engine) et un annotateur (ERP-CS annotator).
  - ✓ *Planificateur ERP-VCS* : est un moteur d'inférence sémantique chargé de générer le processus d'affaire virtuel sous forme d'un Service Composite Virtuel (VCS) selon les besoins fonctionnels du client. Ce VCS est composé de services virtuels (tâches). Chaque service virtuel (VS) est lié à un ensemble de services Cloud concrets ayant les mêmes fonctionnalités avec différentes QoS.
  - ✓ *Moteur CCS-2S* : prend en entrée les contraintes et les préférences du client décrites dans notre ontologie et la description des services Cloud afin de produire un processus d'affaire optimal sous forme d'un service Cloud composite (CCS).
  - ✓ *Annotateur ERP-CS* : fournit un support pour annoter les informations sémantiques des services Cloud et une interface de communication entre le planificateur ERP-VCS et le moteur CCS-2S. Ce composant génère automatiquement des services Cloud sémantiques à l'aide de l'ontologie CxQSCloudSERP. En utilisant les métadonnées fournies à l'ontologie, l'annotateur intègre des informations sémantiques (catégorie de service, rôle de service, sémantique des paramètres, sémantique des actions et sémantique de QoS) du service Cloud. En effet, l'annotation sémantique des services Cloud aide à générer automatiquement des processus d'affaires virtuels.
- **Les registres de services Cloud et de profils clients** : ils stockent la description sémantique des services Cloud concrets, des services virtuels et les profils des clients dans notre ontologie sous forme de fichier RDF/XML afin de générer des processus d'affaires appropriés.

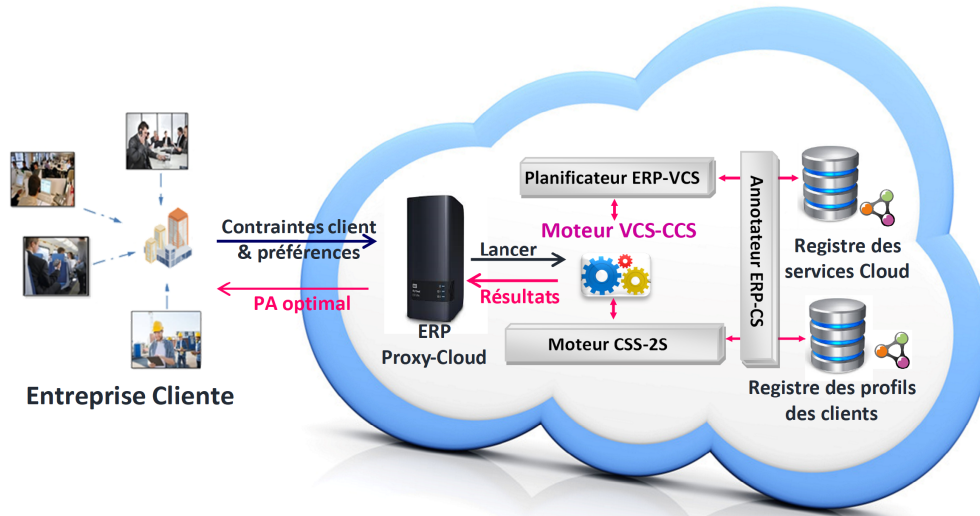


FIGURE 4.1 L'architecture de composition d'ERP sensible au contexte

## 4 Modèle fonctionnel

Lorsqu'une entreprise cliente configure sa session, elle spécifie ses différents besoins fonctionnels et non-fonctionnels. Les besoins fonctionnels comprennent le type de processus d'affaire, les opérations et les entrées/sorties. Les besoins non fonctionnels sont spécifiés visuellement par des contraintes et préférences de QoS. Les QoS peuvent être des QoS d'utilité  $QoS_U$  ou des QoS longueVie ( $QoS_L$ ). Certaines contraintes  $QoS_L$  peuvent être statiques ou dynamiques. Une fois que le proxy Cloud ERP reçoit ces besoins, il lance le planificateur ERP-VCS pour générer le service composite virtuel approprié (VCS : Virtual Business Process) en utilisant des règles sémantiques de l'ontologie (voir figure 4.2).

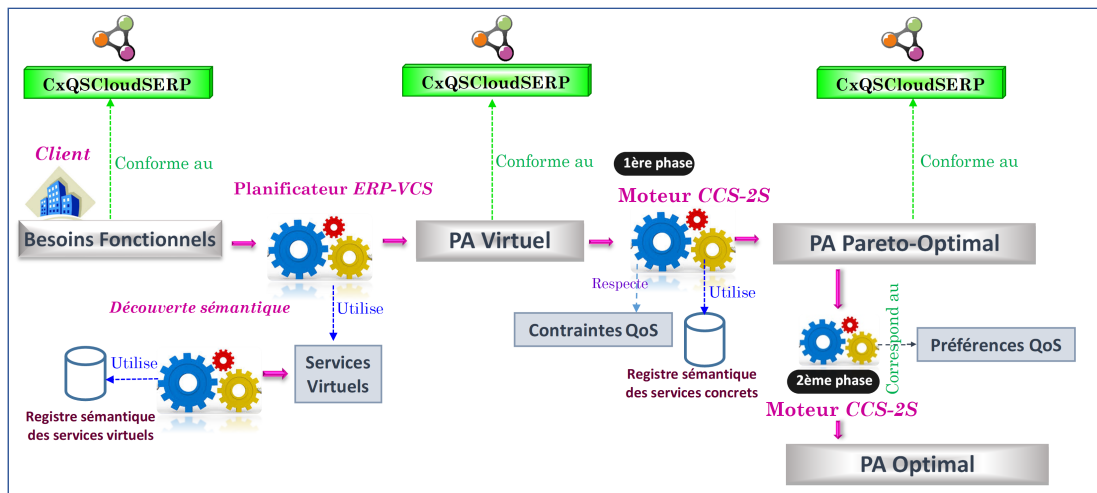


FIGURE 4.2 Le modèle fonctionnel de composition d'ERP sensible au contexte

Étant donné que plusieurs critères de QoS conflictuels doivent être optimisés simultanément, le moteur CCS-2S vise à trouver le CCS optimal qui répond aux contraintes et préférences non-fonctionnelles des clients avec le meilleur compromis de QoS. Par exemple : un client exige d'obtenir un processus d'affaire avec un minimum de prix et un temps d'exécution réduit, donc le moteur doit trouver le meilleur compromis des deux objectifs contradictoires (le temps d'exécution et le prix). En effet, le moteur CCS-2S est un algorithme d'optimisation à deux phases. La première phase utilise l'optimisation multi-objectif pour obtenir un ensemble Pareto-optimal de CCSs pertinents qui respectent les contraintes des clients. L'optimisation est réalisée en utilisant l'algorithme NSGA-II dans lequel nous avons ajouté une nouvelle relation de dominance de Pareto basée sur une nouvelle fonction de pénalité appelée degré de violation de contraintes.

- **Le degré de violation de contraintes :** est une valeur assignée à chaque CCS pour désigner le taux global de violation de contraintes du client. Un degré de violation supérieur à zéro signifie que le CCS correspondant ne respecte pas les contraintes du client et par conséquent, il est pénalisé. Ce degré est calculé en utilisant les équations suivantes :

$$\text{deg}_{v\_CCS} = \sum_i w_i \times \text{deg}_{v\_ccs}^i \quad (4.1)$$

où :

$$\text{deg}_{v\_ccs}^i = \begin{cases} C_i - q_i & \text{If } C_i > q_i \\ 0 & \text{Otherwise} \end{cases} \quad (4.2)$$

- $w_i$  est le poids associé au  $i$ ème attribut de QoS calculé par l'équation 3.4.
- $q_i$  est la valeur du  $i$ ème attribut de QoS.
- $C_i$  est la valeur de la contrainte client du  $i$ ème attribut de QoS.

- **Nouvelle relation de dominance Pareto :** elle est liée au degré de violation des contraintes de QoS. Elle considère que le CCS dominant est celui qui a le plus faible degré de violation de contrainte. Si les degrés de violation de deux CCS sont égaux, la relation de dominance classique de NSGA-II est appliquée. La nouvelle relation de dominance Pareto est formalisée comme suit :

Soit deux services composites concrets différents  $CCS_1$  et  $CCS_2$  caractérisés par un ensemble de valeurs de QoS.  $CCS_1$  domine strictement  $CCS_2$  (notée  $CCS_1 \prec_{cv} CCS_2$ ) si l'une des conditions suivantes est remplie :

1.  $\text{deg}_{v\_CCS_1} < \text{deg}_{v\_CCS_2}$
2.  $\text{deg}_{v\_CCS_1} = \text{deg}_{v\_CCS_2}$  and  $CCS_1 \prec CCS_2$

La deuxième phase sert à sélectionner la solution optimale unique qui répond aux préférences du client à partir de l'ensemble des solutions obtenues du Pareto-optimal. Chaque CCS est évalué par un score en utilisant la pondération additive simple (SAW : Simple Additive Weighting). Ensuite, le CCS avec le meilleur score (maximum) est sélectionné. Nous rappelons que les poids des différents attributs de QoS sont calculés avec une nouvelle technique (voir l'équation 3.4). Cette phase permet d'adapter dynamiquement la prise de décision en cas de changement des préférences de qualité contextuelles du client ou de l'environnement (par exemple : chute de la bande passante) et, par conséquent, de mettre continuellement à jour la solution optimale sans exécuter la première phase.

## 5 Algorithme de composition et d'optimisation sensible au contexte basé sur la sémantique

L'algorithme présenté combine l'algorithme NSGA-II et des règles sémantiques de l'ontologie pour la sélection et la composition des services Cloud pour l'ERP. L'ontologie est utilisée pour fournir un niveau sémantique qui facilite la génération automatique des processus d'affaires virtuels et l'algorithme NSGA-II est adopté pour générer un ensemble optimal de processus d'affaires concrets qui respectent les contraintes du client. L'algorithme NSGA-II a été choisi parmi les différents algorithmes multi-objectif car il est le plus populaire dans la résolution des problèmes d'optimisation multi-objectifs, et il atteint le front de Pareto en moins d'itérations [113].

Le problème d'optimisation multi-objectif prend en compte les considérations suivantes :

- $VBP = \{VS_1, VS_2, \dots, VS_n\}$ , désigne un processus d'affaire virtuel<sup>1</sup> composé de  $n$  services virtuels  $VS_i$ .
- Chaque service virtuel  $VS_i$  est lié à un ensemble de  $m$  services Cloud concrets  $\{cs_{i,1}, cs_{i,2}, \dots, cs_{i,m}\}$  ayant les mêmes fonctionnalités avec différentes valeurs de QS.  $cs_{i,j}$  représente le jème service concret lié au service  $VS_i$ .
- Chaque service concret désigné par  $cs_i$  est une instance d'un service virtuel et il est caractérisé par  $QoS_U$  et  $QoS_L$ . Le  $cs_i$  prend  $cs_i^{in}$  comme données d'entrée afin de produire  $cs_i^{out}$  en tant que données de sortie.
- $QoS_U$  d'un service concret  $cs_i$  est défini comme un vecteur  $QoS_U(cs_i) = \langle qu_{i,1}, qu_{i,2}, \dots, qu_{i,nu} \rangle$ , où  $nu$  représente le nombre d'attributs de  $QoS_U$  requis par le client.

---

1. Un *processus d'affaire virtuel* est un *service composite virtuel*.

- $QoS_L$  d'un service concret  $cs_i$  est défini comme un vecteur  $QoS_L(cs_i) = \langle ql_{i,1}, ql_{i,2}, \dots, ql_{i,nl} \rangle$ , où  $nl$  représente le nombre d'attributs de  $QoS_L$  requis par le client.
- $QoS$  d'un service concret  $cs_i$  est défini comme un vecteur  $QoS(cs_i) = \langle q_{i,1}, q_{i,2}, \dots, q_{i,nb} \rangle$ , où  $nb = nu + nl$  représente le nombre d'attributs de  $QoS$  requis par le client.
- Les vecteurs  $C = \langle C_1, C_2, \dots, C_{nb} \rangle$  et  $Pref = \langle Pref_1, Pref_2, \dots, Pref_{nb} \rangle$  définissent respectivement les contraintes et les préférences quantitatives du client. Les contraintes sémantiques du client exprimées en valeurs qualitatives dans l'ontologie sont transformées en valeurs quantitatives selon le tableau 3.3 et l'équation 3.3.

Notre algorithme contextuel à base d'ontologie comprend le planificateur ERP-VCS et le moteur CCS-2S avec deux phases (voir figure 4.3). Les étapes de cet algorithme sont détaillées ci-dessous.

**Inputs :** — Besoins fonctionnels, contraintes et préférences contextuelles du client.  
 — Registre des services sémantiques concrets et virtuels.

**Output :** Service-composite-concret optimal.

### Etape 1 – Génération sémantique et automatique du processus d'affaire virtuel ERP (VCS-Planner)

Le VCS-planner génère automatiquement le processus d'affaire virtuel en tant qu'une séquence de services virtuels. Ce composant utilise un algorithme de chaînage arrière (research after) et les relations sémantiques (FollowedBy, AlternateWith, ParallelWith et Repeat) entre les services virtuels selon les besoins fonctionnels des clients [13]. L'algorithme de chaînage arrière est basée sur l'idée que : la sortie du service courant correspond à l'entrée du service prochain. Dans le cas où le nombre de services d'affaires virtuels est très important, la vérification de tous les services à chaque étape prend beaucoup de temps. Pour réduire le temps de vérification et limiter l'espace de recherche, la réputation de chaque service virtuel est utilisée en se basant sur les relations sémantiques entre les services virtuels [114]. Le planificateur ERP-VCS effectue la correspondance entre le service virtuel et les besoins fonctionnels du client. Le processus de correspondance sémantique exploite la catégorisation de la hiérarchie de services virtuels pour trouver les correspondances appropriées. En effet, cette correspondance est basée sur les propriétés du service virtuel (catégorie du service, rôle du service, emplacement du service, entrées/sorties du service, etc.).

Chaque besoin fonctionnel peut être exprimé par une clause "AND" d'une expression simple, comme dans l'exemple suivant : (virtualServiceCategory = "Buy" AND Transac-

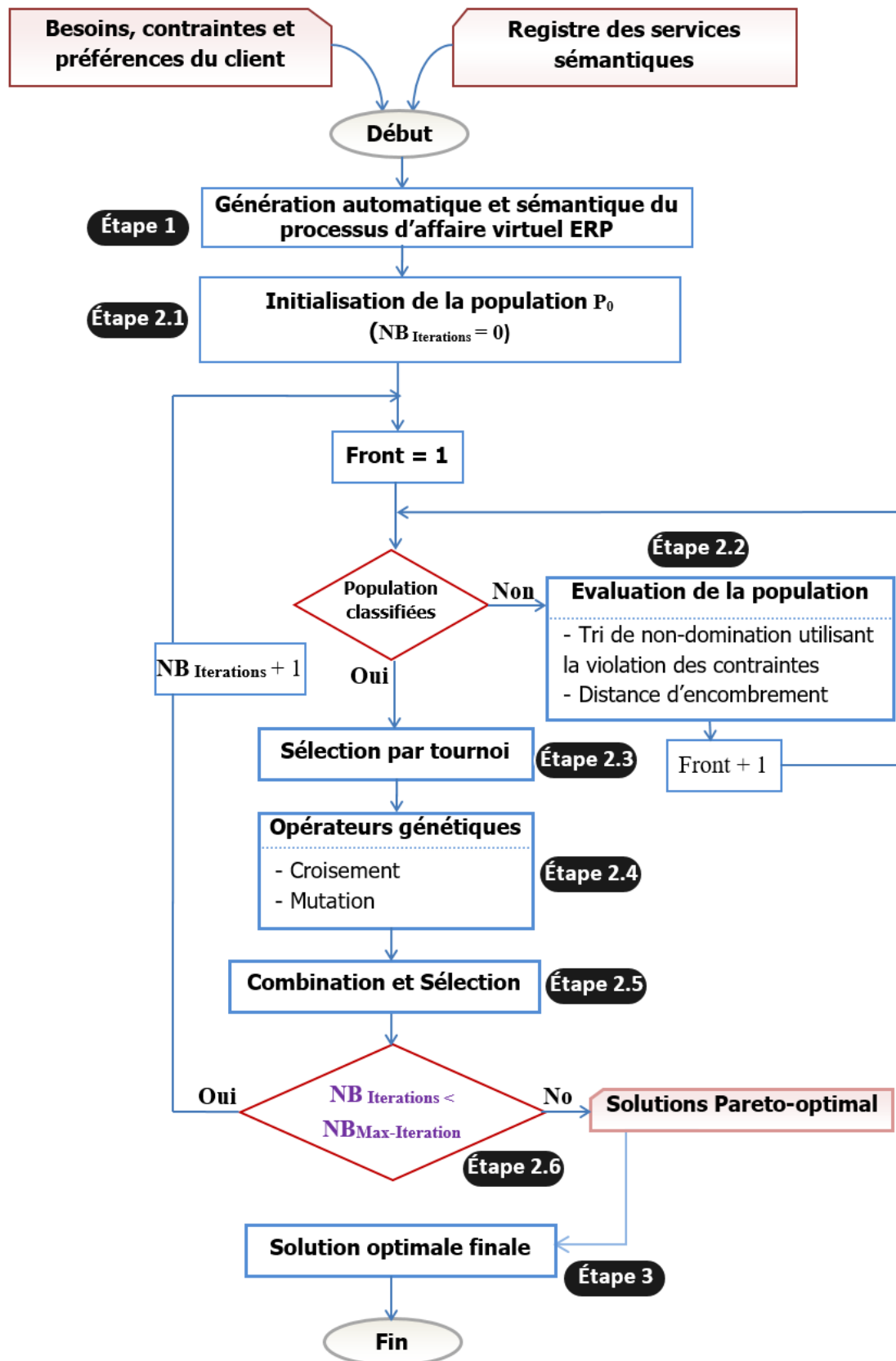


FIGURE 4.3 L'algorithme d'optimisation CCS-2S

tionType = "Internet" **AND** virtualServiceInput = "ProductDetails" **AND** virtualServiceInput = "CustomerDetails" **AND** virtualServiceOutput = "Receipt"). Le moteur d'inférence génère un processus d'affaire virtuel en tant que composition de services virtuels.

### Etape 2 (Phase 1 de CCS-2S) – Optimisation multi-objectif

Cette phase fournit un ensemble de solutions qui respectent les contraintes du client en utilisant l'algorithme d'optimisation multi-objectif NSGA-II avec une nouvelle relation de dominance Pareto. L'optimisation multi-objectif identifie un ensemble de CCS Pareto-optimal notés par  $P_{CCS} = \{CCS_1, CCS_2, \dots\}$  de  $S$  tel que :

$$\forall CCS_i \in S, \forall CCS_j \in P_{CCS} : \neg(CCS_i \prec_{cv} CCS_j) \quad (4.3)$$

Où  $S$  est l'espace d'exploration (tous les CCS possibles).

**Inputs :** — Contraintes et préférences du client.

— Registre des services sémantiques concrets et virtuels.

**Output :** L'ensemble de solutions Pareto-optimal.

#### Etape 2.1 – Initialisation de la population

Dans cette étape, la population initiale est générée aléatoirement. Elle est constituée de  $N$  individus. Chaque individu représente une solution du processus de composition des services Cloud concrets. L'individu (CCS) est codé par un tableau d'entier. L'entier ( $j : 1 \dots m$ ) dans le tableau représente le service concret sélectionné ( $CS_j$ ) à partir de l'ensemble des services concrets liés au service virtuel ( $VS_i i : 1 \dots n$ ) (voir figure 4.4).



FIGURE 4.4 Codage de la solution

#### Etape 2.2 – Évaluation

Pour sélectionner les meilleurs candidats CCSs, chaque individu est évalué en calculant ses valeurs de fonction objective pour chaque attribut QoS conformément au Tableau 3.4.

---

 Algorithme 1 : nouvelle relation de domination entre deux CCSs
 

---

**1 : Début****2 :** Entrées deux CCSs : p et q**3 :** Calcule du degré de violation de contraintes pour chaque CCS en appliquant Eq. 4.1*/\* vérifier le degré de violation de contraintes de p et q \*/***4 :** **if**  $deg_{v_p} < deg_{v_q}$  **then****5 :**   return true**6 :** **elseif**  $deg_{v_p} = deg_{v_q}$  **then***/\* une des fonctions objectives  $F_i$ ,  $F_i(p) < F_i(q)$  ou pour toute fonction objective  $F$ ,  $F_i(p) = F_i(q)$  \*/***7 :**   **For** chaque fonction objective  $F_i \in$  **do****8 :**     **if**  $F_i(p) < F_i(q)$  **then****9 :**       return false ;**10 :**    **if**  $F_i(p) = F_i(q)$  **then****11 :**        $N_{F\_equal} = F\_equal + 1$  ;**12 :**    **End For****13 :** **if**  $N_{F\_equal} = \text{size}(F)$  **then****14 :**    return false ;**15 :** **else****16 :**    return true ;**17 :** **end if****18 :** **else** return false**19 :** **end if****20 :** **End**


---

Ensuite, chaque valeur de fonction objective est normalisée en utilisant les équations 3.5 et 3.6.

- **Tri de non-domination**<sup>2</sup> **utilisant le principe de violations de contraintes** : tous les individus non-dominés sont placés dans le premier front (rang 1). Ces individus devraient être substitués pour atteindre le front suivant jusqu'à ce que tous les prochains fronts soient classés. La nouvelle relation de dominance entre deux individus est décrite en détails dans (Algorithme 1).
- **Distance d'encombrement (Crowding distance)** : est la distance moyenne entre les individus voisins des deux côtés pour chaque fonction objective.

---

2. La phrase "Tri non-domination" est la traduction de l'expression célèbre "non-dominated sorting".

### Étape 2.3 – Sélection

Une fois que les individus sont triés sur la base du classement de non-domination, une sélection par tournoi est effectuée. En effet, le processus de sélection prend aléatoirement deux candidats  $CCS_1$  et  $CCS_2$  de la population, ensuite il sélectionne le CCS dont le rang est le plus bas. Dans le cas où les rangs des deux CCSs sont égaux, le CCS avec la plus grande distance d'encombrement est sélectionné. Notant que ce processus a un grand impact sur la convergence et la diversité des solutions.

