Contributions au déploiement sécurisé de processus métiers dans le cloud

Amina Ahmed Nacer

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Contributions au déploiement sécurisé de processus métiers dans le cloud

THÈSE

présentée et soutenue publiquement le 26 Février 2019

pour l'obtention du

Doctorat de l'Université de Lorraine en cotutelle avec l'Université de Bejaia

(mention informatique)

par

Amina Ahmed Nacer

Composition du jury

Président :	Zizette Boufaida	Professeur, Université de Constantine, Algérie
Rapporteurs :	Daniela Grigori	Professeur, Université Paris Dauphine
	Hassina Nacer	Maitre de conférences, USTHB Alger
Examinateurs :	Salima Benbernou	Professeur, Université Paris Descartes
	Aloui Abdelouahab	Maitre de conférences, Université de Bejaia
Encadrants :	Claude Godart	Professeur, Université de Lorraine
	Samir Youcef	Maitre de conférences,Université de Lorraine
	Abdelkamel Tari	Professeur,Université de Bejaia

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens à remercier du fond du cœur toutes les personnes qui m'ont soutenu et aidé pour la réalisation de cette thèse.

En premier lieu, je tiens à remercier mes encadrants les professeurs Abdelkamel Tari et Claude Godart pour m'avoir encadré et accompagné tout au long de ce projet de thèse. Merci pour tout le temps que vous m'avez accordé. Je remercie également mon co-encadrant Samir Youcef pour sa disponibilité et ses conseils avisés.

Je souhaiterais exprimer ma sincère gratitude envers tous les membres du jury pour avoir accepté d'évaluer ce travail de thèse.

Je remercie aussi l'ensemble du personnel du laboratoire LIMED pour leur gentillesse et leur bonne humeur, en particulier Sarah et Ryma.

Je tiens également à remercier les collègues du laboratoire LORIA-INRIA GrandEST pour les agréables moments que j'ai passé en leur compagnie. Un merci particulier à tous les membres de l'équipe COAST que j'ai eu la chance de connaitre pendant mes séjours au LORIA. Merci pour tous ces déjeunés et cafés animés.

Je remercie également mes meilleures amies Besma et Sarah qui m'ont toujours soutenu.

Un merci particulier à Riad qui m'a toujours soutenu et guidé tout au long de cette thèse.

Un ultime remerciement à mes parents, mon frère Abdi ainsi que ma sœur Édina. Rien de tout cela n'aurait été possible sans vous.

Résumé

L'évolution et l'accroissement actuels des technologies amènent les entreprises à vouloir se développer plus rapidement afin de rester compétitives et offrir des services à la pointe de la technologie, répondant aux besoins du marché. En effet, les entreprises étant sujettes à des changements assez fréquents requièrent un haut niveau de flexibilité et d'agilité. La gestion des processus métiers (BPM) leur permet dans ce sens de mieux appréhender et gérer leurs processus. Par ailleurs, l'apparition du Cloud Computing et de tous ses bénéfices (flexibilité et partage, coût optimisé, accessibilité garantie...etc) le rendent particulièrement attrayant. Ainsi, l'association de ces deux concepts permet aux entreprises de renflouer leur capital. Cependant, l'utilisation du cloud implique également de nouvelles exigences en terme de sécurité, qui découlent de son environnement partagé, et qui mettent un frein à sa large adoption.

Le travail de cette thèse consiste à proposer des concepts et outils pour aider et guider les entreprises dans le déploiement de leurs processus dans un environnement cloud en toute sécurité.

Une première contribution est un algorithme d'obfuscation permettant d'automatiser la décomposition et le déploiement des processus sans intervention humaine, en se basant sur la nature des fragments. Cet algorithme limite le taux d'informations sur chaque cloud à travers un ensemble de contraintes de séparation, permettant de déployer les fragments considérés comme étant sensibles sur différents clouds.

La seconde contribution de cette thèse consiste à complexifier la structure du processus afin de limiter le risque de coalition de clouds. Ceci se fait à travers l'introduction de faux fragments à certains endroits stratégiques du processus. L'objectif étant de rendre les collaborations générées plus résistantes aux attaques, et par conséquent de réduire la probabilité de coalition.