### Étape 2.4 – Les opérateurs génétiques

Dans cette étape, les opérateurs génétiques (croisement et mutation) sont appliqués pour produire de nouveaux individus à la génération suivante. L'opérateur de croisement consiste à combiner deux individus (c.-à-d. CCS) fournis par la sélection donnée à l'étape 2.3. Une position de découpage est choisie aléatoirement chez les deux individus. Après cela, les gènes provenant de la position du découpage sont échangés entre les deux individus (figure 4.5a). L'opération de croisement est répétée  $N/2$  fois pour obtenir  $N$  enfants. Afin d'augmenter la diversité de la population générée et d'éviter l'optimum local, un opérateur de mutation est effectué sur certains individus (c.-à-d. CCS) en fonction de la probabilité de mutation. En fait, un élément (gène) choisi aléatoirement de chaque individu concerné est muté (figure 4.5b).

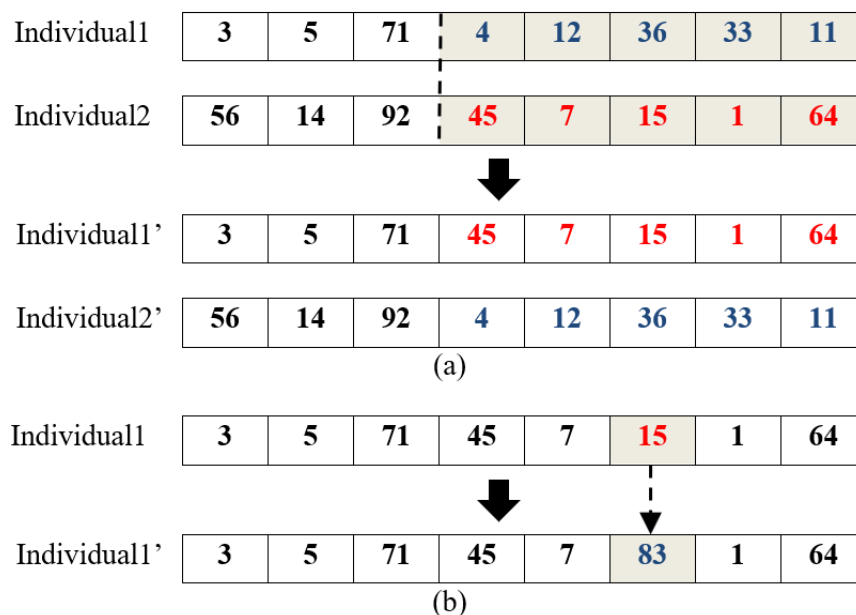


FIGURE 4.5 Les opérateurs génétiques. (a) Croisement, (b) Mutation.

**Etape 2.5 – La recombinaison et la sélection**

Les individus générés à partir des opérations de croisement et de mutation sont ajoutés à la population actuelle. La population est triée sur la base du principe de non-domination. La nouvelle génération est remplie par les individus de chaque front jusqu'à ce que le nombre d'individus dépasse  $N$ . Si en additionnant tous les individus d'un front, la population dépasse  $N$ , alors les individus sont sélectionnés de ce front par ordre décroissant en fonction de la distance d'encombrement (Crowding Distance) jusqu'à la taille de la population atteint  $N$ .

**Etape 2.6 – Le critère d'arrêt**

Le cycle est répété jusqu'à ce que le nombre de générations atteigne le maximum.

**Etape 3 (phase 2) – L'unique solution optimale**

Cette étape sélectionne une solution unique (CCS) à partir de l'ensemble de solutions Pareto-optimal ( $P_{CCS}$ ) obtenue de la première phase. Ensuite, le score de chaque solution est calculé par l'équation 4.4. Finalement, la solution avec le meilleur score (max) est sélectionnée.

**Inputs :** — Contraintes et préférences du client.

— L'ensemble de solutions Pareto-optimal.

**Output :** Le processus d'affaire optimal (le meilleur CCS).

La formulation de la tâche de cette phase est la suivante :

Trouver  $Max\{Score(CCS_i)\}$  tel que  $i \in [1, |P_{CCS}|]$  où

$$Score(CCS) = \sum_{j=1}^{nb} q'_j \times w_j \quad (4.4)$$

—  $|P_{CCS}|$  est le nombre de solutions Pareto-optimal.

—  $nb$  est le nombre des attributs de QoS.

—  $q'_j$  est la valeur normalisée de la QoS  $j$ .

—  $w_j$  est le poids du jème attribut de QoS calculé par l'équation 3.4.

Le moteur CCS-2S évalue en permanence le profil du client (c.-à-d. les préférences et contraintes du client) pour fournir une solution CCS en conséquence. En cas de modification des préférences du client, la solution est sélectionnée directement à partir des solutions Pareto-optimales courantes sans avoir à exécuter la première phase.

## 6 Application de l'algorithme proposé à l'étude de cas

Pour clarifier le déroulement de notre algorithme proposé, nous illustrons son application à l'étude de cas présentée dans la section 2.1. Le client spécifie ses besoins fonctionnels et non-fonctionnels comme suit :

- Les besoins fonctionnels du client :
  - **rôle** = 'Vente'.
  - **catégorie** = 'Vente de produits'.
  - **type de transaction** = 'internet'.
  - **entrées** = ['information client', 'désignation produit', 'type de paiement'].
  - **sorties** = [ticket de réception et d'achat].
- Les contraintes de QoS du client :

Le client spécifie des contraintes pour les deux QoS d'utilité (prix et temps d'exécution) et il ignore la spécification des autres QoS d'utilité (disponibilité, fiabilité, sécurité) et les QoS longue vie (débit et confiance).

  - **Prix** = 'moins cher' qui se traduit par notre ontologie en 'High'.
  - **Temps d'exécution** = 'rapide' qui se traduit par notre ontologie en 'High'.
- Les préférences de QoS du client :
  - le client préfère le **Prix** par rapport au **Temps d'exécution** ( $\text{Prix} \gg \text{Temps d'exécution}$ ).

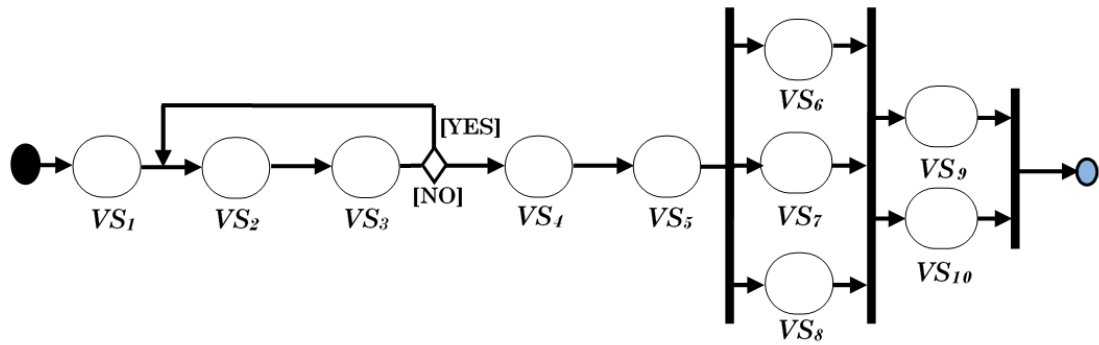
### 6.1 Génération d'un processus d'affaire virtuel (VBP) optimal

Dans cette étape l'algorithme prend comme entrée les besoins fonctionnels du client pour générer un processus d'affaire virtuel (VBP) en tant que sortie. Ce VBP est optimisé par l'élimination de la redondance et la sélection des services selon leur réputation. La figure 4.6 montre le VPB généré à partir des besoins fonctionnels spécifiés par le client.

### 6.2 Génération des solutions Pareto-optimal

La première phase de l'algorithme CCS-2S est appliquée sur notre étude pour générer un ensemble de solutions Pareto-optimal qui respectent les contraintes du client. Les paramètres de l'algorithme sont configurés comme suit :

- La taille de la population = 30.



### List of virtual services

*VS<sub>1</sub>: EnterProductDetails*

*VS<sub>2</sub>: BrowseProductsDetails*

*VS<sub>3</sub>: SelectDesiredProduct*

*VS<sub>4</sub>: CustomerDetails*

*VS<sub>5</sub>: ProductDetails*

*VS<sub>6</sub>: Customer Location*

*VS<sub>7</sub>: Customer Device Context*

*VS<sub>8</sub>: Customer Preferences*

*VS<sub>9</sub>: Update Credit*

*VS<sub>10</sub>: Receive Products*

FIGURE 4.6 Processus d'affaire virtuel optimal de la commande de produits en ligne

- La probabilité de mutation = 40%.
- Le nombre maximum d'itération = 500.

Le tableau 4.1 décrit les valeurs des attributs de QoS de chaque service concret associé à un service virtuel.

Le nombre de combinaisons possibles =  $10^5$ . L'espace de recherche dans notre étude de cas est réduit pour montrer le fonctionnement de notre approche et sa fiabilité.

#### 1. Initialisation de la population

La population initiale est générée aléatoirement. Elle contient 30 individus, et chaque individu représente un CCS. Le CCS est composé de 10 CS dont chaque CS est associé à un VS. La figure 4.7 représente la population initiale générée par CCS-2S.

#### 2. Evaluation et sélection

Avant la sélection, l'algorithme CCS-2S évalue chaque individu de la population. Les fonctions objectives de chaque attribut de QoS d'un individu sont calculées selon le tableau 3.4 et l'équation 3.6. Le degré de violation est calculé par l'équation 4.1. En appliquant cette évaluation sur le premier individu de population présenté dans la figure 4.7, on obtient les valeurs suivantes :

- Les valeurs des objectives : Temps d'exécution = 0.6168 et le Prix = 0.4140
- degré de violation = 0.2819

TABLE 4.1 Les valeurs des attributs de QoS de chaque service concret

| S.Virtuel | S.Concret | Temps | Prix | Fiabilité | Disponibilité | Sécurité | Confidence | Débit |
|-----------|-----------|-------|------|-----------|---------------|----------|------------|-------|
| SV1       | sc1;1     | 0,9   | 0,54 | 0,6       | 0,69          | 0        | 0,55       | 0,65  |
|           | sc1;2     | 0,6   | 0,2  | 0,35      | 0,14          | 0,33     | 0,69       | 0,36  |
|           | sc1;3     | 0,72  | 0,6  | 0,62      | 0,8           | 1        | 0,7        | 0,57  |
|           | sc1;4     | 0,75  | 0,4  | 0,47      | 0,35          | 0,66     | 0,6        | 0,85  |
|           | sc1;5     | 0,6   | 0,6  | 0,29      | 0,5           | 0        | 0,5        | 0,64  |
| SV2       | sc2;1     | 0,8   | 0,35 | 0,85      | 0,24          | 0,66     | 0,68       | 0,85  |
|           | sc2;2     | 0,63  | 0,52 | 0,45      | 0,58          | 0,33     | 0,71       | 0,39  |
|           | sc2;3     | 0,7   | 0,14 | 0,67      | 0,2           | 0,33     | 0,47       | 0,52  |
|           | sc2;4     | 0,24  | 0,4  | 0,7       | 0,36          | 0,33     | 0,49       | 0,25  |
|           | sc2;5     | 0,82  | 0,72 | 0,5       | 0,68          | 0,33     | 0,8        | 0,77  |
| SV3       | sc3;1     | 0,54  | 0,26 | 0,41      | 0,3           | 0,66     | 0,3        | 0,26  |
|           | sc3;2     | 0,68  | 0,63 | 0,19      | 0,51          | 0,33     | 0,48       | 0,42  |
|           | sc3;3     | 0,86  | 0,45 | 0,59      | 0,25          | 0,33     | 0,9        | 0,59  |
|           | sc3;4     | 0,75  | 0,4  | 0,47      | 0,35          | 0,66     | 0,6        | 0,85  |
|           | sc3;5     | 0,4   | 0,8  | 0,63      | 0,4           | 0        | 0,74       | 0,27  |
| SV4       | sc4;1     | 0,7   | 0,27 | 0,23      | 0,6           | 1        | 0,67       | 0,52  |
|           | sc4;2     | 0,2   | 0,84 | 0,89      | 0,8           | 0,33     | 0,4        | 0,68  |
|           | sc4;3     | 0,88  | 0,59 | 0,78      | 0,9           | 0,33     | 0,86       | 0,45  |
|           | sc4;4     | 0,49  | 0,8  | 0,8       | 0,7           | 0,66     | 0,52       | 0,85  |
|           | Sc4;5     | 0,54  | 0,6  | 0,24      | 0,8           | 0,66     | 0,75       | 0,32  |
| SV5       | sc5;1     | 0,26  | 0,68 | 0,65      | 0,4           | 0,66     | 0,3        | 0,5   |
|           | sc5;2     | 0,37  | 0,85 | 0,27      | 0,1           | 1        | 0,27       | 0,41  |
|           | sc5;3     | 0,5   | 0,3  | 0,8       | 0,8           | 1        | 0,74       | 0,8   |
|           | sc5;4     | 0,8   | 0,4  | 0,5       | 0,65          | 0        | 0,76       | 0,3   |
|           | sc5;5     | 0,69  | 0,6  | 0,45      | 0,24          | 0,66     | 0,49       | 0,9   |
| SV6       | sc6;1     | 0,54  | 0,65 | 0,39      | 0,5           | 0,33     | 0,43       | 0,58  |
|           | sc6;2     | 0,27  | 0,47 | 0,7       | 0,89          | 0,66     | 0,24       | 0,2   |
|           | sc6;3     | 0,56  | 0,58 | 0,61      | 0,26          | 0,66     | 0,6        | 0,41  |
|           | sc6;4     | 0,85  | 0,3  | 0,8       | 0,47          | 0        | 0,63       | 0,72  |
|           | sc6;5     | 0,23  | 0,75 | 0,26      | 0,39          | 0        | 0,6        | 0,69  |
| SV7       | sc7;1     | 0,7   | 0,47 | 0,35      | 0,4           | 0,66     | 0,8        | 0,32  |
|           | sc7;2     | 0,3   | 0,28 | 0,25      | 0,28          | 0        | 0,27       | 0,4   |
|           | sc7;3     | 0,45  | 0,8  | 0,54      | 0,69          | 1        | 0,36       | 0,5   |
|           | sc7;4     | 0,49  | 0,6  | 0,75      | 0,45          | 1        | 0,5        | 0,63  |
|           | sc7;5     | 0,88  | 0,59 | 0,78      | 0,9           | 0,33     | 0,86       | 0,45  |
| SV8       | sc8;1     | 0,2   | 0,43 | 0,83      | 0,23          | 0        | 0,3        | 0,3   |
|           | sc8;2     | 0,82  | 0,72 | 0,5       | 0,68          | 0,33     | 0,8        | 0,77  |
|           | sc8;3     | 0,54  | 0,6  | 0,24      | 0,8           | 0,66     | 0,75       | 0,32  |
|           | sc8;4     | 0,49  | 0,8  | 0,8       | 0,7           | 0,66     | 0,52       | 0,85  |
|           | sc8;5     | 0,24  | 0,4  | 0,7       | 0,36          | 0,33     | 0,49       | 0,25  |
| SV9       | sc9;1     | 0,2   | 0,43 | 0,83      | 0,23          | 0        | 0,3        | 0,3   |
|           | sc9;2     | 0,82  | 0,72 | 0,5       | 0,68          | 0,33     | 0,8        | 0,77  |
|           | sc9;1     | 0,49  | 0,6  | 0,75      | 0,45          | 1        | 0,5        | 0,63  |
|           | sc9;4     | 0,8   | 0,4  | 0,5       | 0,65          | 0        | 0,76       | 0,3   |
|           | sc9;5     | 0,3   | 0,28 | 0,25      | 0,28          | 0        | 0,27       | 0,4   |
| SV10      | sc10;1    | 0,9   | 0,54 | 0,6       | 0,69          | 0        | 0,55       | 0,65  |
|           | sc10;2    | 0,63  | 0,52 | 0,45      | 0,58          | 0,33     | 0,71       | 0,39  |
|           | sc10;3    | 0,88  | 0,59 | 0,78      | 0,9           | 0,33     | 0,86       | 0,45  |
|           | sc10;4    | 0,85  | 0,3  | 0,8       | 0,47          | 0        | 0,63       | 0,72  |
|           | Sc10;5    | 0,88  | 0,59 | 0,78      | 0,9           | 0,33     | 0,86       | 0,45  |

|            |            |            |            |            |            |            |            |            |             |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| $CS_{1,5}$ | $CS_{2,4}$ | $CS_{3,3}$ | $CS_{4,2}$ | $CS_{5,2}$ | $CS_{6,5}$ | $CS_{7,2}$ | $CS_{8,4}$ | $CS_{9,4}$ | $CS_{10,1}$ |
| $CS_{1,5}$ | $CS_{2,4}$ | $CS_{3,5}$ | $CS_{4,1}$ | $CS_{5,1}$ | $CS_{6,2}$ | $CS_{7,4}$ | $CS_{8,5}$ | $CS_{9,2}$ | $CS_{10,2}$ |
| ⋮          |            |            |            |            |            |            |            |            |             |
| $CS_{1,3}$ | $CS_{2,2}$ | $CS_{3,1}$ | $CS_{4,4}$ | $CS_{5,2}$ | $CS_{6,5}$ | $CS_{7,4}$ | $CS_{8,3}$ | $CS_{9,3}$ | $CS_{10,5}$ |

FIGURE 4.7 Population initiale

La sélection est effectuée après le tri de la population selon la relation de dominance Pareto et la distance d'encombrement. Par exemple, le CCS présenté dans la figure 4.8 a le rang = 24 et la distance\_d'encombrement = infini.

|            |            |            |            |            |            |            |            |            |             |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| $CS_{1,1}$ | $CS_{2,2}$ | $CS_{3,4}$ | $CS_{4,4}$ | $CS_{5,2}$ | $CS_{6,4}$ | $CS_{7,2}$ | $CS_{8,4}$ | $CS_{9,2}$ | $CS_{10,1}$ |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|

FIGURE 4.8 Individu de la population

### 3. Croisement et mutation

Les opérations génétiques sont appliquées sur la population pour engendrer la nouvelle génération. La figure 4.9 représente le croisement de deux CCSs. Notant que le point de croisement est généré aléatoirement. La figure 4.10 représente la mutation appliquée sur un CCS de la population. 40% des individus sont mutés et la position de mutation est sélectionnée de façon aléatoire.

### 4. Recombinaison et sélection

Après l'application des opérations génétiques, nous obtenons 72 individus ; 30 individus de la population actuelle, 30 individus générés par l'opération de croisement et 12 individus générés par l'opération de mutation (le taux de mutation = 40%). La population est triée selon le rang et la distance d'encombrement des individus. La nouvelle population est constituée des 30 meilleurs individus parmi les 72 (voir figure 4.11).

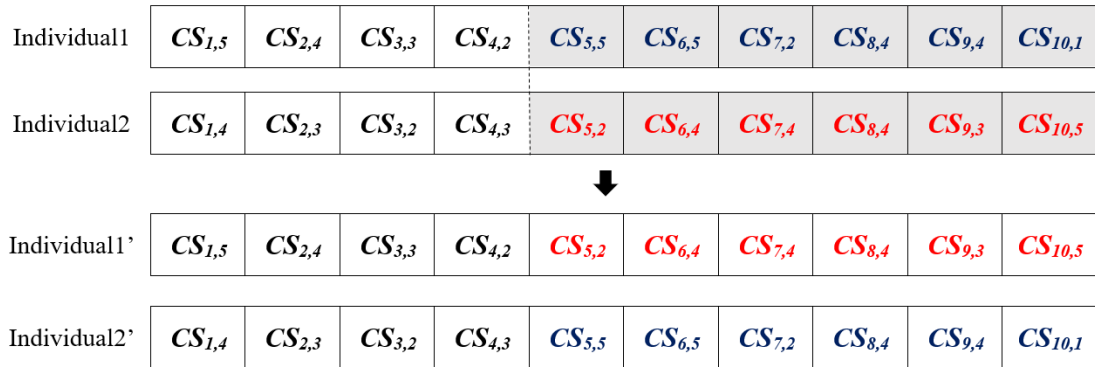


FIGURE 4.9 Croisement

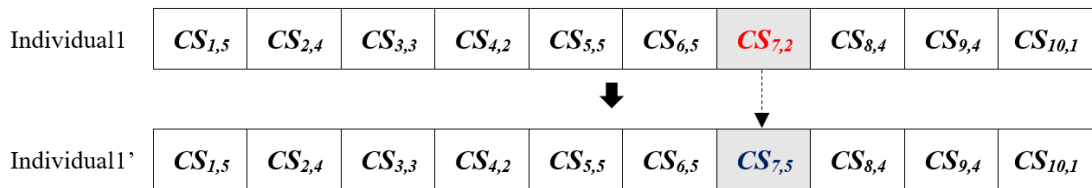


FIGURE 4.10 Mutation

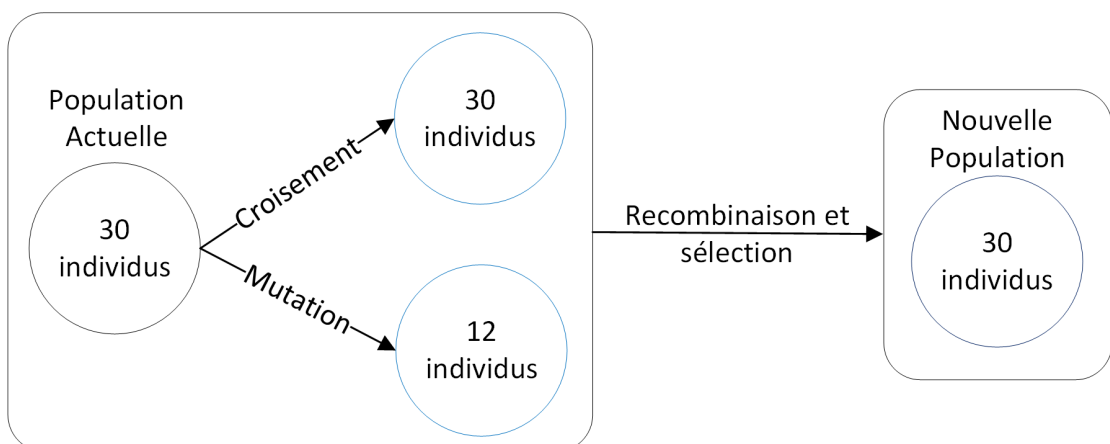


FIGURE 4.11 Recombinaison et sélection

### 5. Pareto optimal

Pour obtenir le Pareto-optimal, les étapes précédentes sont répétées jusqu'à ce que le nombre d'itérations atteigne le `max_iterations`. L'ensemble des solutions du front Pareto-optimal de notre étude de cas après 500 itérations contient 13 individus comme illustré dans la figure 4.12.

Les valeurs des attributs (temps d'exécution et prix) des individus du front Pareto-optimal sont illustrées par le graphe de la figure 4.13.

## 6.3 Sélection de la solution finale

La solution finale est sélectionnée par la deuxième phase de l'algorithme CCS-2S à partir des solutions obtenues de la première phase. En effet, chaque solution est évaluée en calculant son score selon l'équation 4.4, ensuite la solution ayant le meilleur score est sélectionnée. La solution finale de notre étude de cas est le CCS représenté dans la figure 4.14 dont les valeurs des attributs de QoS sont montrées par le petit cercle rouge de la figure 4.15.

où :

- Temps d'exécution = 0.7282, et son poids  $W_{Time} = 0.0196$
- Prix = 0.9315, son poids  $W_{Price} = 0.9803$
- Score = 0.9275

## 7 Nouvelle extension : CxQSCloudERP à base d'agents

Dans cette section, nous présentons une extension de notre approche, notée SCwA-CCS-ERP (Semantic-Based Context-aWare Agents for the Composition of Cloud Services for ERP), en utilisant les agents pour améliorer la performance de notre système [115]. En effet, pour réduire le temps d'exécution de l'optimisation de la composition des services Cloud, le SCwA-CCS-ERP bénéficie de la caractéristique collaborative et parallèle de l'approche multi-agent afin de distribuer l'espace de recherche des services Cloud aux différents agents. En outre, l'autonomie des agents permet d'affecter les tâches du processus de composition aux différents agents pour être exécutées indépendamment.

On peut utiliser différentes stratégies de coopération des agents pour optimiser le processus de sélection et de composition sémantique et dynamique des services Cloud. Parmi ces stratégies nous citons :

### 1. Stratégie maître-esclave

Une stratégie collaborative multi-agent appelée maître-esclave est utilisée pour trouver

|             |            |            |             |            |            |             |            |            |             |
|-------------|------------|------------|-------------|------------|------------|-------------|------------|------------|-------------|
| $CS_{1,3}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,1}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,3}$ | $CS_{10,3}$ |
| $CS_{1,2}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,4}$ | $CS_{7,1}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,3}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,4}$ | $CS_{7,1}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,2}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,1}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,23}$ | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,1}$  | $CS_{8,4}$ | $CS_{9,3}$ | $CS_{10,3}$ |
| $CS_{1,3}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,1}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,2}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,3}$ | $CS_{10,3}$ |
| $CS_{1,3}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,1}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,2}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,2}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,1}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,3}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,1}$ | $CS_{10,2}$ |
| $CS_{1,2}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,21}$ | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,13}$ | $CS_{8,4}$ | $CS_{9,3}$ | $CS_{10,3}$ |
| $CS_{1,3}$  | $CS_{2,5}$ | $CS_{3,4}$ | $CS_{4,2}$  | $CS_{5,3}$ | $CS_{6,2}$ | $CS_{7,3}$  | $CS_{8,4}$ | $CS_{9,3}$ | $CS_{10,3}$ |

FIGURE 4.12 L'ensemble des individus du front Pareto-optimal

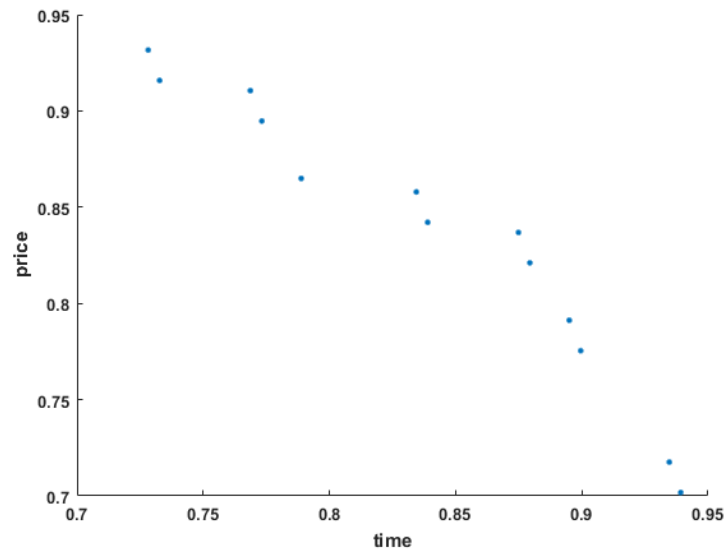


FIGURE 4.13 Les valeurs des attributs du front Pareto-optimal

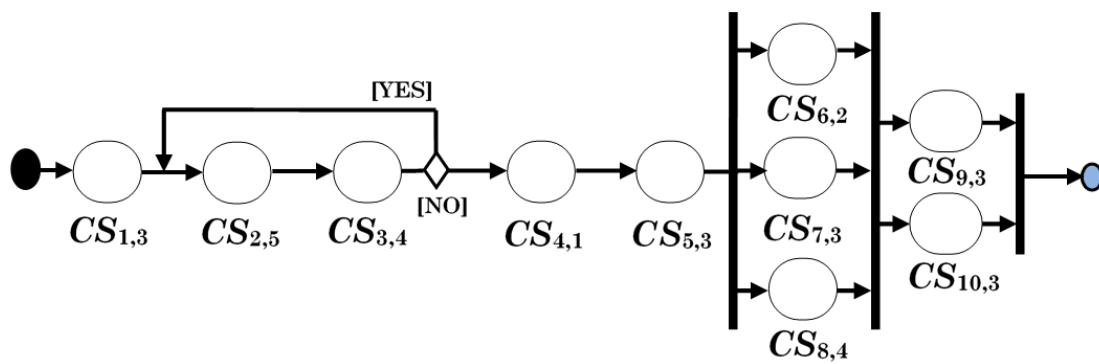


FIGURE 4.14 La solution finale

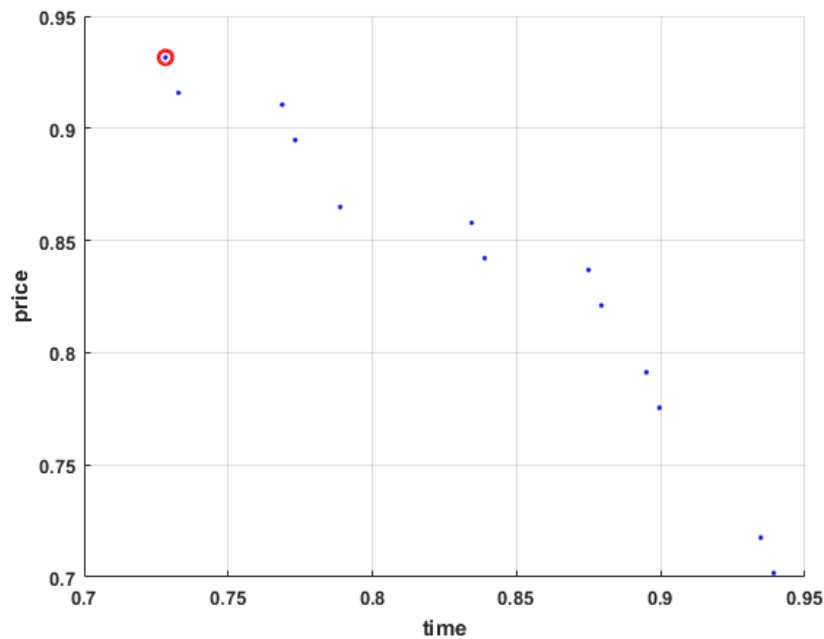


FIGURE 4.15 Les valeurs des attributs de QoS de la solution finale

une solution optimale. Un agent joue le rôle de maître et les autres sont des esclaves. Les agents esclaves évoluent en parallèle pour atteindre leurs meilleures solutions locales, qui sont envoyées à l'agent maître. Ce dernier détermine la solution optimale globale parmi toutes les meilleures solutions locales transmises par les agents esclaves. Cette stratégie est illustrée dans la figure 4.16.

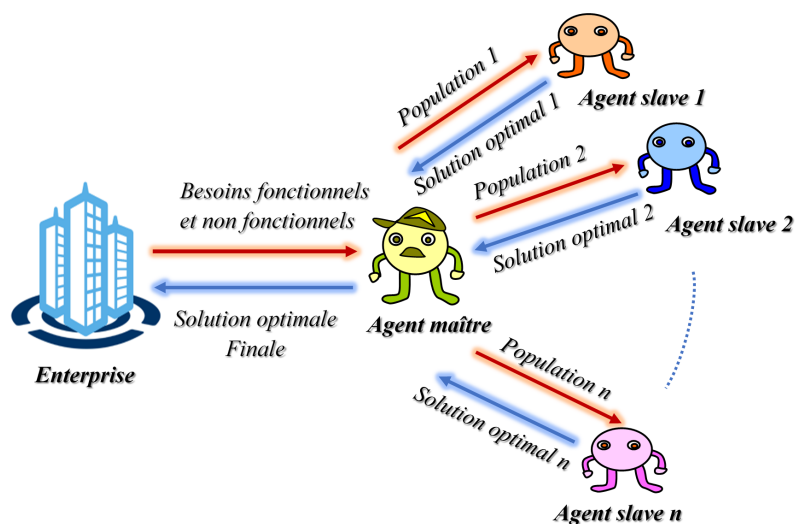


FIGURE 4.16 Stratégie maître-esclave pour SCwA-CCS-ERP

## 2. Stratégie en anneau

La stratégie coopérative en anneau est l'une des stratégies les plus connues. Dans le cadre de cette stratégie, un certain nombre de sous-populations évoluent en parallèle pour trouver la solution optimale. En outre, chaque agent échange sa meilleure solution (Pareto-optimale) avec son voisin si la qualité de sa solution ne s'améliore pas après un certain nombre d'itérations. Cette communication circulaire est effectuée de manière orientée comme illustré dans la figure 4.17.

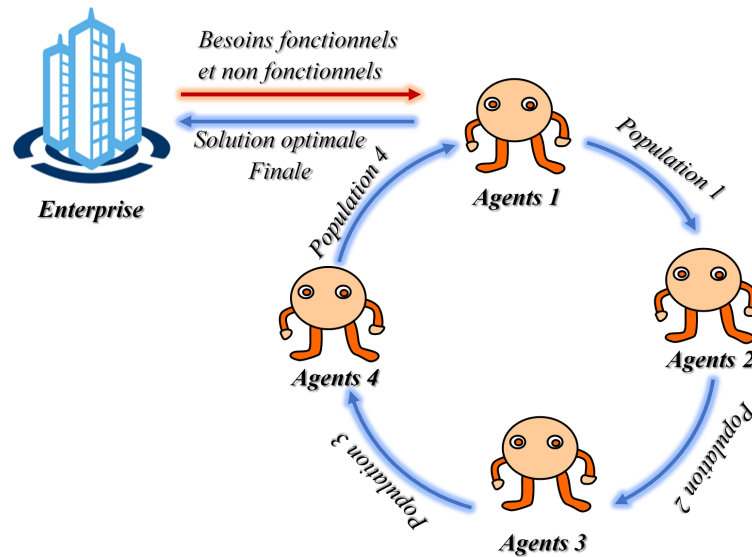


FIGURE 4.17 Stratégie en anneau pour SCwA-CCS-ERP

## 3. Stratégie hybride : maître-esclave et anneau

Une stratégie hybride est une combinaison des deux stratégies : stratégie maître-esclave et stratégie en anneau. L'idée principale de cette hybridation est d'intégrer à la stratégie maître-esclave une coopération en anneau entre les populations esclaves. La figure 4.18 illustre la stratégie hybride. Dans cette stratégie, une copie de la solution optimale de chaque esclave est transmise à l'agent maître. Chaque esclave partage sa solution optimale avec son voisin. Enfin, la solution globale est renvoyée par l'agent maître.

Dans notre travail nous avons choisi d'utiliser la stratégie maître-esclave à cause de sa simplicité.

## 7.1 Architecture et principe de fonctionnement

L'architecture de l'approche SCwA-CCS-ERP contient les mêmes composants principaux que l'approche présentée précédemment, à savoir l'entreprise (client), le proxy de

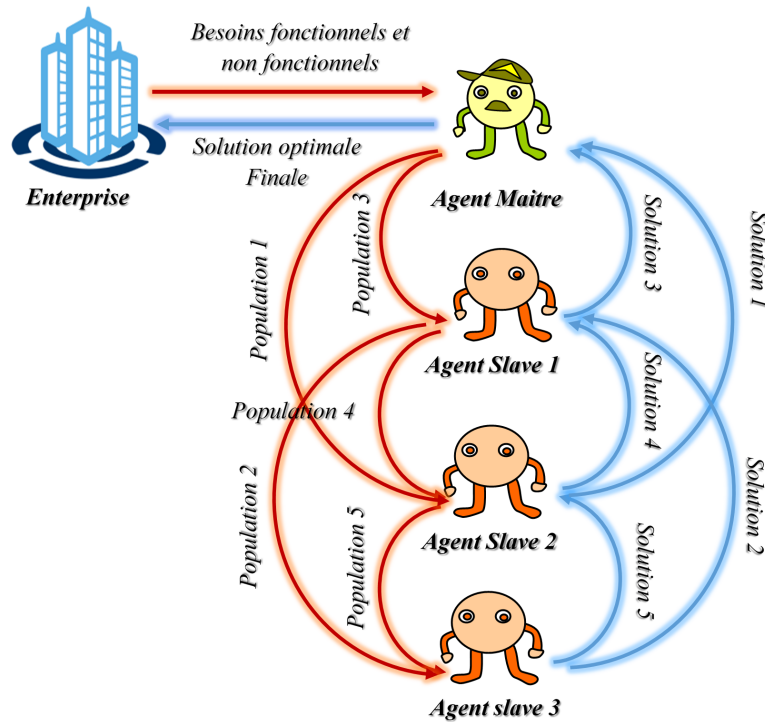


FIGURE 4.18 Stratégie hybride pour SCwA-CCS-ERP

l'ERP Cloud et le registre des services Cloud et profils clients. La différence entre les deux approches réside dans le composant proxy où l'approche SCwA-CCS-ERP utilise l'aspect multi-agent. En effet, l'approche SCwA-CCS-ERP se base sur les agents durant la découverte, la sélection et la composition dynamique sémantique des services Cloud pour l'obtention d'un processus d'affaire ERP personnalisé selon les besoins fonctionnels et non-fonctionnels du client. La figure 4.19 montre une vue globale de l'architecture de l'approche SCwA-CCS-ERP. Le composant proxy de l'approche SCwA-CCS-ERP est décrit comme suit :

**Proxy multi-agent de l'ERP Cloud :** est constitué d'un agent planificateur (ERP-VCS : ERP Virtual Composite Service), d'un agent maître et des agents esclaves d'optimisation de service composite (SCwA-CCS-2S) et d'un gestionnaire du contexte.

- ✓ Agent planificateur ERP-VCS : est un moteur d'inférence sémantique chargé de générer un plan global de composition d'une séquence de services virtuels et permet de produire un processus d'affaire virtuel à mettre en œuvre selon les besoins fonctionnels des clients.
- ✓ Agent SCwA-CCS-2S : prend en entrée les contraintes et les préférences du client décrites dans notre ontologie et la description des services Cloud, afin de produire un

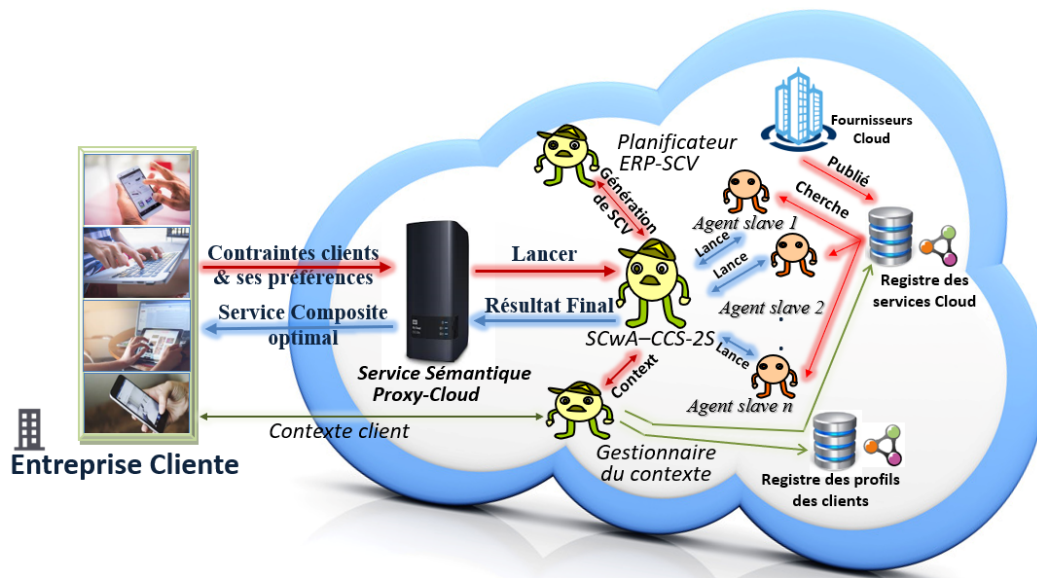


FIGURE 4.19 L'architecture générale de SCwA-CCS-ERP

processus d'affaire optimal sous forme d'un service Cloud composite en fonction des contraintes et préférences du client.

- ✓ Agent de gestion sémantique du contexte : le composant de gestion sémantique évalue dynamiquement le profil utilisateur (besoins, contraintes, préférences et contexte d'utilisation) et le contexte du service Cloud (connexion réseau, taux de charge des serveurs Cloud, fournisseur de services disponible, etc.) pour sélectionner les services Cloud convenables à l'utilisateur.

Le principe de fonctionnement de SCwA-CCS-ERP est illustré dans la figure 4.20.

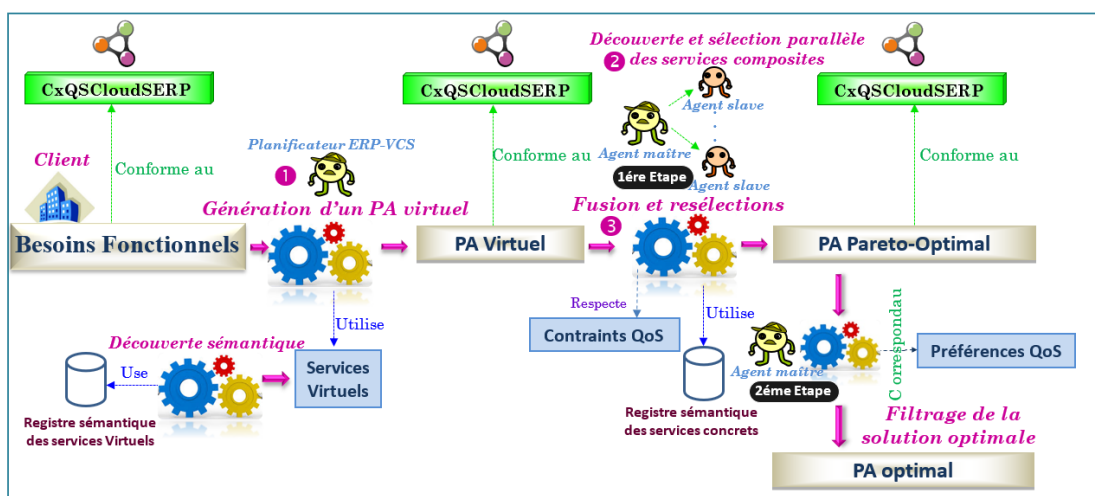


FIGURE 4.20 Modèle fonctionnel de l'approche SCwA-CCS-ERP

Le composant **Proxy-Cloud multi-agent d'ERP** reçoit le profil, il fait appel au composant analyseur de profil sémantique pour traduire les contraintes et les préférences qualitatives vers une représentation quantitative. L'agent générateur sémantique du processus d'affaire virtuel génère un service composite virtuel ERP qui satisfait les besoins fonctionnels du client en basant sur les règles d'inférence de l'ontologie et les informations sémantiques (rôle de service, catégorie, entrées/sorties, QoS, etc.). Ensuite, le Proxy-Cloud lance l'agent principal de l'algorithme SCwA-CCS-2S pour sélectionner, composer et optimiser le service Cloud composite en termes de QoS selon les contraintes et préférences du client. Le SCwA-CCS-2S est un algorithme à deux phases à base d'agents. La première phase consiste à engendrer un ensemble de CCSs optimaux (Pareto-optimal) qui respectent les contraintes du client. Cette phase utilise l'algorithme NSGA-II en introduisant un nouveau critère de dominance basé sur la violation des contraintes. Pour assurer les performances d'exécutions et augmenter la qualité des résultats de composition des services Cloud, l'approche utilise la stratégie collaborative multi-agent. On distingue deux catégories d'agents : **les agents esclaves** et **l'agent maître**. L'agent maître lance en parallèle plusieurs agents esclaves pour la sélection et la composition des services de qualité à partir d'un nombre important de services Cloud. Chaque agent esclave permet de fournir un ensemble de CCSs Pareto-optimal de solutions locales. L'agent maître regroupe les meilleures solutions locales des ensembles fournis par les agents esclaves afin de sélectionner l'ensemble de CCSs Pareto-optimal globale. La figure 4.21 montre le diagramme de classe du SCwA-CCS-2S et la figure 4.22 montre le diagramme de collaboration des agents maître et esclave.