Même si les opérations d'obfuscation et de complexification protègent le savoir-faire des entreprises lors d'un déploiement cloud, un risque subsiste toujours. Dans ce contexte, cette thèse propose également un modèle de risque permettant d'évaluer et de quantifier les risques de sécurité auxquels restent exposés les processus après déploiement. L'objectif de ce modèle est de combiner les informations de sécurité avec d'autres dimensions de la qualité de service tel que le coût, pour la sélection de configurations optimisées.

Les approches proposées sont implémentées et testées à travers différentes configurations de processus. Leur validité est vérifiée à travers un ensemble de métriques dont l'objectif est de mesurer la complexité des processus après l'opération d'obfuscation ainsi que le niveau de risque subsistant.

Mots-clés: Cloud Computing, Processus Métiers, Obfuscation, Gestion du risque.

Abstract

The fast evolution and development of technologies lead companies to grow faster in order to remain competitive and to offer services which are at the cutting edge of technology, meeting today's market needs. Indeed, companies that are subject to frequent changes require a high level of flexibility and agility. Business Process Management (BPM) allows them to better manage their processes. Moreover, the emergence of Cloud Computing and all its advantages (flexibility and sharing, optimized cost, guaranteed accessibility ... etc) make it particularly attractive. Thus, the combination of these two concepts allows companies to refloat their capital. However, the use of the cloud also implies new requirements in term of security, which stem from its shared environment, and which slow down its widespread adoption.

The objective of this thesis consists in proposing concepts and tools that help and guide companies to deploy safely their processes in a cloud environment.

A first contribution is an obfuscation algorithm that automates the decomposition and deployment of processes without any human intervention, based on the nature of the fragments. This algorithm limits the rate of information on each cloud through a set of separation constraints, which allow to deploy fragments considered as sensitive on different clouds.

The second contribution of this thesis consists in complicating the structure of the process in order to limit the risk of clouds coalition. This is done through the introduction of fake fragments at certain strategic points in the process. The goal is to make generated collaborations more resistant to attacks, and thus reducing the likelihood of coalition.

Even if obfuscation and complexification operations protect companies' know-how during a cloud deployment, a risk remains. In this context, this thesis also proposes a risk model for evaluating and quantifying the security risks to which the process remain exposed after deployment. The purpose of this model is to combine security information with other dimensions of quality of service such as cost, for the selection of optimized configurations.

The proposed approaches are implemented and tested through different process configurations. Their validity is verified through a set of metrics, whose objective is to measure the complexity of the processes as well as the remaining risk level after obfuscation.

Keywords: Cloud Computing, Business Processes, Obfuscation, Risk management.

Je dédie cette thèse
à mes parents.

Table des matières

Table des figures

Introduction générale

L'évolution sans cesse croissante des besoins en informatique et la nécessité de communication entre les différents acteurs associés à plusieurs domaines a vu l'émergence de nouvelles technologies de l'information et de la communication (TIC). Celles-ci offrent alors plusieurs possibilités de coopération, de coordination et de communication facilitant la circulation de l'information sous toutes ses formes, et faisant évoluer de nouvelles méthodes de travail, permettant aux entreprises une croissance exponentielle.

Un des piliers de ces innovations est l'utilisation des processus métiers, qui sont devenus indispensables pour mieux organiser et gérer les différentes tâches d'une entreprise. Leur gestion inclut les concepts, les méthodes et les techniques nécessaires pour la conception, l'administration, la configuration, l'exécution et l'analyse de ces processus [Bao13]. Elle permet ainsi d'accroitre l'efficacité des processus organisationnels à travers de nouvelles stratégies de gestion et d'amélioration des performances d'une entreprise. Ceci se fait en analysant et adaptant régulièrement ses processus, workflows et procédures en fonction des contraintes identifiées pour développer ses performances ou sa qualité de service [VDATHW03].

L'introduction du cloud computing, abrégé en cloud (le « Nuage »en français) a également permis une révolution du monde de l'informatique. Cette technologie dite « en nuage » consiste à exploiter la puissance de calcul et/ou stockage de serveurs informatiques distants à travers un réseau, généralement Internet. Ces serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon des critères techniques (puissance, bande passante, etc.) mais également au forfait. Le cloud computing se caractérise par sa grande souplesse : selon le niveau de compétence de l'utilisateur client, il est possible de gérer soi-même son serveur ou de se contenter d'utiliser des applications distantes en mode SaaS [1].