Dans la deuxième phase, un agent est chargé de sélectionner un CCS optimal parmi les CCSs obtenus à partir de la première phase afin de le fournir au client selon ses préférences.

## 7.2 Description de l'algorithme SCwA-CCS-2S

Dans cette section nous présentons l'algorithme SCwA-CCS-2S qui est basé sur l'ontologie contextuelle présenté dans le chapitre 3, L'algorithme NSGA-II et les concepts agents. Nous avons introduit les agents dans cette nouvelle approche car le caractère concurrentiel des agents permet de découvrir des solutions optimales qui répondent aux contraintes du client dans un temps réduit. L'algorithme SCwA-CCS-2S permet la sélection du service Cloud composite optimal en termes de QoS avec un minimum degré de violation de contraintes du client. Il se base sur des services sémantiques concrets disponibles enregistrés dans un registre de l'ontologie. Les services Cloud concrets sont classés dans des services Cloud virtuels selon leurs fonctionnalités (rôle : "type de service - vente, achat, commande, etc.", entrées/sorties et contexte). La capacité des agents de supervision de contexte est exploitée pour réagir aux changements de contexte des services et/ou aux changements des préférences

des clients. L'utilisation des solutions trouvées permet d'exploiter l'expérience de recherche acquise par les agents de construction de solutions dans les itérations futures de l'algorithme. Les étapes de l'algorithme SCwA-CCS-2S sont :

### Etape1 – Génération automatique et sémantique du processus virtuel

Le modèle de processus d'affaire virtuel global est généré automatiquement par un agent maitre. Il utilise un algorithme de chaînage arrière et les relations sémantiques entre les services virtuels [14] selon les besoins fonctionnels des clients. Il est optimisé en se basant sur la réputation de chaque service virtuel pour éviter la redondance de services.

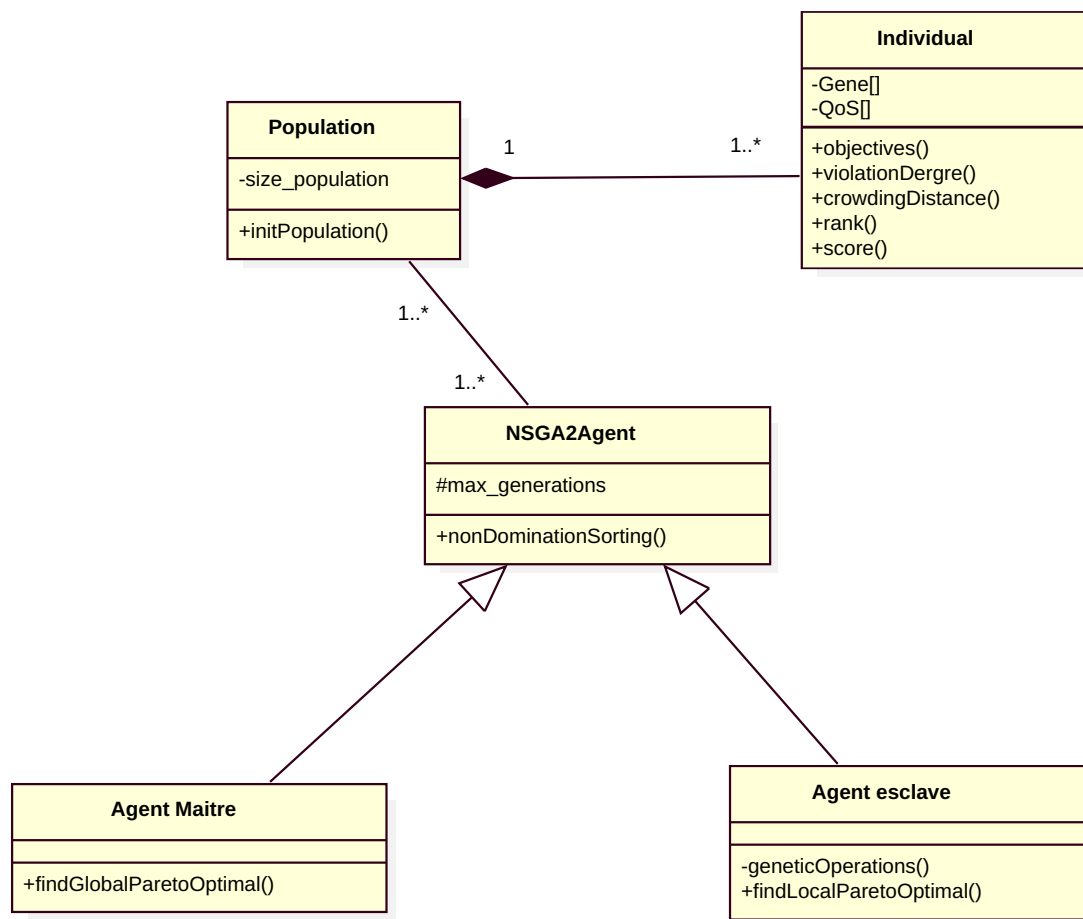


FIGURE 4.21 Diagramme de Classe de SCwA-CCS

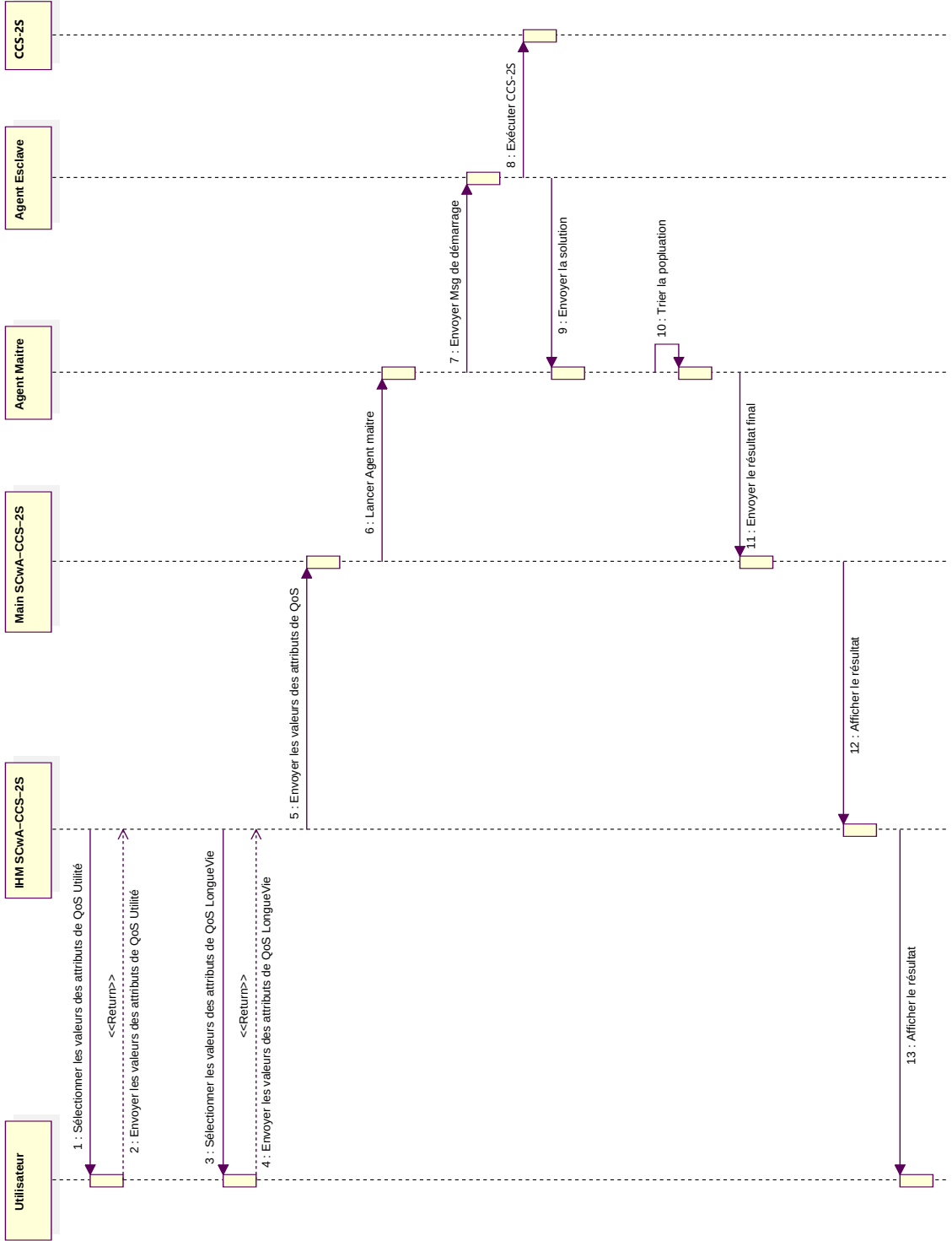


FIGURE 4.22 Diagramme d’interaction de SCwA–CCS

**Etape 2 (phase 1) – Optimisation parallèle multi-objectif au niveau de chaque agent esclave :**

Dans cette étape chaque agent esclave a comme objectif de vérifier la satisfaction des contraintes du client afin de sélectionner le meilleur service Cloud composite. En effet, lorsque les services concrets formant le CCS sont combinés ensemble, le service composite obtenu doit satisfaire la contrainte de qualité. Les agents visent à minimiser le degré de violation de contrainte et à maximiser les valeurs des attributs de QoS. L'optimisation est effectuée en adoptant le principe des algorithmes issus de la génétique (NSGA-II) guidée par une fonction de pénalité afin de sélectionner le meilleur service composite en termes de QoS en respectant les contraintes du client.

A la fin de toutes les itérations, les résultats obtenus par les agents esclaves sont fusionnés et ordonnés. Ensuite, les meilleurs services composites (Pareto-optimal global) sont sélectionnés.

**Etape 3 (phase 2) – Sélection de la solution finale optimale :**

Dans cette étape un agent est présent pour sélectionner la solution finale optimale à partir de l'ensemble Pareto-optimal des solutions obtenues de la première phase. En effet, cet agent évalue chaque CCS de l'ensemble de Pareto-optimal par un score calculé selon l'équation 4.4 et ensuite, il sélectionne le CCS ayant le meilleur score.

L'agent de gestionnaire de contexte évalue en permanence le contexte du client (les préférences et contraintes du client) pour fournir une solution CCS en conséquence. En cas de modification des préférences du client, l'agent chargé de la sélection de la solution finale optimale sélectionne le CCS approprié directement à partir des solutions du Pareto-optimal actuel sans devoir relancer les agents de la phase 1.

**7.3 Application de l'algorithme SCwA-CCS-2S à l'étude de cas**

Dans cette section, nous illustrons le déroulement de l'algorithme SCwA-CCS-2S appliqué à l'étude de cas présenté dans la section 2.1. Le processus d'affaire étudié contient 10 services virtuels et chaque service virtuel est lié à un ensemble de candidats comprenant 5 services Cloud. Les étapes de l'application de l'algorithme SCwA-CCS-2S à notre étude de cas sont résumées dans la figure 4.23.

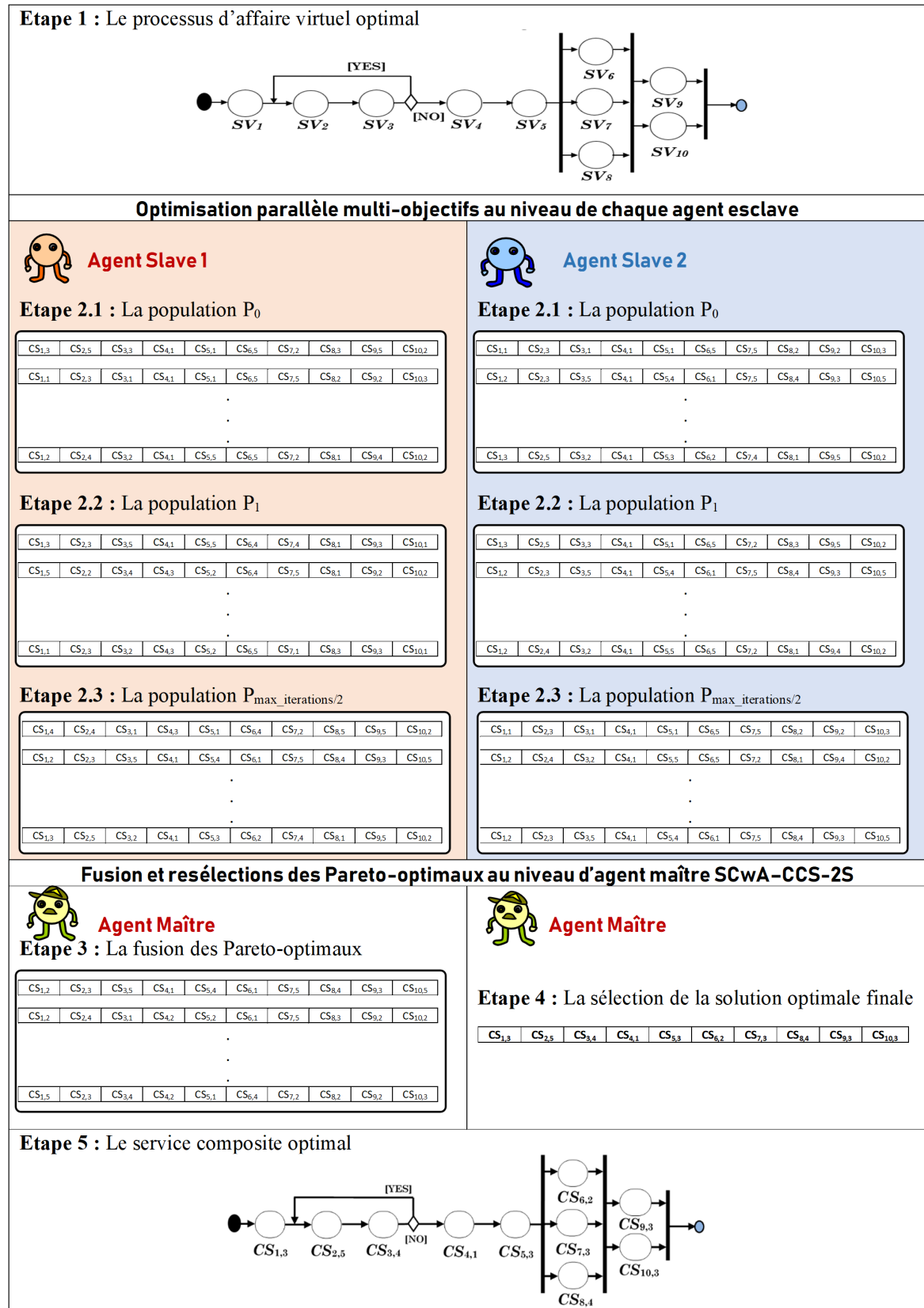


FIGURE 4.23 Les étapes de SCwA-CCS-2S appliquées à l'étude de cas

## 8 Conclusion

Dans ce chapitre, nous avons exposé notre approche à base d'ontologie, dédiée aux ERP sensibles au contexte. Après avoir présenté l'architecture et le modèle de fonctionnement de notre approche, nous avons détaillé les étapes de la construction de notre système. L'objectif de l'approche proposée est de fournir au client un ERP personnalisé qui répond à ses besoins fonctionnels, ainsi que ses contraintes et ses préférences non-fonctionnels dans un contexte spécifique. En effet, l'approche fournit automatiquement un processus d'affaire en basant sur l'ontologie présentée dans le chapitre précédent. Cette ontologie sert à la description sémantique des services d'affaires dans le Cloud et de guider le processus de composition afin de produire un ERP adapté aux exigences variables du client. Elle permet de générer automatiquement un processus d'affaire virtuel en fonction des besoins fonctionnels du client en utilisant l'algorithme de chainage arrière et des règles sémantiques. Ensuite, un algorithme à deux phases a été proposé pour l'optimisation contextuelle de la composition des services Cloud afin d'obtenir un processus d'affaire ERP personnalisé optimal en termes de QoS. Lors de la première phase, l'optimisation multi-objectif permet de générer un ensemble de solutions Pareto-optimal qui satisfont les contraintes du client en utilisant l'algorithme NSGA-II avec une nouvelle relation de dominance Pareto. Cette relation est basée sur une fonction de pénalité appelée degré de violation de contraintes. Elle est utilisée pour sélectionner les CCS pertinents qui respectent les contraintes du client. La deuxième phase sélectionne une solution optimale parmi les solutions de l'ensemble Pareto-optimal obtenus de la première phase en utilisant la pondération additive simple (SAW) pour évaluer le score de chaque solution. Une nouvelle technique a été proposée pour le calcul des poids des attributs de QoS basée sur les contraintes et les préférences du client.

En outre, nous avons proposé une extension de notre approche basée sur les agents. Cette extension permet de distribuer les étapes et les tâches du processus de composition de services aux différents agents pour améliorer les performances de notre système en termes de vitesse d'exécution.

Dans le chapitre suivant, nous allons présenter un prototype implémenté en java qui concrétise notre approche. Nous allons également présenter quelques évaluations et comparaisons avec d'autres méthodes existantes dans la littérature.

# Chapitre 5

## Réalisation : Prototype et évaluations

### Sommaire

---

|          |                                                    |            |
|----------|----------------------------------------------------|------------|
| <b>1</b> | <b>Introduction</b>                                | <b>120</b> |
| <b>2</b> | <b>Réalisation du prototype</b>                    | <b>121</b> |
| 2.1      | Environnement et outils techniques                 | 121        |
| 2.2      | Implémentation de l'ontologie                      | 121        |
| 2.3      | Présentation du prototype                          | 126        |
| 2.4      | Application du prototype à l'étude de cas          | 127        |
| <b>3</b> | <b>Évaluation des performances et comparaisons</b> | <b>129</b> |
| 3.1      | Évaluation de la pertinence des solutions          | 131        |
| 3.2      | Comparaison de la précision des solutions          | 132        |
| 3.3      | Comparaison des solutions Pareto-optimales         | 133        |
| 3.4      | Comparaisons sur la solution finale                | 134        |
| 3.5      | Amélioration des performances : Multi-agent        | 136        |
| <b>4</b> | <b>Conclusion</b>                                  | <b>138</b> |

---

### 1 Introduction

Notre challenge est de proposer un système qui permet de générer un ERP Cloud personnalisé qui s'adapte au contexte du client, ainsi que son environnement d'exécution via la composition sémantique des services Cloud. La prise en considération du contexte d'exécution au sein de notre ontologie permet au système d'offrir aux clients des ERP adaptés aux différents contextes.

Dans ce chapitre, nous exposons un prototype de système de génération d'ERP personnalisés sensibles au contexte. En outre, nous présentons des critères et des métriques

d'évaluations pour valider nos propositions. Nous présentons également une étude comparative entre les techniques utilisées dans notre approche avec celles existantes dans la littérature.

## 2 Réalisation du prototype

Afin de valider nos approches de composition dynamique et sémantique des services Cloud, nous avons utilisé notre modèle d'ontologie pour développer un prototype de génération automatique d'ERP personnalisés, flexibles et sensibles au contexte [108]. Le prototype permet au client de spécifier ses besoins fonctionnels et non-fonctionnels (contraintes et préférences). Le prototype fournit un ERP personnalisé via la composition des services Cloud appropriés. Cet ERP est optimal en matière de QoS selon les contraintes et les préférences spécifiées par le client. La spécificité de notre prototype est qu'il génère un ERP personnalisé et flexible qui respecte les contraintes contextuelles du client via la composition sémantique des services Cloud.

### 2.1 Environnement et outils techniques

Notre prototype a été développé en Java sous NetBeans pour la personnalisation d'ERP selon les exigences du client. Dans cette section nous présentons en premier lieu nos choix d'outils avant la description du fonctionnement du prototype dans le cadre de l'étude de cas "Commande de produits en ligne". Le tableau 5.1 décrit l'environnement et les outils techniques adoptés pour la réalisation de notre prototype. En effet, nous avons eu besoin d'outils d'implémentation et d'interrogation de l'ontologie proposée, ainsi que des outils nécessaires pour le développement de notre approche de composition et d'optimisation présentée dans le chapitre précédent.

### 2.2 Implémentation de l'ontologie

Dans le chapitre 3, nous avons décrit le modèle conceptuel de notre ontologie et nous avons détaillé tous ses concepts et ses relations. Dans cette section, nous présentons l'implémentation des modèles de l'ontologie CxQSCloudSERP. Ces modèles sont implémentés en langage OWL en utilisant l'éditeur Protégé. Nous présentons également quelques règles SWRL et quelques requêtes SPARQL. La figure 5.1 montre le modèle global de l'ontologie CxQSCloudSERP avec toutes ses classes et ses relations.

TABLE 5.1 Environnement technique

| Outil                            | Rôle                                                                                                                                                                                                 |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Java (Plateforme NetBeans [116]) | Environnement utilisé pour implémenter l'interface utilisateur pour la spécification des besoins du client, l'implémentation de l'algorithme CCS-2S et l'interaction avec l'ontologie CxQSCloudSERP. |
| Protégé-3.4.8 [117]              | Editeur utilisé pour implémenter l'ontologie CxQS-CloudSERP avec le langage OWL.                                                                                                                     |
| Java Expert System Shell (JESS)  | Moteur d'inférence utilisé pour exécuter les règles sémantiques sur les descriptions OWL et en déduire les résultats.                                                                                |
| SPARQL                           | Langage utilisé pour interroger l'ontologie par des requêtes sémantiques afin d'extraire tout type d'informations liées aux services Cloud.                                                          |
| SWRL                             | Langage utilisé pour exprimer les règles d'inférence sémantiques liées à l'ontologie.                                                                                                                |

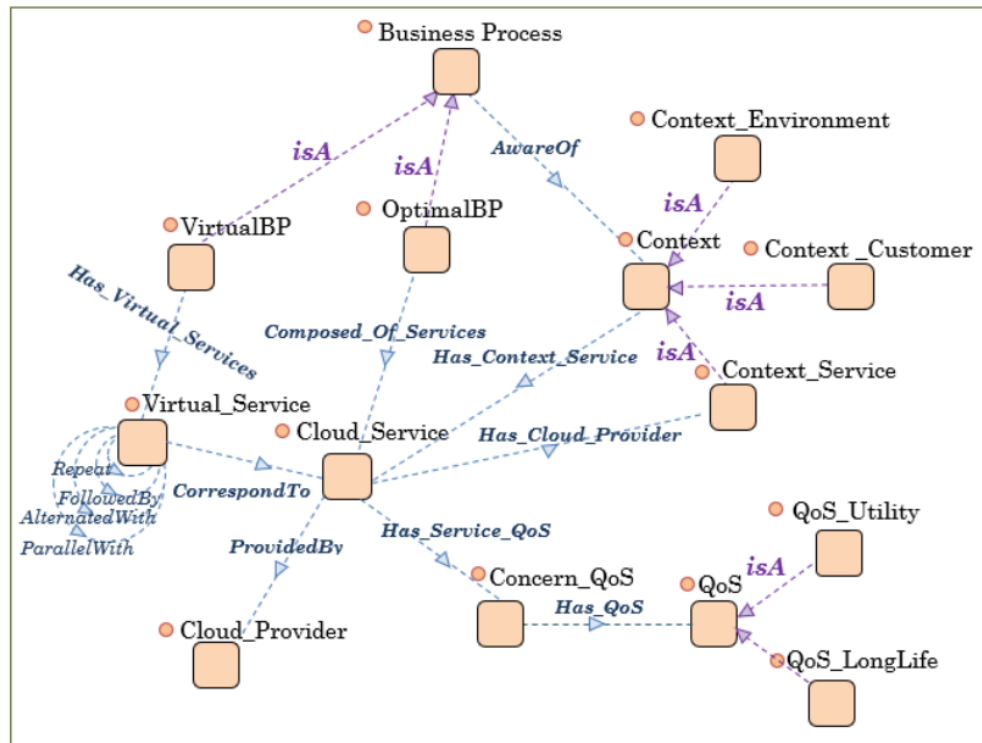


FIGURE 5.1 L'ontologie CxQSCloudSERP

La figure 5.2 illustre l'implémentation des classes et des propriétés de l'ontologie CxQS-CloudSERP et la figure 5.3 montre l'implémentation de ses individus par l'environnement "Protégé3.4.8".

Le tableau 5.2 montre les principales règles [112] qui permettent de générer automatiquement des processus d'affaires virtuels basés sur les besoins fonctionnels des clients et le tableau 5.3 décrit quelques requêtes SPARQL [111] qui servent à interroger les qualités sémantiques des services Cloud définis par le modèle ontologique. Les règles décrites dans les deux tableaux sont expliquées ci-dessous.

**Règle 1 :** Si les besoins fonctionnels d'un client "*FunctionalNeeds( ?needs)*" exprimés en tant qu'une expression complexe "*ExpressedBy( ?needs, ? Cexp)*" qui est composée d'expressions simples "*ExpressionComposedOf( ? Cexp, ? Sexp)*", alors le moteur de raisonnement basé sur des règles prend chaque service virtuel "*vs*" ayant les entrées "*VirtualServiceHasInput( ? vs, ? in)*" où chaque entrée exprime des paramètres sémantiques avec "*ParameterSemantic( ? in, ? s1)*" et correspond aux besoins fonctionnels particuliers du client "*FunctionalNeeds*", puis le moteur infère le premier service virtuel du processus d'affaire "*StartedBy( ? bp, ? vs)*" et affecte un numéro à la propriété "*step*" du service virtuel selon la valeur actuelle en utilisant "*HasStep( ? vs, ?current)*".

**Règle 2 :** Chaque service virtuel suivi du service virtuel courant ayant les mêmes paramètres d'entrées sémantiques et une conséquence comme expression simple, sera inclus dans le processus d'affaire.

**Règle 3 :** Si les contraintes explicites du client "*CustomerHasConstraints( ?E, ?C)*" s'expriment en expressions complexes "*ConstraintExpressedBy( ?C, ?Exp)*" composées d'expressions simples "*ExpressionComposedOf( ? Exp, ? Sexp)*", avec la valeur sémantique "High" "*SemanticValue( ? Sexp, "High")*", le moteur de raisonnement assigne "3" au rang de la contrainte.

**Règle 4 :** Cette règle permet de déduire la priorité de chaque QoS selon les contraintes et préférences du client. Les contraintes du client explicite sont exprimées en expressions complexes et expressions simples, chaque expression simple a un rang en utilisant "*ConstraintRank( ? Sexp, ? ConstraintRank)*". Une expression simple est définie par le nom de paramètre "*ParametreName( ? Sexp, ? Name)*" qui correspond aux préférences du client "*CustomerPreferenceName( ? Pref, ? Name)*". Le moteur de raisonnement déduit la priorité des préférences selon le rang de la contrainte, le nombre d'attributs de QoS et la priorité explicite en utilisant respectivement les opérations : "*swrlb : multiply( ? Multi, ? ConstraintRank, ? Count)*" et "*swrlb : add( ? Result\_priority, ? Multi, ? Exp\_priority)*".

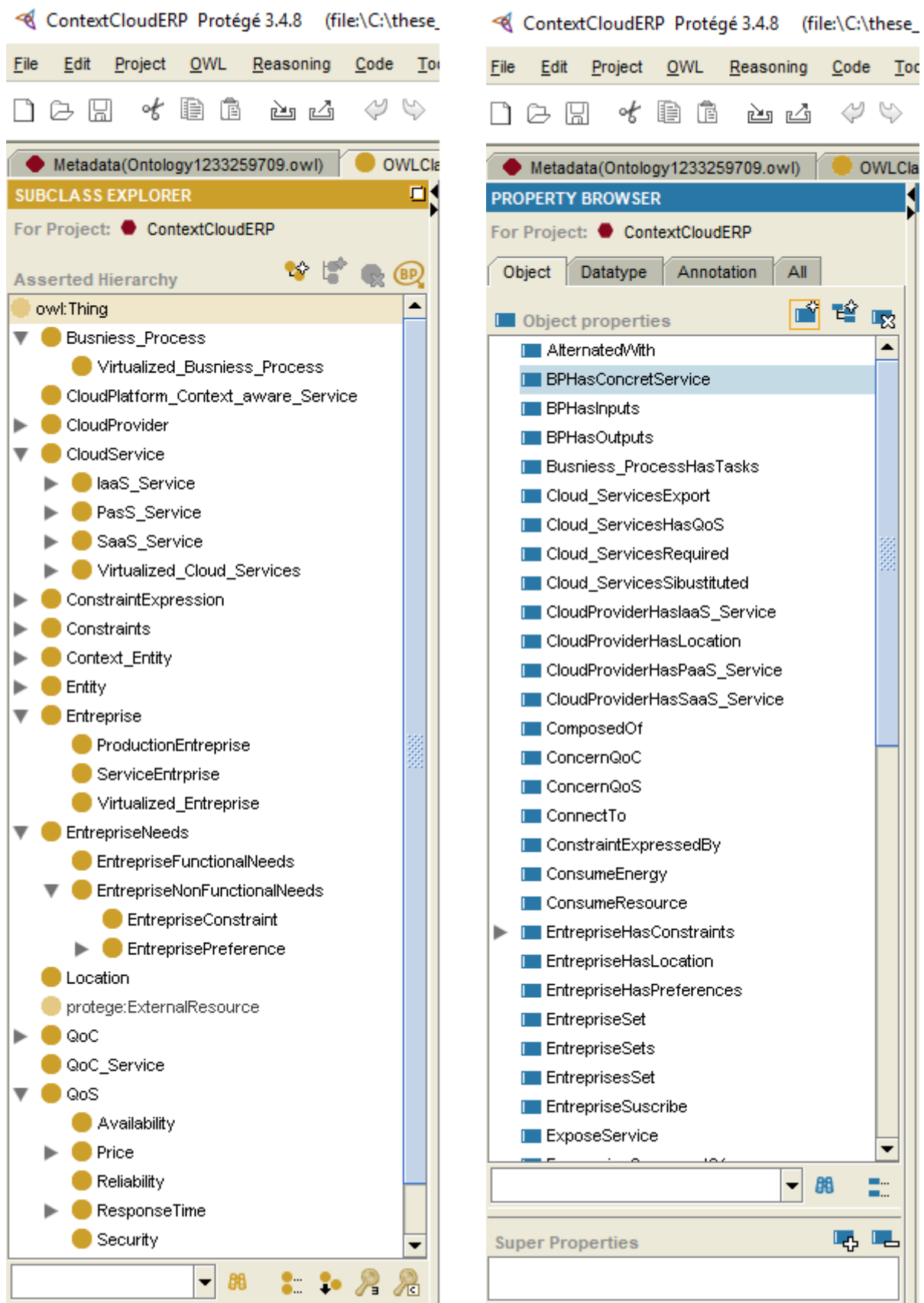


FIGURE 5.2 Les classes et les propriétés de l'ontologie CxQSCloudSERP

TABLE 5.2 Description des règles SWRL

| Description                                                                                                                                                                                                | Règles SWRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Règle 1 :</b> Effectue la correspondance entre les besoins fonctionnels du client avec les entrées des services virtuels afin de déduire les premiers services virtuels du processus d'affaire virtuel. | <i>Virtualized_Busniess_Process</i> (?bp) $\wedge$ <i>FunctionalNeeds</i> (?needs)<br>$\wedge$ <i>ExpressedBy</i> (?needs, ?cexp) $\wedge$ <i>BusinessType</i> (?cexp, "Sale")<br>$\wedge$ <i>ExpressionComposedOf</i> (?cexp, ?sexp) $\wedge$<br><i>HasParameter</i> (?sexp, ?parameter_in) $\wedge$ <i>ParameterSemantic</i> (?parameter_in, ?s) $\wedge$ <i>VirtualService</i> (?vs) $\wedge$<br><i>VSHasInput</i> (?vs, ?in) $\wedge$ <i>ParameterSemantic</i> (?in, ?s1) $\wedge$<br><i>swrlb:equal</i> (?s, ?s1) $\rightarrow$<br><i>Has_Virtual_Services</i> (?bp, ?vs) $\wedge$<br><i>HasStep</i> (?vs, "current") $\wedge$ <i>StartedBy</i> (?bp, ?vs)                                                                                                     |
| <b>Règle 2 :</b> Permet d'inclure les services virtuels suivi du service virtuel en cours dans le processus d'affaire virtuel.                                                                             | <i>Virtualized_Busniess_Process</i> (?bp) $\wedge$<br><i>Has_Virtual_Services</i> (?bp, ?vs) $\wedge$ <i>HasStep</i> (?vs, "current") $\wedge$<br><i>FollowedBy</i> (?vs, ?vs1) $\wedge$ <i>VSHasInput</i> (?vs1, ?in1) $\wedge$<br><i>ParameterSemantic</i> (?in1, ?s1) $\wedge$ <i>FunctionalNeeds</i> (?needs) $\wedge$<br><i>ExpressedBy</i> (?needs, ?cexp) $\wedge$ <i>BusinessType</i> (?cexp, "Sale") $\wedge$<br><i>ExpressionComposedOf</i> (?cexp, ?sexp) $\wedge$<br><i>HasParameter</i> (?sexp, ?param_in) $\wedge$<br><i>ParameterSemantic</i> (?param_in, ?s) $\wedge$ <i>swrlb:equal</i> (?s, ?s1) $\rightarrow$<br><i>Has_Virtual_Services</i> (?bp, ?vs1) $\wedge$ <i>HasStep</i> (?vs, "passe") $\wedge$<br><i>HasStep</i> (?vs1, "current")     |
| <b>Règle 3 :</b> Permet de déduire le rang de la contrainte de la valeur sémantique de QoS.                                                                                                                | <i>Customer</i> (?e) $\wedge$ <i>CustomerHasConstraints</i> (?e, ?c) $\wedge$<br><i>Constraint</i> (?c) $\wedge$ <i>ConstraintExpressedBy</i> (?c, ?exp) $\wedge$<br><i>ExpressionComposedOf</i> (?exp, ?sexp) $\wedge$<br><i>SemanticValue</i> (?sexp, "High") $\rightarrow$ <i>ConstraintRank</i> (?sexp, 3)                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Règle 4 :</b> Permet de déduire la priorité de chaque QoS à partir des contraintes et préférences du client.                                                                                            | <i>Customer</i> (?e) $\wedge$ <i>CustomerHasConstraints</i> (?e, ?c) $\wedge$<br><i>Constraint</i> (?c) $\wedge$ <i>ConstraintExpressedBy</i> (?c, ?exp) $\wedge$<br><i>ConstraintsCount</i> (?exp, ?count) $\wedge$<br><i>ExpressionComposedOf</i> (?exp, ?sexp) $\wedge$<br><i>ConstraintRank</i> (?sexp, ?constraintRank) $\wedge$<br><i>ParametreName</i> (?sexp, ?name) $\wedge$<br><i>CustomerHasPreferences</i> (?e, ?pref) $\wedge$<br><i>CustomerPreferenceName</i> (?pref, ?name) $\wedge$<br><i>ExplicitPriority</i> (?pref, ?exp_priority)<br>$\wedge$ <i>swrlb:multiply</i> (?multi, ?constraintRank, ?count) $\wedge$<br><i>swrlb:add</i> (?result_priority, ?multi, ?exp_priority)<br>$\rightarrow$ <i>DeducedPriority</i> (?pref, ?result_priority) |

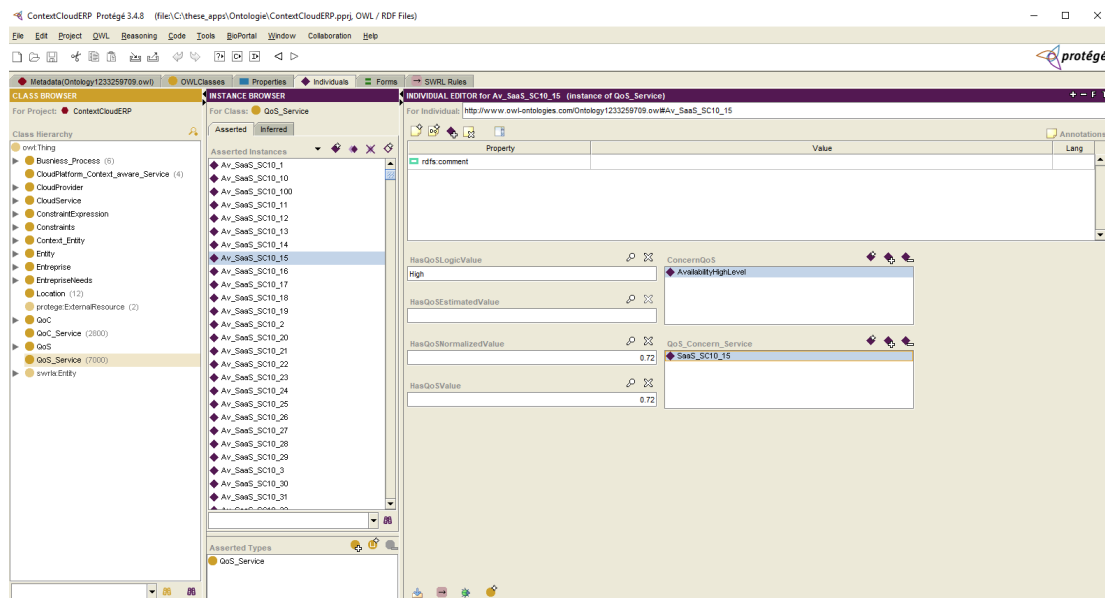


FIGURE 5.3 Les individus de l'ontologie CxQSCloudSERP

**Règle 5 :** Elle est utilisée pour extraire les services virtuels d'un processus d'affaire donné et toutes ses relations sémantiques.

**Règle 6 :** Elle est utilisée pour obtenir des services sémantiquement équivalents avec différents paramètres de QoS pour un service virtuel donné.

## 2.3 Présentation du prototype

Le prototype réalisé comporte les modules du processus de composition sémantique des services Cloud pour l'obtention d'un ERP personnalisé sensible au contexte. Il contient également des interfaces graphiques pour permettre au client de spécifier ses besoins fonctionnels et non-fonctionnels. Les fonctionnalités principales de notre prototype sont :

- **Gestion des besoins du client :** après la spécification des besoins du client via l'interface utilisateur, le système interprète les besoins fonctionnels en règles logiques et règles sémantiques. Les contraintes et préférences de QoS sont traduites en valeurs sémantiques et quantitatives en attribuant des poids pour chaque QoS.
- **Gestion des services :** la liste des services Cloud concrets est enregistrée dans un registre du proxy. Un service virtuel est associé à un ensemble de services concrets ayant les mêmes fonctionnalités avec des qualités de service différentes. Le gestionnaire de contexte est chargé de vérifier le contexte de service et de mettre à jour le registre des services.

TABLE 5.3 Description des règles SPARQL

| Description                                                                                                                                         | Requêtes SPARQL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Règle 5 :</b> Permet d'obtenir un processus d'affaire en tant que séquence de services virtuels et leurs relations.                              | <pre> PREFIX cloudERPonto : &lt;http://www.owlontologies.com/EO.owl#&gt; PREFIX rdf : &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; SELECT ?BP ?task ?taskname ?taskF ?taskP ?taskA WHERE {   ?BP cloudERPonto :BusinessProcessName \"\" + name + \"\" .   ?BP cloudERPonto :Busniess_ProcessHasTasks ?task .   ?task cloudERPonto :TaskHasName ?taskname . \"   ?task cloudERPonto :FollowedBy ?taskF .   ?task cloudERPonto :ParallelWith ?taskP .   ?task cloudERPonto :AlternatedWith ?TaskA . }; </pre>                                                                         |
| <b>Règle 6 :</b> Permet d'obtenir sémantiquement des services équivalents avec différents QoS et de les lier avec le service virtuel correspondant. | <pre> PREFIX cloudERPonto : &lt;http://www.owlontologies.com/EO.owl#&gt; PREFIX rdf : &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; SELECT ?sid ?sname ?slocation ?qos_service ?qos_value WHERE {   ?sc cloudERPonto :HasCloudServiceID ?sid .   ?sc cloudERPonto :HasCloudServiceLocation ?slocation .   ?sc cloudERPonto :HasCloudServiceName ?sname .   ?sv cloudERPonto :HasConcretService ?sc .   ?qos_service cloudERPonto :QoS_Concern_Service ?sc .   ?qos_service cloudERPonto :HasQoSValue ?qos_value .   ?sv cloudERPonto :TaskHasID \"\" + task_id + \"\" . \" }; </pre> |

- **Composition des services Cloud :** le système génère un processus d'affaire virtuel qui se compose de services virtuels selon les besoins fonctionnels du client. Une fois que les contraintes et les préférences non-fonctionnels du client sont spécifiées, le système génère le service Cloud composite approprié.
- **Optimisation des QoS :** le système fournit un processus d'affaire concret qui respecte les contraintes et les préférences du client via la composition des services Cloud concrets en optimisant les QoS d'utilité et les QoS longue-vie.

## 2.4 Application du prototype à l'étude de cas

Nous reprenons l'étude de cas du processus "Commande en ligne" (chapitre 4, section 2.1) pour montrer l'ensemble des fonctionnalités de notre prototype. En effet, le prototype doit engendrer un processus d'affaire "Commande de produit en ligne" composé de services

Cloud en optimisant les QoS selon la spécification des contraintes et des préférences du client. Le client dispose de l'interface d'accueil présentée par la figure 5.4 qui lui permet de spécifier ses besoins fonctionnels tels que le type d'activité, les entrées/sorties du processus et d'autres informations supplémentaires. Par exemple, un client a besoin d'un processus d'affaire de "Vente" ayant comme entrées : détails des produits, détails de client, transaction de type "INTERNET" et notification par email, et comme sorties : un ticket de réception. Le client peut modifier ses besoins à tout moment grâce à l'interface graphique. Le composant planificateur ERP-VCS génère sémantiquement et automatiquement un processus d'affaire virtuel en se basant sur les besoins fonctionnels du client. Le processus d'affaire résultant de l'étude de cas B2B implique 10 services virtuels (voir figure 4.6).

FIGURE 5.4 GUI pour les besoins fonctionnels du client

Une fois le processus d'affaire virtuel est généré, le client spécifie ses besoins non fonctionnels sous forme de contraintes en termes de  $QoS_U$  (temps, prix, fiabilité, disponibilité et sécurité) et  $QoS_L$  (confiance et débit) comme le montre la figure 5.5. Le système génère un ensemble de solutions Pareto-optimales qui respectent les contraintes du client (phase 1). Les valeurs sémantiques des contraintes sont associées à des intervalles tels que :

- "high" correspond à l'intervalle  $[0.7, 1]$
- "medium" correspond à l'intervalle  $[0.4, 0.7[$
- "low" correspond à l'intervalle  $[0.1, 0.4[$

FIGURE 5.5 Interface pour la spécification des contraintes QoS de l'ERP

Enfin, pour sélectionner la solution optimale finale, le client doit spécifier l'ordre de ses préférences comme indiqué sur la figure 5.6. En effet, le système calcule les poids des attributs selon l'équation 3.4 et sélectionne la solution optimale ayant le meilleur score parmi l'ensemble des solutions du Pareto-optimal (c.-à-d. Services composites concrets) (phase 2). Le score de chaque solution est calculé par l'équation 4.4. Par conséquent, si le client change ses préférences, l'application sélectionne la nouvelle solution directement à partir de l'ensemble Pareto-optimal actuel sans générer un nouvel ensemble Pareto-optimal (c.-à-d. sans exécuter la phase 1).