Outre les services qu'il offre en termes de puissances, stockage, etc, le cloud offre les platformes sur lesquelles peuvent être exécutées les applications clientes. En effet, le fournisseur cloud maintient la plate-forme d'exécution de ces applications : le matériel du/des serveurs (la carte mère, sa mémoire vive. . .). Par ailleurs, il gère également l'infrastructure informatique permettant aux clients de bénéficier de ressources informatiques virtualisées (serveurs, connexions réseau, bande passante) à la demande, facturées en fonction de la quantité des ressources consommées.

Étant donné que les besoins des entreprises sont sujets à des changements assez fréquents, l'agilité et la flexibilité deviennent donc des critères essentiels permettant de gagner en compétitivité, en s'adaptant le plus rapidement possible aux changements du marché. Alors que la gestion des processus métiers BPM «Business Process Management» permet aux entreprises de mieux appréhender et gérer leurs processus métiers, le cloud computing

1. Software as a Service

leur permet un gain considérable en terme de coût et de temps quant à l'exécution de leur processus métiers, leur permettant par conséquent de rester compétitives au sein du marché. L'association de ces deux concepts leur permet de renflouer leur capital. En effet, réutiliser des fragments de processus comme un service i.e. BPaaS [2] évite aux entreprises d'une part de construire un processus de zéro, et leur permet d'autre part de se concentrer sur le cœur de leur processus en évitant toutes autres distractions (qui seront donc gérées par le cloud). Par conséquent, ceci leur offre la possibilité de rester compétitives en offrant un haut degré d'efficacité. De plus, l'utilisation du cloud offre une certaine flexibilité des processus et donne accès à un vaste ensemble de ressources partagées. En effet, les services utilisés peuvent être ajustés à tout moment en fonction des besoins et de l'activité de l'entreprise.

Cependant, et malgré tous les avantages qu'elle offre, la technologie cloud rencontre de sérieux problèmes de sécurité qui sont un frein à sa large adoption. En effet, étant un environnement partagé entre plusieurs utilisateurs, le cloud expose les informations sensibles de ses clients à de sérieux risques de divulgation. Par conséquent, les entreprises sont prêtes à utiliser la technologie cloud pour profiter de ses avantages, mais veulent aussi être certains que le savoir faire inclus dans leurs processus reste protégé.

Il est ainsi nécessaire de trouver un moyen efficace permettant de protéger le savoir-faire de ces entreprises au niveau logique i.e. préserver la logique de fonctionnement et de prise de décision du processus avant déploiement dans le cloud. En effet, le niveau logique exprime la synchronisation des tâches à exécuter afin d'atteindre l'objectif du processus. Par conséquent, il est primordial de protéger sa structure au niveau logique afin de permettre la préservation du savoir faire des entreprises.

Durant la dernière décennie, et dans l'optique de sécuriser au maximum les informations incluses dans les processus métiers, plusieurs travaux de recherches ont été menés [DPdSS14] [CDLLRVDA13]. Néanmoins, les approches proposées sont prioritairement concernées par la préservation des différentes propriétés d'un processus métier au niveau configuration, et n'interviennent pas au niveau *logique*. D'autres approches concernées par la préservation des processus métiers ont été proposées [Fil12] [BBDA12]. Cependant, elles génèrent des processus obfusqués dans un contexte de partage et d'analyse sans considération de l'aspect sécurité.

Le principal objectif de cette thèse est de fournir aux entreprises des moyens de garantir la préservation du savoir faire de leurs processus métiers lors d'un déploiement dans le cloud.

Dans cet objectif, nous proposons :

1. Une première contribution qui consiste en la définition d'un algorithme d'obfuscation [GANYG15] qui génère un ensemble de contraintes (*separation/co-localisation*) pour l'ensemble des tâches du processus. L'efficacité de cet algorithme est mesurée à travers l'introduction d'une métrique d'obfuscation proposée à partir de l'état de l'art [NGY+18], et qui mesure le niveau de complexité des collaborations générées par cet algorithme.

2. La deuxième contribution consiste à complexifier le modèle de processus avant déploiement dans le cloud. Ceci se fait en introduisant de faux fragments à certains endroits considérés comme *stratégiques* [NGY+16], pour rendre les collaborations générées plus résistantes aux attaques.

2. Business Process as a Service

3. La troisième contribution consiste à mesurer le niveau de risque auquel reste exposée une collaboration après application des deux approches citées précédemment. Le niveau de risque représente le niveau de facilité qu'un cloud malicieux a, pour former une coalition avec les autres clouds de la collaboration [NGYT17]. Cette coalition ayant pour objectif de reconstruire le savoir-faire du processus.

4. La quatrième contribution consiste à sélectionner les clouds de la collaboration qui permettent d'exécuter le processus métier dans un environnement cloud. Cette approche de sélection a pour objectif d'assurer un compromis entre les risques de sécurité et le coût générés par le déploiement du processus.

Organisation du document

La présente thèse est divisée en trois grandes parties.

La première partie définit la problématique de notre travail et décrit brièvement nos contributions. Elle propose aussi un survol des notions de base nécessaires à la compréhension de cette thèse, ainsi qu'une étude bibliographique des travaux traitant cette problématique. Elle souligne également les limites de ces travaux et présente les apports de notre travail par rapport à ces limites.

La deuxième partie constitue le cœur de notre travail et concerne les solutions que nous proposons pour résoudre la problématique posée, ainsi que les différentes expérimentations menées au cours de cette thèse dans le but de valider les approches proposées.

La troisième partie est dédiée aux bilans et aux perspectives.

L'organisation détaillée du manuscrit suit le schéma suivant :

- Dans le **chapitre 1**, nous présentons la problématique de notre travail : comment permettre aux entreprises de profiter des avantages qu'apporte la technologie cloud computing, tout en assurant la préservation du savoir-faire de leur processus métiers. Cette problématique est introduite à travers quatre questions de recherche et est illustrée via un exemple de motivation. Nous synthétisons par la suite l'ensemble des travaux de départ sur lesquels se base notre travail de recherche, et résumons l'ensemble de nos contributions introduites tout au long de cette thèse.

- Dans le **chapitre 2**, nous présentons les concepts et définitions nécessaires à la compréhension de cette thèse. Ces définitions concernent les trois principaux axes de recherche à savoir : *le cloud computing, les processus métiers* et *l'obfuscation*. Nous présentons également un état de l'art autour des problèmes de sécurité rencontrés par l'adoption du cloud computing, la sélection de ses services, l'externalisation des processus métiers dans le cloud, et enfin les travaux effectués autour de l'obfuscation des processus métiers.

- Dans le **chapitre 3**, nous présentons en premier lieu notre algorithme d'obfuscation et introduisons la notion de *tâche sensible* sur laquelle se base nos travaux de thèse. Nous introduisons par la suite notre approche de complexification, qui consiste en l'introduction de faux fragments à certains endroits *stratégiques* du processus. Ces fragments ont pour objectif d'augmenter la taille et le nombre de chemins séparant les fragments sensibles, et donc réduire la probabilité d'une éventuelle coalition. Par ailleurs, ce chapitre introduit également un ensemble de métriques (proposées

au cours de cette thèse) pour mesurer l'efficacité des approches proposées. Ces métriques basées sur l'état de l'art, permettent de mesurer le niveau de complexité des collaborations obtenues suite à notre algorithme d'obfuscation.

- Dans le **chapitre 4**, nous présentons l'approche adoptée pour le calcul du niveau de risque auquel est exposée une collaboration de fragments de processus lors d'un déploiement cloud. Cette approche se base principalement sur le calcul des niveaux de risques auxquels sont exposées toutes les tâches sensibles du processus. Ces derniers représentent le niveau de difficulté auquel est confronté le cloud déployant une tâche sensible afin de conspirer avec tous les clouds le séparant du cloud qui déploie sa tâche complémentaire. Ainsi, nous présentons notre modèle de risque qui calcule le risque auquel sont exposées les tâches sensibles en premier lieu, et auquel est exposée la collaboration dans un second temps. Nous introduisons par la suite à partir de notre exemple de motivation, l'exemple d'une tâche sensible et calculons à travers différentes étapes, les différents niveaux de risque auxquels elle est exposée sur les différents chemins la reliant à sa tâche complémentaire. Enfin, nous introduisons l'ensemble des expérimentations effectuées afin de vérifier l'efficacité et l'applicabilité de notre approche.