La figure 5.7 montre le meilleur service composite concret utilisant un système de prise de décision. Le client a spécifié deux contraintes (temps d'exécution plus rapide et un prix plus bas), et il a préféré le temps d'exécution par rapport au prix. Le résultat respecte les contraintes ( $0.7245 > 0.7$ ,  $0.9892 > 0.7$ ) et répond aux préférences ( $0.9892 \gg 0.7245$ ).

Dans ce qui suit, nous nous intéressons à l'évaluation de notre approche de composition des services Cloud dédiée aux ERPs.

### 3 Évaluation des performances et comparaisons

Après avoir présenté l'implémentation de notre système et son application sur une d'étude de cas, nous abordons les expérimentations effectuées pour évaluer la performance de notre approche de composition de services Cloud par rapport aux solutions existantes des différents auteurs. Nous présentons également des résultats de comparaisons avec ces solutions selon différents critères. Nous avons réalisé toutes nos expérimentations sur un PC avec un processeur Intel 3 GHz, une RAM 8Go, le système d'exploitation Windows 7 64 bits,

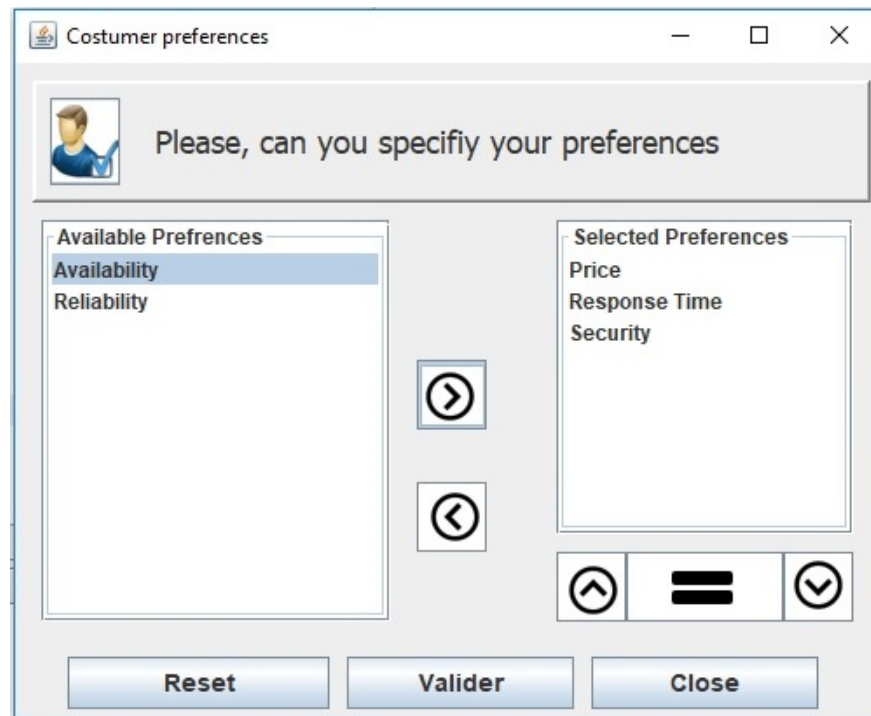


FIGURE 5.6 Interface pour la spécification de l'ordre des préférences

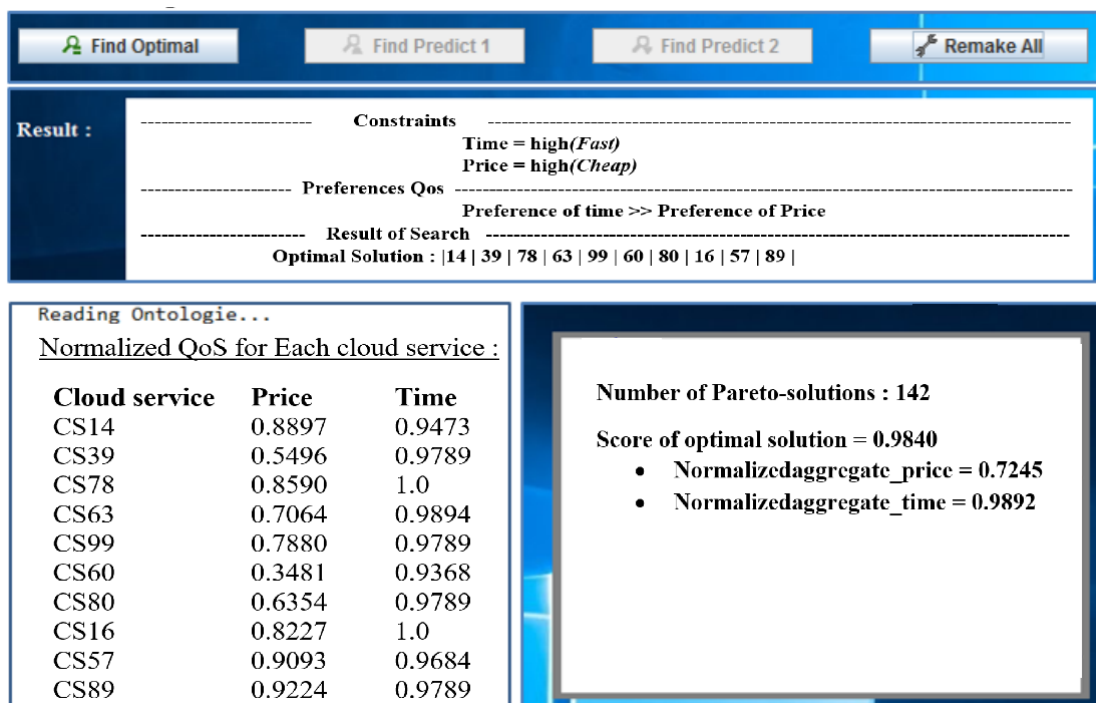


FIGURE 5.7 Le résultat de la solution optimale du cas d'étude par notre prototype

NetBeans et MATLAB. Un jeu de données est considéré comprenant 10 services virtuels. Pour chaque service virtuel, l'ensemble de services candidat associé contient 100 services Cloud concrets (l'espace de recherche est de taille  $10^{100}$ ). Chaque service concret considère cinq attributs  $QoS_U$  et deux attributs  $QoS_L$  dont leurs valeurs sont normalisées. Les valeurs de QoS sont générées de manière aléatoire. Par souci de clarté, les résultats sont présentés avec deux attributs de QoS.

### 3.1 Évaluation de la pertinence des solutions

Le taux de violation (violation rate) est utilisé pour mesurer la pertinence de la solution obtenue par l'approche proposée en termes du respect des contraintes définies par le client. La solution pertinente est la solution avec le taux le plus bas. Ce taux est défini comme un rapport entre les solutions qui violent les contraintes du client et l'ensemble de toutes les solutions obtenues. Il est calculé par la formule suivante :

$$violation\_rate = \frac{|\{constraints\_violation\_solutions\}|}{|\{all\_solutions\}|} \quad (5.1)$$

La taille de la population varie entre 50 et 200 et le nombre d’itérations augmente de 200 à 2000, la probabilité de mutation est de 0,4. Les contraintes client sont définies par  $C_{time}^{High} \in [0.7, 1]$  et  $C_{price}^{High} \in [0.7, 1]$ . Comme le montre le tableau 5.4, en variant le nombre de générations et la taille de la population, le taux de violation du CCS-2S est égal à 0 lorsque la population dépasse 150 individus ou lorsque le nombre de générations dépasse 1600. Ces résultats montrent que notre approche respecte de manière significative les contraintes du client lorsque le nombre d’itérations et la taille de la population sont choisis de manière appropriée.

TABLE 5.4 Le taux de violation de CCS-2S

[illegible]

### 3.2 Comparaison de la précision des solutions

Afin d'évaluer la précision de la solution Pareto-optimale trouvée, nous avons utilisé une métrique de comparaison qui se présente comme une distance entre le Pareto-optimal attendu et le Pareto-optimal trouvé (voir figure 5.8). Cette distance est calculée par l'équation 5.2.

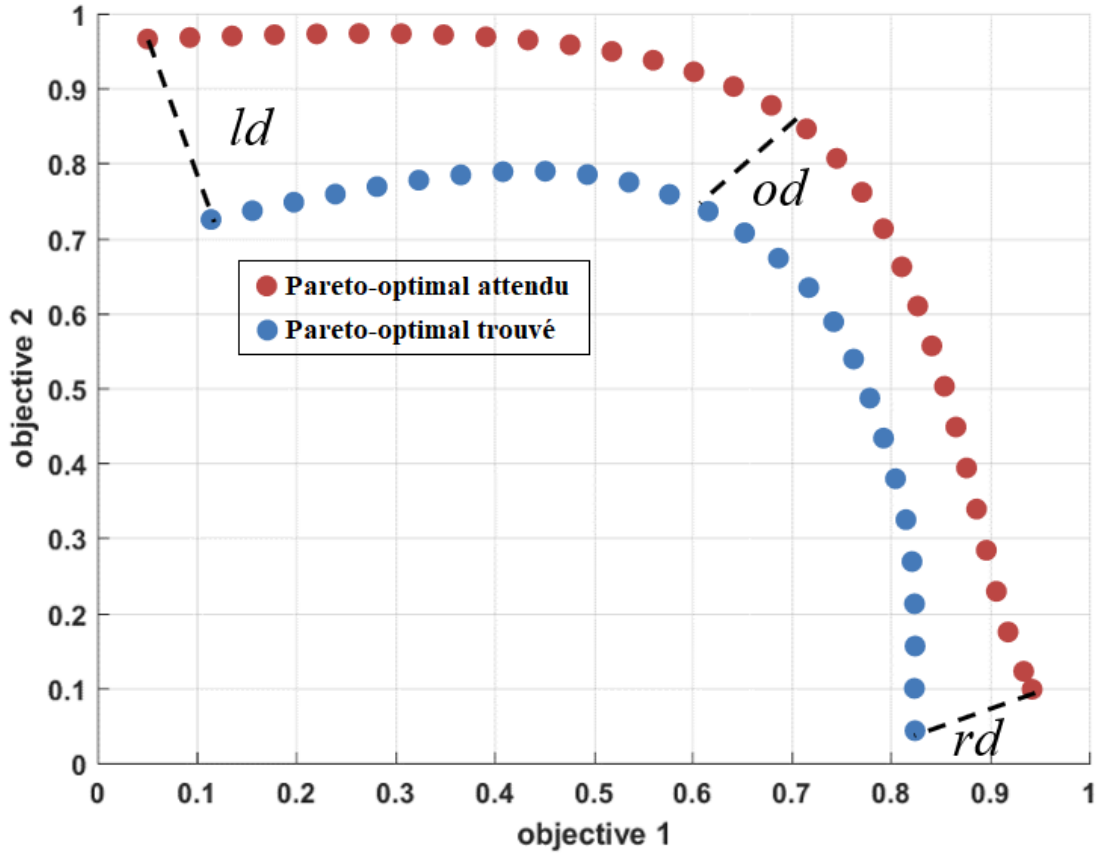


FIGURE 5.8 La distance entre le Pareto-optimal attendu et le Pareto-optimal trouvé

$$d(P_e, P_f) = ld(P_e, P_f) + rd(P_e, P_f) + od(P_e, P_f) \quad (5.2)$$

où :

**ld** est la distance euclidienne entre la limite gauche du Pareto-optimal attendu et le Pareto-optimal trouvé. **rd** est la distance euclidienne entre la limite droite du Pareto-optimal attendu et le Pareto-optimal trouvé. **od** est la distance euclidienne entre la solution optimale calculée à la phase 2 (étape 3 de l'algorithme CCS-2S) avec des préférences égales, du Pareto-optimal attendu et du Pareto-optimal trouvé.

Le tableau 5.5 montre les distances entre le Pareto-optimal attendu et le Pareto-optimal trouvé par : CCS-2S, NSGA-II, MOPSO et SPEA2 pour des différentes tailles de population et de nombres d'itérations.

TABLE 5.5 La distance entre la solution Pareto-optimal attendu est la solution Pareto-optimal trouvé

| Population size |            | Number of iterations |        |        |        |        |        |          |          |          |          |
|-----------------|------------|----------------------|--------|--------|--------|--------|--------|----------|----------|----------|----------|
|                 |            | 200                  | 400    | 600    | 800    | 1000   | 1200   | 1400     | 1600     | 1800     | 2000     |
| <b>CCS-2S</b>   | <b>50</b>  | 1.2312               | 0.0073 | 0.0269 | 0.0358 | 1.2953 | 1.1525 | 0.0387   | 0.0195   | <b>0</b> | 0.0096   |
|                 | <b>100</b> | 1.2928               | 1.0916 | 1.2876 | 1.2796 | 0.1063 | 1.0773 | 0.0096   | 0.0554   | <b>0</b> | <b>0</b> |
|                 | <b>150</b> | 1.2406               | 1.1558 | 1.282  | 1.246  | 0.0203 | 0.0072 | 0.0096   | 0.0032   | <b>0</b> | <b>0</b> |
|                 | <b>200</b> | 0.4285               | 1.2268 | 1.0177 | 1.0569 | 0.0029 | 0.2631 | <b>0</b> | <b>0</b> | <b>0</b> | <b>0</b> |
| <b>NSGAII</b>   | <b>50</b>  | 0.4396               | 0.1652 | 0.2656 | 0.4246 | 0.3027 | 0.4581 | 0.193    | 0.1753   | 0.3453   | 0.0774   |
|                 | <b>100</b> | 0.3604               | 0.2435 | 0.1002 | 0.36   | 0.2876 | 0.2459 | 0.2844   | 0.0882   | 0.2387   | <b>0</b> |
|                 | <b>150</b> | 0.4192               | 0.2735 | 0.4797 | 0.2382 | 0.2215 | 0.2226 | 0.1511   | <b>0</b> | 0.0457   | <b>0</b> |
|                 | <b>200</b> | 0.3744               | 0.1174 | 0.3013 | 0.2844 | 0.1357 | 0.3048 | 0.0674   | 0.372    | <b>0</b> | <b>0</b> |
| <b>MOPSO</b>    | <b>50</b>  | 1.6543               | 1.5235 | 1.5037 | 1.4839 | 1.6419 | 1.4007 | 1.3815   | 1.3623   | 1.3431   | 1.3239   |
|                 | <b>100</b> | 1.3970               | 1.6380 | 1.4235 | 1.4019 | 1.1723 | 1.1540 | 1.1356   | 1.1173   | 1.0990   | 1.0807   |
|                 | <b>150</b> | 1.1031               | 1.4549 | 1.2545 | 0.9839 | 0.9660 | 0.9482 | 0.9303   | 0.9124   | 0.8945   | 0.8766   |
|                 | <b>200</b> | 1.1163               | 1.0895 | 1.0699 | 1.0466 | 1.0233 | 1.0001 | 0.9768   | 0.9536   | 0.9303   | 0.9071   |
| <b>SPEA2</b>    | <b>50</b>  | 1.6255               | 1.6574 | 1.6262 | 1.3763 | 1.3580 | 1.3644 | 1.3457   | 1.2837   | 1.2657   | 1.2896   |
|                 | <b>100</b> | 1.4101               | 1.3458 | 1.2335 | 1.2056 | 1.1961 | 1.1775 | 1.3426   | 1.3282   | 1.3065   | 1.1375   |
|                 | <b>150</b> | 1.1270               | 1.1450 | 1.0466 | 1.0201 | 1.0106 | 0.9906 | 1.1127   | 0.9910   | 1.0749   | 0.8992   |
|                 | <b>200</b> | 1.0540               | 1.0057 | 0.8317 | 0.9547 | 0.9335 | 0.9499 | 0.9278   | 0.8698   | 0.8486   | 0.8709   |

Les contraintes client sont définies par  $C_{time}^{High} \in [0.7, 1]$  et  $C_{price}^{High} \in [0.7, 1]$ . Les résultats obtenus montrent que l'optimisation dans notre approche atteint la solution Pareto-optimale avec moins de générations par rapport aux autres algorithmes, à cause de l'élimination précoce des services qui violent les contraintes client lors de la sélection du service à partir de la liste des services candidats.

### 3.3 Comparaison des solutions Pareto-optimales

Afin d'évaluer davantage les performances de notre approche, les solutions Pareto-optimales de NSGA-II et de CCS-2S ont été comparées en matière du respect des contraintes du client. Pour cette comparaison, les contraintes de temps et de confiance sont définies par  $C_{time} \in [0.7, 1]$  et  $C_{trust} \in [0.7, 1]$ . La taille de la population est fixée à 100 et le nombre de

générations est égale à 1500. La figure 5.9 montre que dans le cas de CCS-2S, les solutions expérimentales Pareto-optimales respectent parfaitement les contraintes du client (étoiles rouges) contrairement à NSGA-II (cercles bleus). Ce résultat est dû au fait que CCS-2S élimine tout CCS qui viole les contraintes du client tandis que NSGA-II considère toutes les solutions CCS disponibles.

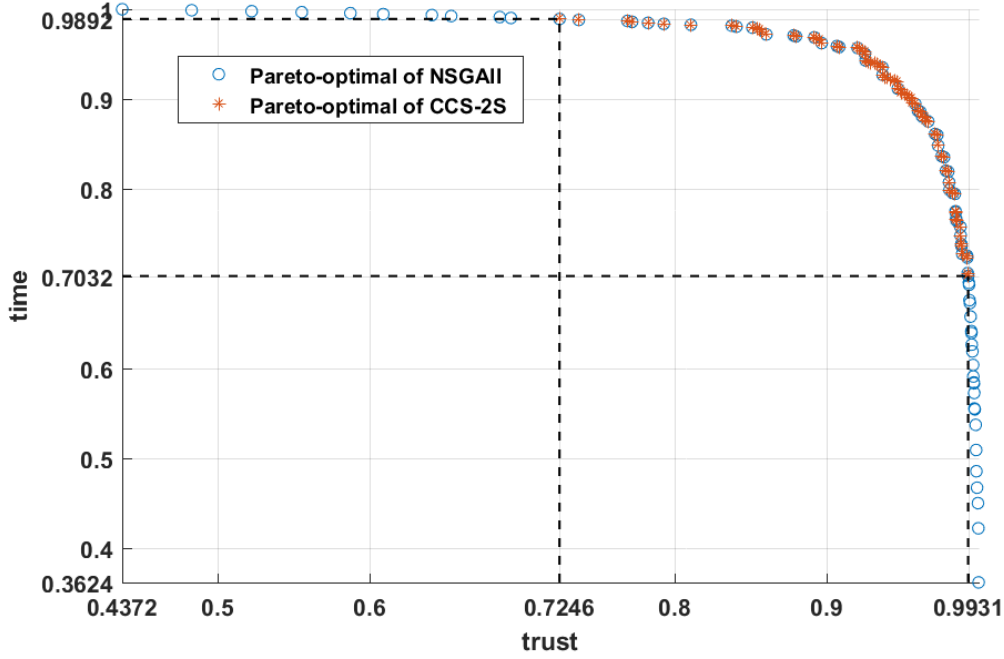


FIGURE 5.9 Comparaison des solutions Pareto-optimales entre CCS-2S et NSGA-II

### 3.4 Comparaisons sur la solution finale

La personnalisation non-fonctionnelle du processus d'affaire ERP offre à l'approche proposée la faculté de s'adapter aux différentes contraintes et préférences du client, sensibles au contexte. La figure 5.10a et la figure 5.10b montrent la solution Pareto-optimale et la solution finale obtenus par l'algorithme CCS-2S avec violation de contraintes et sans violation de contraintes respectivement selon les trois préférences possibles du client (un client peut accorder plus d'importance au temps d'exécution par rapport au prix, un autre peut préférer le prix par rapport au temps d'exécution, tandis que le troisième s'en fout). Les contraintes de tous les clients sont spécifiées par : *temps d'exécution rapide* et *prix moins cher*. Ces deux contraintes sont interprétées en  $C_{time} \in [0.7, 1]$  et  $C_{price} \in [0.7, 1]$ .

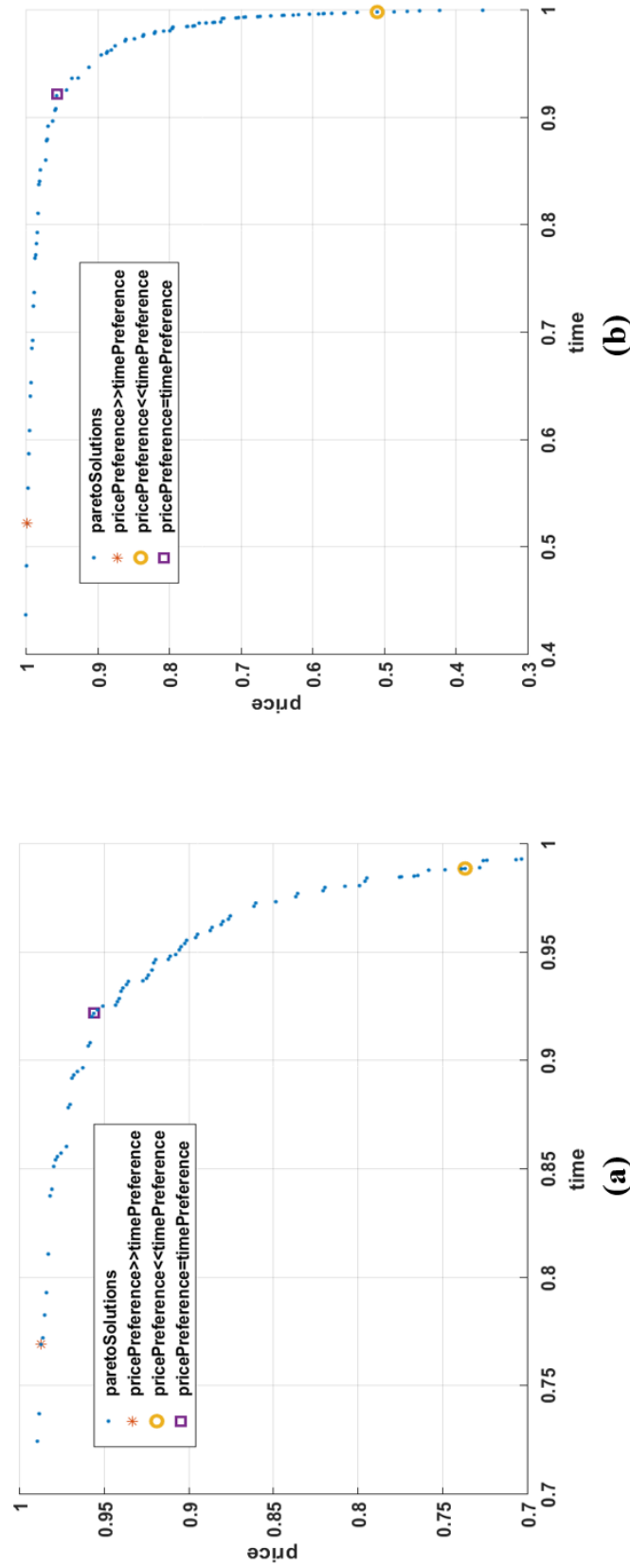


FIGURE 5.10 La solution finale optimale obtenue par CCS-2S en utilisant les préférences du client (temps et prix) : (a) avec violation de contraintes, (b) sans violation de contraintes.

Dans la figure 5.10a, CCS-2S cherche un Pareto-optimal qui respecte les contraintes QoS du client et fournit de meilleures solutions selon les différentes préférences du client. Le tableau 5.6 montre les valeurs de QoS (temps, prix) avec différentes spécifications de préférences en utilisant l'algorithme CCS-2S dans deux cas : avec violation de contraintes et sans violation de contraintes. Lorsque les préférences (le temps d'exécution et le prix) sont égales, aucune violation de contrainte ne s'est produite. Alors que dans le cas où les préférences sont différentes, la solution optimale utilisant NSGA-II viole les contraintes du client comme illustré dans le tableau 5.6 et la figure 5.10b.

TABLE 5.6 Valeurs de QoS "temps" et "prix" obtenues par CCS-2S avec les différentes spécifications de préférences.

|                            | CCS-2S avec violation de contraintes |        | CCS-2S sans violation de contraintes |               |
|----------------------------|--------------------------------------|--------|--------------------------------------|---------------|
|                            | Time                                 | Price  | Time                                 | Price         |
| <b>PPrice&gt;&gt;PTime</b> | 0.7692                               | 0.9870 | <b>0.5222</b>                        | 0.9978        |
| <b>PPrice&lt;&lt;PTime</b> | 0.9887                               | 0.7365 | 0.9983                               | <b>0.5096</b> |
| <b>PPrice=PTime</b>        | 0.9219                               | 0.9559 | 0.9219                               | 0.9559        |

La solution finale CCS-2S est comparée à l'optimisation mono-objectif (scalarisation) par les méta-heuristiques PSO et ICA. L'optimisation mono-objective engendre une solution optimale avec une valeur de fitness (score) optimale mais cette solution ne respecte pas les contraintes du client. Le tableau 5.7 montre le score, les QoS (temps, débit) optimisées et le taux de violation de QoS de chaque algorithme. Les valeurs en gras représentent les valeurs qui violent la contrainte de QoS.

### 3.5 Amélioration des performances : Multi-agent

Afin de justifier l'utilisation de l'aspect multi-agent présenté dans le chapitre précédent en tant qu'extension de notre approche, des tests de comparaison sont effectués entre l'algorithme SCwA-CCS-2S (multi-agent) avec l'algorithme CCS-2S (sans agent) en variant le nombre d'itérations (de 250 à 2500) [115]. Le nombre d'agents esclaves utilisés pour trouver l'ensemble Pareto des solutions locales est égale à *quatre*. Le nombre de services virtuels est fixé à 10 et chaque service virtuel est associé à 100 services concrets. Chaque service concret contient deux attributs de QoS (Temps d'exécution et le Prix).

L'algorithme SCwA-CCS-2S lance un ensemble d'agents pour trouver les meilleurs services composites de la population dans un temps réduit grâce à la propriété coopérative et parallèle des agents permettant d'accélérer le processus d'optimisation. La figure 5.11 montre

TABLE 5.7 Comparaisons sur la solution finale avec l'optimisation mono-objectif.

| Number<br>of VS | Number<br>of CS |        | Fitness<br>(Score) | Normalized<br>Time | Normalized<br>Throughput | Violation<br>Ratio |
|-----------------|-----------------|--------|--------------------|--------------------|--------------------------|--------------------|
| 5               | 25              | PSO    | 0.7216             | 0.9200             | <b>0.5231</b>            | 0.1263             |
|                 |                 | ICA    | 0.7212             | 0.9208             | <b>0.5217</b>            | 0.1273             |
|                 |                 | CCS-2S | 0.7201             | 0.7391             | 0.7011                   | <b>0</b>           |
|                 | 50              | PSO    | 0.7440             | 0.9060             | <b>0.5821</b>            | 0.0842             |
|                 |                 | ICA    | 0.7442             | 0.8868             | <b>0.6006</b>            | 0.0710             |
|                 |                 | CCS-2S | 0.7315             | 0.7606             | 0.7024                   | <b>0</b>           |
|                 | 100             | PSO    | 0.7564             | 0.9399             | <b>0.5729</b>            | 0.0907             |
|                 |                 | ICA    | 0.7584             | 0.9543             | <b>0.5625</b>            | 0.0982             |
|                 |                 | CCS-2S | 0.7499             | 0.7985             | 0.7013                   | <b>0</b>           |
| 10              | 25              | PSO    | 0.7217             | 0.9206             | <b>0.5228</b>            | 0.1265             |
|                 |                 | ICA    | 0.7202             | 0.9162             | <b>0.5242</b>            | 0.1255             |
|                 |                 | CCS-2S | 0.7196             | 0.7373             | 0.7019                   | <b>0</b>           |
|                 | 50              | PSO    | 0.7440             | 0.9259             | <b>0.5621</b>            | 0.0985             |
|                 |                 | ICA    | 0.7457             | 0.9170             | <b>0.5745</b>            | 0.0896             |
|                 |                 | CCS-2S | 0.7418             | 0.7788             | 0.7048                   | <b>0</b>           |
|                 | 100             | PSO    | 0.7557             | 0.8763             | <b>0.6350</b>            | 0.0464             |
|                 |                 | ICA    | 0.7523             | 0.8525             | <b>0.6521</b>            | 0.0342             |
|                 |                 | CCS-2S | 0.7505             | 0.7968             | 0.7042                   | <b>0</b>           |
| 15              | 25              | PSO    | 0.7382             | 0.9022             | <b>0.5742</b>            | 0.0898             |
|                 |                 | ICA    | 0.7316             | 0.8791             | <b>0.5841</b>            | 0.0827             |
|                 |                 | CCS-2S | 0.7328             | 0.7624             | 0.7014                   | <b>0</b>           |
|                 | 50              | PSO    | 0.7552             | 0.8954             | <b>0.6149</b>            | 0.0607             |
|                 |                 | ICA    | 0.7515             | 0.9169             | <b>0.5861</b>            | 0.0813             |
|                 |                 | CCS-2S | 0.7509             | 0.7995             | 0.7023                   | <b>0</b>           |
|                 | 100             | PSO    | 0.7591             | 0.9026             | <b>0.6155</b>            | 0.0603             |
|                 |                 | ICA    | 0.7588             | 0.9280             | <b>0.5896</b>            | 0.0788             |
|                 |                 | CCS-2S | 0.7563             | 0.8107             | 0.7019                   | <b>0</b>           |
| 20              | 25              | PSO    | 0.7430             | 0.8721             | <b>0.6139</b>            | 0.0615             |
|                 |                 | ICA    | 0.7407             | 0.9223             | <b>0.5591</b>            | 0.1006             |
|                 |                 | CCS-2S | 0.7384             | 0.7754             | 0.7014                   | <b>0</b>           |
|                 | 50              | PSO    | 0.7620             | 0.9184             | <b>0.6055</b>            | 0.0675             |
|                 |                 | ICA    | 0.7623             | 0.9323             | <b>0.5923</b>            | 0.0769             |
|                 |                 | CCS-2S | 0.7578             | 0.8113             | 0.7043                   | <b>0</b>           |
|                 | 100             | PSO    | 0.7634             | 0.9360             | <b>0.5907</b>            | 0.0780             |
|                 |                 | ICA    | 0.7693             | 0.9378             | <b>0.6008</b>            | 0.0708             |
|                 |                 | CCS-2S | 0.7611             | 0.8204             | 0.7018                   | <b>0</b>           |

la performance de l'algorithme utilisant l'aspect multi-agent en matière de temps d'exécution par rapport à l'algorithme sans agents. En particulier lorsque le nombre d'itérations augmente. En effet, pour 2500 itérations, le temps d'exécution de l'algorithme CCS-2S est près de 230 secondes (226968 millisecondes), alors que le temps d'exécution de l'algorithme SCwA-CCS-2S ne dépasse pas 10 secondes (9265 millisecondes).

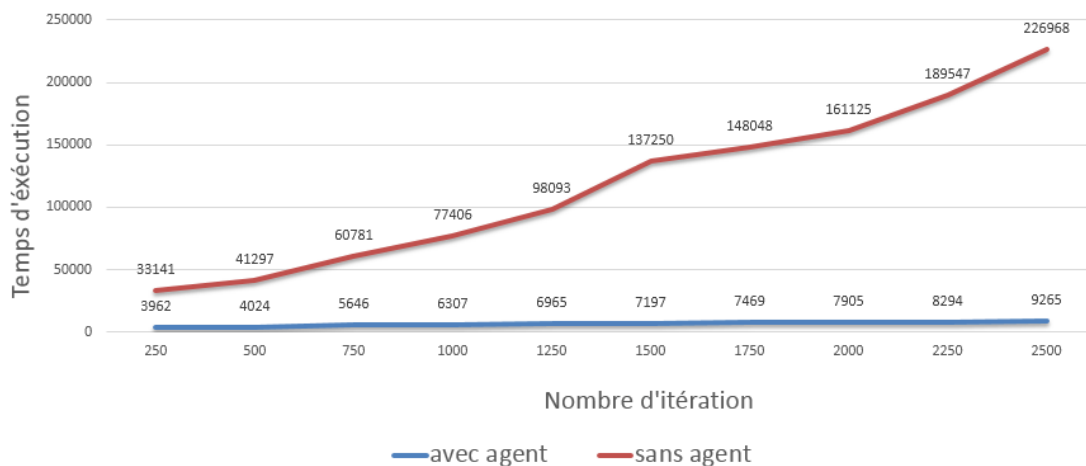


FIGURE 5.11 Temps d'exécution de CCS-2S vs SCwA-CCS-2S

## 4 Conclusion

Ce chapitre est consacré à la réalisation et la validation expérimentale de nos approches de composition dynamique et sémantique des services Cloud dédiée aux ERPs. Nous avons commencé par décrire l'implémentation de l'ontologie proposée qui joue le rôle du noyau de notre système, avec quelques règles SPARQL qui servent à interroger les données de l'ontologie et quelques règles sémantiques SWRL. Ensuite, nous avons présenté un prototype implémenté en JAVA pour la génération de systèmes ERP personnalisés sensibles au contexte avec ses différentes interfaces et les résultats obtenus par son application à l'étude de cas du "Commande de produits en ligne". Ce prototype peut être complété et hébergé en tant que plateforme de Cloud qui permet de générer des ERPs Cloud personnalisés sensibles au contexte.

En outre, nous avons positionné nos propositions par rapport aux travaux existants en comparant nos résultats avec ceux des autres approches en appuyant sur quelques critères et métriques d'évaluation. En effet, nous avons comparé la première phase de l'algorithme CCS-2S avec d'autres méthodes multi-objectif existantes (NSGA-II, SPEA2 et MOPSO). Ensuite, nous avons comparé les résultats obtenus de la deuxième phase de CCS-2S qui permet de

générer la solution finale optimale avec des algorithmes méta-heuristiques mono-objectif (PSO et ICA). Les expérimentations montrent que l'algorithme CCS-2S fournit des résultats pertinents et précis par rapport aux autres algorithmes, notamment le respect des contraintes du client. Nous avons également montré expérimentalement que l'extension multi-agent de notre approche est plus performante en matière de vitesse d'exécution.

# Conclusion générale

L'objectif global de ce travail de thèse est de réaliser un système permettant de générer des processus d'affaires ERP personnalisés adaptés au contexte du client. En effet, le système doit réagir convenablement aux différents changements du contexte. Pour cette raison, nous avons divisé le contexte en trois catégories principales : contexte client (entreprise), contexte fournisseur Cloud, et contexte de l'environnement. Le contexte client est défini par les besoins fonctionnels et non-fonctionnels du client, tandis que les contextes fournisseur et environnement sont liés respectivement aux critères de QoS des services Cloud (prix, temps d'exécution, etc.) et les QoS de l'environnement (débit, trust, et.). Notre travail vise à fournir au client un processus d'affaire ERP pertinent qui devrait répondre aux changements éventuels des besoins du client en tenant compte de ses contraintes et ses préférences contextuelles.

En premier lieu nous avons proposé une nouvelle ontologie pour une meilleure orientation du processus de génération automatique et sémantique du processus d'affaire ERP adapté aux besoins fonctionnels et non-fonctionnels du client. Elle est utilisée pour la description sémantique des services Cloud hétérogènes et des processus d'affaires. Elle permet d'interpréter sémantiquement les contraintes et préférences de QoS du client en tant que valeurs qualitatives et déduire leurs valeurs quantitatives grâce à des règles d'inférence. Elle permet également de générer automatiquement un processus d'affaire virtuel en fonction des besoins fonctionnels du client.

Afin de générer un processus d'affaire ERP personnalisé et optimal en termes de QoS, un algorithme à deux phases est proposé pour l'optimisation de la qualité contextuelle des services composites engendrés par le processus de composition des services Cloud. Lors de la première phase, l'optimisation multi-objectif est utilisée pour obtenir un ensemble de solutions Pareto-optimales qui répondent aux contraintes des clients en utilisant l'algorithme NSGA-II avec une nouvelle relation de Pareto-dominance. Cette relation est basée sur une nouvelle fonction de pénalité appelée degré de violation de contraintes pour sélectionner les CCS (Composite Cloud Service) pertinents qui respectent les contraintes des clients. La deuxième phase permet de sélectionner la solution optimale parmi les résultats Pareto-optimaux obtenus de la première phase. En effet, chaque solution de l'ensemble du Pareto-

optimale est évaluée par une fonction d'utilité (score) en utilisant la pondération additive simple (SAW) afin de sélectionner la solution ayant le meilleur score. Dans ce cadre, nous avons proposé une nouvelle technique basée sur les contraintes et les préférences du client pour le calcul de la pondération des attributs de qualité de service.

Une extension de notre approche a été proposée basée sur les agents pour améliorer la performance du système en matière de temps d'exécution.

Pour concrétiser notre approche, nous avons développé un prototype en utilisant JAVA et Protégé. Ce prototype permet de générer un processus d'affaire ERP personnalisé qui s'adapte aux différents changements du contexte client via la composition sémantique et dynamique des services Cloud. En outre, plusieurs expérimentations ont été réalisées pour montrer l'efficacité et la précision de l'approche proposée en termes de qualité de solution et du respect des contraintes du client. En effet, la mise en œuvre de cette approche offre de meilleures performances, une sélection et une composition dynamique de services Cloud prenant en compte les paramètres de QoS d'utilité (temps d'exécution, prix, disponibilité, sécurité et fiabilité) les paramètres de QoS longue durée de vie (fiabilité et débit) pour une meilleure flexibilité.

L'étude comparative de notre approche avec d'autres approches mono-objectif et multi-objectif a montré l'efficacité des deux phases de l'algorithme CCS-2S. Elle a montré également que la contribution de degré de violation des contraintes fournit des résultats précises respectant les contraintes non-fonctionnelles du client.

Afin d'améliorer la performance de l'approche proposée, un modèle collaboratif est adopté pour la composition des services Cloud en utilisant les aspects multi-agents. Une étude comparative entre les résultats des approches proposées sans/avec agents a été réalisée. Les résultats d'expérimentations ont montré que la performance de l'aspect multi-agent en termes de temps d'exécution est meilleure que celle de l'approche CCS-2S sans agents.

Finalement, nous constatons que notre travail de thèse peut être enrichi dans les différents axes que nous avons abordés. En effet, le prototype que nous avons développé est une silhouette de plateforme qui peut être finalisée et déployée dans le Cloud. Pour cela, quelques perspectives qui nous semblent pertinentes à court et à long terme peuvent être abordées pour l'amélioration des propositions réalisées et la construction intégrale de la plateforme. Ces perspectives sont :

#### ✓ **Application d'autres stratégies multi-agents**

L'utilisation de la stratégie maître-esclave de l'aspect multi-agent dans notre approche a amélioré clairement la performance du système en matière de temps d'exécution. Cependant, il existe d'autres stratégies multi-agent tels que la stratégie en anneaux et la

stratégie hybride. Dans le futur proche, nous envisageons d'implémenter ces stratégies pour essayer d'obtenir une meilleure performance.

✓ **Assurance de la sécurité de données**

Un service composite engendré par notre approche est une séquence de services Cloud qui communiquent entre eux par des messages contenant les données (entrées/sorties). La sécurité et la confidentialité de ces données nécessitent l'intégration de techniques de sécurité avancées pour assurer la sûreté et la fiabilité des données.

✓ **Utilisation des QoS incertaines**

Dans notre travail, nous avons utilisé les valeurs de QoS agrégées déterministes (c'est-à-dire la moyenne) qui peuvent ne pas refléter avec précision les performances réelles d'un service en raison de la nature stochastique de ses QoS. Par exemple, le temps d'exécution fluctuerait avec le changement de connectivité réseau. Il peut également se fluctuer par d'autres critères de QoS des services composés au moment de l'exécution et ne peut pas répondre correctement aux exigences du client. Les futurs travaux viseront à utiliser cet aspect incertain pour améliorer encore la précision des valeurs de QoS et la précision de la sélection et de la composition des services Cloud.

# Liste des publications personnelles

- **Publication internationale indexée ISI Thomson Reuters**

- ✓ Hamza Reffad and Adel Alti. New approach for optimal semantic-based context-aware cloud service composition for erp. *New Generation Computing*, 36(4) :307–347, 2018.

- **Communications internationales**

- ✓ Hamza Reffad, Adel Alti, and Philippe Roose. Cloud-based semantic platform for dynamic management of context-aware mobile erp applications. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems*, pages 181–188. ACM, 2016.
- ✓ Hamza Reffad, Adel Alti, and Philippe Roose. Composition sémantique et dynamique à base d’agents des services cloud pour erp. In *Proceedings of the 36th International Conference on "Construire les Systèmes d’Information pour la Transformation des Organisations à l’ère de l’Innovation Numérique"*, pages 213–222, INFORSID 2018.

# Bibliographie

- [1] Fleur-Anne Blain. Présentation générale des erp et leur architecture modulaire. <https://fablain.developpez.com/tutoriel/presenterp/>, Mars, 2006.
- [2] Björn Johansson and Pedro Ruivo. Exploring factors for adopting erp as saas. *Procedia Technology*, 9 :94–99, 2013.
- [3] Baoan Li and Mingxing Li. Research and design on the refinery erp and eerp based on soa and the component oriented technology. In *Networking and Digital Society, 2009. ICNDS'09. International Conference on*, volume 1, pages 85–88. IEEE, 2009.
- [4] Christos D Tarantilis, Chris T Kiranoudis, and ND Theodorakopoulos. A web-based erp system for business services and supply chain management : Application to real-world process scheduling. *European Journal of Operational Research*, 187(3) :1310–1326, 2008.
- [5] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. Web services. In *Web Services*, pages 123–149. Springer, 2004.
- [6] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [7] Stuart J Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Malaysia ; Pearson Education Limited,, 2016.
- [8] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning : theory and practice*. Elsevier, 2004.
- [9] Yuhong Yan, Min Chen, and Yubin Yang. Anytime qos optimization over the plangraph for web service composition. In *Proceedings of the 27th annual acm symposium on applied computing*, pages 1968–1975. ACM, 2012.
- [10] Shuiguang Deng, Longtao Huang, Hongyue Wu, and Zhaohui Wu. Constraints-driven service composition in mobile cloud computing. In *Web Services (ICWS), 2016 IEEE International Conference on*, pages 228–235. IEEE, 2016.
- [11] Florian Rosenberg, Philipp Leitner, Anton Michlmayr, Predrag Celikovic, and Schahram Dustdar. Towards composition as a service-a quality of service driven approach. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1733–1740. IEEE, 2009.
- [12] Beniamino Di Martino, Giuseppina Cretella, and Antonio Esposito. Cloud services composition through cloud patterns : a semantic-based approach. *Soft Computing*, 21(16) :4557–4570, 2017.

- [13] Freddy Lecue and Nikolay Mehandjiev. Towards scalability of quality driven semantic web service composition. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 469–476. IEEE, 2009.
- [14] Adel Alti, Sébastien Laborie, and Roose Phillipe. Dynamic semantic-based adaptation of multimedia documents. *Transactions on Emerging Telecommunications Technologies*, 25(2) :239–258, 2014.
- [15] Saied Asghari and Nima Jafari Navimipour. Review and comparison of meta-heuristic algorithms for service composition in cloud computing. *Majlesi Journal of Multimedia Processing*, 4(4), 2016.
- [16] Yujie Yao and Haopeng Chen. Qos-aware service composition using nsga-ii 1. In *Proceedings of the 2nd International Conference on Interaction Sciences : Information Technology, Culture and Human*, pages 358–363. ACM, 2009.
- [17] Chin-Sheng Chen, Wen-Yau Liang, and Hui-Yu Hsu. A cloud computing platform for erp applications. *Applied Soft Computing*, 27 :127–136, 2015.
- [18] N Sasikaladevi and L Arockiam. Genetic approach for service selection problem in composite web service. *International Journal of Computer Applications*, 44(4) :22–29, 2012.
- [19] Mohammad Bagher Karimi, Ayaz Isazadeh, and Amir Masoud Rahmani. Qos-aware service composition in cloud computing using data mining techniques and genetic algorithm. *The Journal of Supercomputing*, 73(4) :1387–1415, 2017.
- [20] Zhen Ye, Xiaofang Zhou, and Athman Bouguettaya. Genetic algorithm based qos-aware service compositions in cloud computing. In *International Conference on Database Systems for Advanced Applications*, pages 321–334. Springer, 2011.
- [21] Li Li, Peng Cheng, Ling Ou, and Zili Zhang. Applying multi-objective evolutionary algorithms to qos-aware web service composition. In *International Conference on Advanced Data Mining and Applications*, pages 270–281. Springer, 2010.
- [22] Hiroshi Wada, Paskorn Champrasert, Junichi Suzuki, and Katsuya Oba. Multiobjective optimization of sla-aware service composition. In *2008 IEEE Congress on Services 2008-Part I*, pages 368–375. IEEE, 2008.
- [23] Heidi A Taboada, Jose F Espiritu, and David W Coit. Moms-ga : A multi-objective multi-state genetic algorithm for system reliability optimization design problems. *IEEE Transactions on Reliability*, 57(1) :182–191, 2008.
- [24] Jun Long and Weihua Gui. An environment-aware particle swarm optimization algorithm for services composition. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4. IEEE, 2009.
- [25] Amin Jula, Zalinda Othman, and Elankovan Sundararajan. A hybrid imperialist competitive-gravitational attraction search algorithm to optimize cloud service composition. In *Memetic Computing (MC), 2013 IEEE Workshop on*, pages 37–43. IEEE, 2013.

- [26] Shahriar Lotfi and Khalil Mowlani. Ica-wss : A qos-aware imperialist competitive algorithm for web service selection. In *International Conference on Informatics Engineering and Information Science*, pages 135–148. Springer, 2011.
- [27] Qiang Yu, Ling Chen, and Bin Li. Ant colony optimization applied to web service compositions in cloud computing. *Computers & Electrical Engineering*, 41 :18–27, 2015.
- [28] Cristina Bianca Pop, Monica Vlad, Viorica Rozina Chifu, Ioan Salomie, and Mihaela Dinsoreanu. A tabu search optimization approach for semantic web service composition. In *Parallel and Distributed Computing (ISPD), 2011 10th International Symposium on*, pages 274–277. IEEE, 2011.
- [29] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, et al. Owl-s : Semantic markup for web services. *W3C member submission*, 22(4), 2004.
- [30] Douglas F Parkhill. Challenge of the computer utility. Addison-Wesley Educational Publishers Inc., US, 1966.
- [31] Luis M Vaquero, Luis Roderio-Merino, Juan Caceres, and Maik Lindner. A break in the clouds : towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1) :50–55, 2008.
- [32] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds : A berkeley view of cloud computing. Technical report, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.
- [33] Cloud computing — wikipédia. [https://fr.wikipedia.org/wiki/Cloud\\_computing](https://fr.wikipedia.org/wiki/Cloud_computing). (Accessed on 09/02/2018).
- [34] James Staten, Ted Schadler, JR Rymer, and Chenxi Wang. Q&a : by 2011, cios must answer the question, why not run in the cloud. *S. Leaver, & A. Viglianti, Interviewers*, 2009.
- [35] Mohammad Hamdaqa and Ladan Tahvildari. Cloud computing uncovered : a research landscape. In *Advances in Computers*, volume 86, pages 41–85. Elsevier, 2012.
- [36] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.
- [37] [https ://aws.amazon.com/fr/what-is-cloud-computing/](https://aws.amazon.com/fr/what-is-cloud-computing/).
- [38] Syed A Ahson and Mohammad Ilyas. *Cloud computing and software services : theory and techniques*. CRC Press, 2010.
- [39] Ce que vous devriez savoir sur le cloud computing (owasp quebec). <https://fr.slideshare.net/PatrickLeclerc/ce-que-vous-devriez-savoir-sur-le-cloud-computing-owasp-quebec>. (Accessed on 12/01/2018).

- [40] la-scurit-et-le-cloud-computing-13-728.jpg (728×546). <https://image.slidesharecdn.com/cloudetscuritconfcqsiv8-11111203651-phpapp02/95/la-scurit-et-le-cloud-computing-13-728.jpg?cb=1321044089>. (Accessed on 09/01/2018).
- [41] F Robert Jacobs et al. Enterprise resource planning (erp)—a brief history. *Journal of Operations Management*, 25(2) :357–363, 2007.
- [42] Alain Courtois, Chantal Martin-Bonnefous, Maurice Pillet, and Maurice Pillet. *Gestion de production*. Les Ed. d’Organisation, 2003.
- [43] Présentation générale des erp et leur architecture modulaire. <https://fablain.developpez.com/tutoriel/presenterp/>. (Accessed on 09/01/2018).
- [44] Erp/crm management - view specifications & details of crm solutions by webcom information technology private limited, bengaluru | id : 6514522348. <https://www.indiamart.com/proddetail/erp-crm-management-6514522348.html>. (Accessed on 09/01/2018).
- [45] Nicolas Degroodt. Opportunités de l’open-source pour les solutions erp. academia.edu.
- [46] JEAN-LOUIS TOMAS. Erp et pgi, comment réussir le changement. *édition DUNOD*, 2005.
- [47] Assma Habadi, Yousra Samih, Khadeejah Almehdar, and Eman Aljedani. An introduction to erp systems : Architecture, implementation and impacts. *International Journal of Computer Applications*, 167(9), 2017.
- [48] Shruti Nagpal, Sunil Kumar Khatri, and PK Kapur. Exploratory study to identify critical success factors penetration in erp implementations. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2014 3rd International Conference on*, pages 1–6. IEEE, 2014.
- [49] Shruti Nagpal, Sunil Kumar Khatri, and Ashok Kumar. Comparative study of erp implementation strategies. In *Systems, Applications and Technology Conference (LISAT), 2015 IEEE Long Island*, pages 1–9. IEEE, 2015.
- [50] Islam K Sowon and Radwan Tahboub. Erp systems critical success factors ict perspective. Citeseer, 2015.
- [51] [https://www.enterpriseinnovation.net/files/whitepapers/top10\\_erp\\_vendors.pdf](https://www.enterpriseinnovation.net/files/whitepapers/top10_erp_vendors.pdf).
- [52] Apps run the world, december 2017. <https://www.appsruntheworld.com/top-10-erp-software-vendors-and-market-forecast/>.
- [53] Sandeep Singhal, Puneet Tandon, and SK Sharma. Hardware and software requirements for implementation of erp in technical education institutes in india.
- [54] Ronald D Kay, William M Edwards, et al. Farm management : Instructor’s manual. Technical report, Iowa State University, Department of Economics, 1999.

- [55] Ardian Indra Gunawan and Kridanto Surendro. Enterprise architecture for cloud-based erp system development. In *Advanced Informatics : Concept, Theory and Application (ICAICTA), 2014 International Conference of*, pages 57–62. IEEE, Bandung, Indonesia, 2014.
- [56] William Noah Schilit. *A system architecture for context-aware mobile computing*. PhD thesis, Columbia University New York, NY, 1995.
- [57] Nick Ryan, Jason Pascoe, and David Morse. Enhanced reality fieldwork : the context aware archaeological assistant. *Bar International Series*, 750 :269–274, 1999.
- [58] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 85–90. IEEE, 1994.
- [59] Nick Ryan. Mcfe metadata elements, version 0.2. *Working document. University of Kent at Canterbury. Kent, UK*, 1997.
- [60] Peter J Brown. Triggering information by context. *Personal Technologies*, 2(1) :18–27, 1998.
- [61] Stephen Fickas, Gerd Kortuem, and Zary Segall. Software organization for dynamic and adaptable wearable systems. In *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, pages 56–63. IEEE, 1997.
- [62] Mathias Weske. Business process management architectures. In *Business Process Management*, pages 333–371. Springer, 2012.
- [63] Arthur R Tenner and Irving J DeToro. *Process redesign : The implementation guide for managers*. Addison-Wesley, 1997.
- [64] Bpmn specification - business process model and notation. <http://www.bpmn.org/>. (Accessed on 09/07/2018).
- [65] Ounsa Roudies, Stéphane Badreau, and Jean-Louis Boulanger. Ingénierie des exigences. méthodes et bonnes pratiques pour construire et maintenir un référentiel. *E-Ti : E-Review in Technologies Information*, (8), 2015.
- [66] Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing : a research roadmap. *International Journal of Cooperative Information Systems*, 17(02) :223–255, 2008.
- [67] Tomas Vitvar, Jacek Kopecký, Maciej Zaremba, and Dieter Fensel. Wsmo-lite : Lightweight semantic descriptions for services on the web. In *ECOWS*, volume 7, pages 77–86, 2007.
- [68] Zafar U Singhera and Abad Ali Shah. Extended web services framework to meet non-functional requirements. In *Workshop proceedings of the sixth international conference on Web engineering*, page 21. ACM, New York, USA, 2006.
- [69] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C recommendation*, 26 :19, 2007.

- [70] D Fensel and C Bussler. The web service modeling framework wsmf. *eletronic commerce : Research and applications*, vol. 1.(2002). 113-137 flickr, 2002.
- [71] Steve Battle, Abraham Bernstein, Harold Boley, Benjamin Grosf, Michael Gruninger, Richard Hull, Michael Kifer, David Martin, Sheila McIlraith, Deborah McGuinness, et al. Semantic web services framework (swsf) overview. *World Wide Web Consortium, Member Submission SUBM-SWSF-20050909*, 2005.
- [72] Jyotishman Pathak, Samik Basu, and Vasant Honavar. Modeling web services by iterative reformulation of functional and non-functional requirements. In *International Conference on Service-Oriented Computing*, pages 314–326. Springer, 2006.
- [73] Dumitru Roman, James Scicluna, Dieter Fensel, Axel Polleres, Jos de Bruijn, and Stijn Heymans. D14v0. 3. ontology-based choreography of wsmo services. WSMO Final Draft 19th May, 2006.
- [74] Youakim Badr, Ajith Abraham, Frédérique Biennier, and Crina Grosan. Enhancing web service selection by user preferences of non-functional features. In *Next Generation Web Services Practices, 2008. NWESP'08. 4th International Conference on*, pages 60–65. IEEE, Seoul, South Korea, 2008.
- [75] Ioan Toma, Kashif Iqbal, Matthew Moran, Dimitru Roman, Thomas Strang, and Dieter Fensel. An evaluation of discovery approaches in grid and web services environments. *Proceedings of the Net. ObjectDays 2005*, 2005.
- [76] Alessio Bosca, Andrea Ferrato, Fulvio Corno, Ilenia Congiu, and Giuseppe Valetto. Composing web services on the basis of natural language requests. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, Orlando, FL, USA, 2005.
- [77] Marcel Cremene, Jean-Yves Tigli, Stéphane Lavirotte, Florin-Claudiu Pop, Michel Riveill, and Gaëtan Rey. Service composition based on natural language requests. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 486–489. IEEE, Bangalore, India, 2009.
- [78] Alexander Lazovik, Marco Aiello, and Rosella Gennari. Encoding requests to web service compositions as constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 782–786. Springer, 2005.
- [79] Erman Karakoc and Pinar Senkul. Composing semantic web services under constraints. volume 36, pages 11021–11029. Elsevier, 2009.
- [80] Stefano Ceri, Sara Comai, Ernesto Damiani, Piero Fraternali, Stefano Paraboschi, and Letizia Tanca. Xml-gl : a graphical language for querying and restructuring xml documents1. *Computer networks*, 31(11-16) :1171–1187, 1999.
- [81] Jos De Bruijn, Holger Lausen, Axel Polleres, and Dieter Fensel. The web service modeling language wsml : an overview. In *European Semantic Web Conference*, pages 590–604. Springer, 2006.

- [82] Richard Fikes, Patrick Hayes, and Ian Horrocks. Owl-ql—a language for deductive query answering on the semantic web. *Web semantics : Science, services and agents on the World Wide Web*, 2(1) :19–29, 2004.
- [83] Tom Bellwood, Luc Clément, David Ehnebuske, Andrew Hatelly, Maryann Hondo, Yin Leng Husband, Karsten Januszewski, Sam Lee, Barbara McKee, Joel Munter, et al. Uddi version 3.0. *Published specification, Oasis*, 5 :16–18, 2002.
- [84] Ws index. <https://wsindex.org/>. (Accessed on 06/27/2018).
- [85] Henry Song, Doreen Cheng, Alan Messer, and Swaroop Kalasapur. Web service discovery using general-purpose search engines. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 265–271. IEEE, 2007.
- [86] Bo Zhou and Tinglei Huang. Semantic web service discovery search with ontology learning. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 5, pages 1048–1051. IEEE, 2008.
- [87] Hong-Jie Guan, Fan-Rong Meng, Jin-Fei Sun, and Peijun Du. Web service discovery based on the cooperation of uddi and df. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pages 1–4. IEEE, Dalian, China, 2008.
- [88] Heather Kreger et al. Web services conceptual architecture (wsca 1.0). *IBM software group*, 5(1) :6–7, 2001.
- [89] Yiqiao Wang and Eleni Stroulia. Flexible interface matching for web-service discovery. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 147–156. IEEE, Rome, Italy, 2003.
- [90] Aliaksandr Birukou, Enrico Blanzieri, Vincenzo DAndrea, Paolo Giorgini, and Natallia Kokash. Improving web service discovery with usage data. *IEEE software*, 24(6), 2007.
- [91] Wei Jiang, Charles Zhang, Zhenqiu Huang, Mingwen Chen, Songlin Hu, and Zhiyong Liu. Qsynth : A tool for qos-aware automatic service composition. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 42–49. IEEE, Miami, FL, USA, 2010.
- [92] Kaijun Ren, Jinjun Chen, Nong Xiao, Weimin Zhang, and Junqiang Song. A qsql-based collaboration framework to support automatic service composition and workflow execution. In *Grid and Pervasive Computing Workshops, 2008. GPC Workshops' 08. The 3rd International Conference on*, pages 87–92. IEEE, Kunming, China, 2008.
- [93] Xiaogang Ji. Research on web service discovery based on domain ontology. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 65–68. IEEE, Beijing, China, 2009.
- [94] George Spanoudakis, Khaled Mahbub, and Andrea Zisman. A platform for context aware runtime web service discovery. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 233–240. IEEE, Salt Lake City, UT, USA, 2007.

- [95] Wenge Rong, Kecheng Liu, and Lin Liang. Association rule based context modeling for web service discovery. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, pages 299–304. IEEE, Washington, DC, USA, 2008.
- [96] Howard Foster, Sebastian Uchitel, Jeff Magee, and Jeff Kramer. Model-based verification of web service compositions. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pages 152–161. IEEE, Montreal, Que., Canada, 2003.
- [97] Waqar Sadiq and Felix Racca. *Business services orchestration : The hypertier of information technology*. Cambridge University Press, 2003.
- [98] Alistair Barros, Marlon Dumas, and Phillipa Oaks. A critical overview of the web services choreography description language. *BPTrends Newsletter*, 3 :1–24, 2005.
- [99] Jun Li, Xiao-Lin Zheng, Song-Tao Chen, William-Wei Song, and De-ren Chen. An efficient and reliable approach for quality-of-service-aware service composition. volume 269, pages 238–254. Elsevier, 2014.
- [100] Faisal N Abu-Khzam, Cristina Bazgan, Joyce El Haddad, and Florian Sikora. On the complexity of qos-aware service selection problem. In *International Conference on Service-Oriented Computing*, pages 345–352. Springer, 2015.
- [101] Hamed Rezaie, Naser NematBaksh, and Farhad Mardukhi. A multi-objective particle swarm optimization for web service composition. In *International Conference on Networked Digital Technologies*, pages 112–122. Springer, 2010.
- [102] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2) :182–197, 2002.
- [103] C Ranganathan, Thompson SH Teo, and Jasbir Dhaliwal. Web-enabled supply chain management : Key antecedents and performance impacts. *International Journal of Information Management*, 31(6) :533–545, 2011.
- [104] Kamala Gollakota. Ict use by businesses in rural india : The case of eid parry’s indiagriline. *International Journal of Information Management*, 28(4) :336–341, 2008.
- [105] KeJing Zhang, PingJun Dong, Biao Ma, BingYong Tang, and Hong Cai. Innovation of it service in textile industrial clusters from the service system perspective. In *Logistics Systems and Intelligent Management, 2010 International Conference on*, volume 3, pages 1819–1822. IEEE, Harbin, China, 2010.
- [106] Monika Mital, Ashis Pani, and Ram Ramesh. Determinants of choice of semantic web based software as a service : An integrative framework in the context of e-procurement and erp. *Computers in Industry*, 65(5) :821–827, 2014.

- [107] Hamza Reffad, Adel Alti, and Philippe Roose. Cloud-based semantic platform for dynamic management of context-aware mobile erp applications. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems*, pages 181–188. ACM, Biarritz, France, 2016.
- [108] Hamza Reffad and Adel Alti. New approach for optimal semantic-based context-aware cloud service composition for erp. *New Generation Computing*, 36(4) :307–347, 2018.
- [109] Ravi Khurana and Rajesh Kumar Bawa. Quality based cloud service broker for optimal cloud service provider selection. *International Journal of Applied Engineering Research*, 12(18) :7962–7975, 2017.
- [110] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook : Theory, implementation and applications*. Cambridge university press, 2003.
- [111] Eric Prud, Andy Seaborne, et al. Sparql query language for rdf. Available : <http://www.w3.org/TR/rdf-sparql-query/>, 2006.
- [112] World Wide Web Consortium et al. A semantic web rule language combining owl and ruleml. <http://www.w3.org/Submission/SWRL>, 2004.
- [113] Thomas Wöhling, Jasper A Vrugt, and Gregory F Barkle. Comparison of three multiobjective optimization algorithms for inverse modeling of vadose zone hydraulic properties. *Soil Science Society of America Journal*, 72(2) :305–319, 2008.
- [114] Karim Tari, Yacine Amirat, Abdelghani Chibani, Ali Yachir, and Abdelhamid Mel-louk. Context-aware dynamic service composition in ubiquitous environment. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE, Cape Town, South Africa, 2010.
- [115] Hamza Reffad, Adel Alti, and Philippe Roose. Composition sémantique et dynamique à base d’agents des services cloud pour erp. In *Proceedings of the 36th International Conference on "Construire les Systèmes d’Information pour la Transformation des Organisations à l’ère de l’Innovation Numérique"*, pages 213–222. Nantes, France, 2018.
- [116] Welcome to netbeans. <https://netbeans.org/>. (Accessed on 07/20/2018).
- [117] protégé. <https://protege.stanford.edu/>. (Accessed on 07/20/2018).

## Résumé

Actuellement, la technologie Cloud a été largement adoptée par les entreprises pour développer des solutions informatiques de qualité. En effet, les Petites et les Moyennes Entreprises (PME) sont à la recherche d'ERP (Enterprise Resource Planning) sur mesure afin d'automatiser judicieusement leurs activités commerciales. Notre contribution se décline en trois volets : le premier volet définit une nouvelle ontologie pour la description sémantique des services Cloud hétérogènes issues de différents fournisseurs avec différentes QoS. Notre ontologie est spécialisée à la personnalisation des applications de type ERP sensible aux contextes. Le deuxième volet est une nouvelle approche orientée Cloud sémantique multi-objective à deux phases qui intègre la sélection et la composition optimale des services pertinents selon les contraintes du client. Une extension de cette approche avec les aspects d'agents est également proposée. Le troisième volet est un prototype pour l'élaboration automatique et efficace d'un processus d'affaire ERP avec une meilleure flexibilité selon les besoins du client et son contexte.

**Mots clés :** ERP, Cloud, ontologie, composition des services, QoS, contexte, NSGA-II, optimisation, multi-agent.

## Abstract

Nowadays, Cloud technology is widely adopted by companies to develop quality IT solutions. Small and Medium Enterprises (SMEs) are looking for the best preeminent customized Enterprise Resource Planning (ERP) to automate wisely their business activities. Our contribution is situated around three axes : the first axe defines a new ontology which is used for semantic description of heterogeneous Cloud services of different providers with different QoS. Our proposed ontology is used for customizing context-aware ERP applications. The second axe is a new two-phase multi-objective semantic Cloud oriented approach that integrates the selection and the optimal composition of the relevant services according to the client's constraints. An extension of this approach based on the multi-agent aspect is also proposed. The third axe is a prototype for automatic and efficient generation of an ERP business process according to customer needs and his current context with high flexibility.

**Keywords :** ERP, Cloud, ontology, service composition, QoS, context, NSGA-II, optimization, multi-agent.

## ملخص

حاليا، التكنولوجيا السحابية معتمدة على نطاق واسع من قبل الشركات لتطوير حلول الحوسبة عالية الجودة. من ناحية أخرى، تبحث الشركات الصغيرة ومتوسطة الحجم (SMEs) عن برامج تخطيط الموارد المصممة خصيصا من أجل التسيير الآلي لعملياتها التجارية بحكمة. تنقسم مساهمتنا إلى ثلاثة أجزاء: الجزء الأول يتعلق بتعريف الهيكل الدلالي (أنطولوجيا) الجديد لتعريف الوصف الدلالي للخدمات السحابية غير المتجانسة القادمة من مزودين مختلفين مع مختلف جودة الخدمة، ومهمته هو تخصيص تطبيقات تخطيط موارد المؤسسات الحساسة للسياق. والمكون الثاني هو منهج جديد موجه للسحابة الدلالية متعدد الأغراض يتألف من مرحلتين ودمج الاختيار والتكوين الأمثل للخدمات ذات الصلة. ويقترح أيضا تمديدا لهذا النهج باستخدام الجانب متعدد الأعوان (agents). العنصر الثالث هو نموذج أولي للتوليد الآلي والفعال لعملية تخطيط موارد المؤسسات وفقا لاحتياجات الزبون وسياقه مع مرونة عالية.

### كلمات البحث:

تخطيط موارد المؤسسات، السحابة (Cloud)، الهيكل الدلالي (أنطولوجيا)، تركيب الخدمات، جودة الخدمة (QoS)، NSGA-II، السياق، التحسين إلى الأمثل، متعدد الأعوان (multi-agent).