- Dans **le chapitre 5**, nous présentons nos approches de sélection des clouds pour l'exécution d'une collaboration. Ces approches ont pour objectif de respectivement minimiser le risque auquel une collaboration est exposée en contraignant le coût à un seuil défini, et de minimiser son coût de déploiement et d'exécution en limitant son niveau de risque. Les heuristiques développées se basent sur des modèles de risque et de coût, ainsi que sur un ensemble de contraintes qui peuvent être définies par l'utilisateur selon ses besoins. Leur efficacité est testée à travers l'introduction de solutions optimales, permettant d'évaluer la validité des résultats obtenus.

- Enfin, nous rappelons dans le **le chapitre 6** les objectifs de la thèse, résumons nos différentes contributions et présentons les perceptives directes en rapport avec nos travaux de recherche.

Première partie

Problématique et état de l'art

Chapitre 1

Problématique et Motivations

Sommaire

Ce chapitre illustre la problématique traitée au cours de cette thèse en mettant en exergue les avantages d'utiliser les architectures cloud, mais aussi les problèmes de sécurité liés à ces dernières. Nous présenterons par la suite un résumé de nos différentes contributions, illustrées par un exemple de motivation.

1.1 Problématique

L'avènement des nouvelles technologies ainsi que l'introduction du *cloud computing* apportent de nouvelles solutions qui augmentent la compétitivité des entreprises. En effet, afin de rester à la pointe du marché actuel, ces dernières doivent garantir un haut degré d'efficacité, tout en respectant les délais de livraison.

De plus, ces entreprises doivent supporter un important degré de flexibilité quant aux changement des besoins, lois, régulations, et directives internes, afin d'assurer un haut degré de conformité [DCD+09] [SLM+10]. Pour ce faire, une idée essentielle est de développer et d'implémenter des processus métiers (Business Process) basés sur les nouvelles technologies du cloud pour répondre au mieux aux exigences du marché.

Ces solutions basées sur le cloud, donnent aux entreprises l'opportunité d'éviter les coûts initiaux d'infrastructures, mais les aident également à se concentrer sur leurs activités commerciales au lieu de leur infrastructure système [JSGI09].

Parmi ces solutions, l'externalisation des données des entreprises, et/ou l'exécution partielle ou complète de leurs processus dans le cloud leur permet de se concentrer au mieux sur leurs activités commerciales/stratégiques. Les fragments externalisés se trouvent dans des répertoires BPaaS [3] et peuvent éventuellement être réutilisés par d'autres entreprises pour le développement de leur processus métiers ("ces fragments peuvent être vus comme des groupes connectés d'éléments de processus ayant un haut potentiel de ré-utilisabilité" [BBDA12]). Ceci permet ainsi, de réduire non seulement le coût et le temps de développement (éviter de construire un processus de zéro), mais aussi d'améliorer la robustesse. [SLM+10].

Les entreprises se basaient déjà traditionnellement sur des processus inter-organisationnels, et particulièrement via les architecture SaaS [4], où les différents acteurs de la collaboration étaient connus et bien définis. L'externalisation des fragments de processus dans le cloud quant à elle ne fait pas l'hypothèse de partenaires connus, et donc représente une coopération plus risquée.

Utiliser ou externaliser des fragments à distance sur une architecture cloud, par définition un environnement partagé, induit de nouveaux problèmes de sécurité et expose les systèmes d'information des entreprises à de nouveaux risques. En effet, en externalisant leurs BP fragments, les entreprises perdent le contrôle sur ces fragments. En conséquence, *le savoir-faire de l'entreprise contenu dans ces fragments est exposé au niveau du cloud, exposant ainsi sa stratégie de prise de décisions à des risques de divulgation.*

De plus, externaliser des fragments de processus sur le cloud peut induire un coût considérable, qui ne se justifie pas forcement en terme de qualité de service QoS, e.g. la sécurité. En effet, les configurations sélectionnées pour le déploiement du process (tâches/cloud) peuvent avoir un coût important, qui d'une part ne rentre pas dans le budget de l'organisation cliente, et d'autre part expose le processus à des risques de sécurité.

C'est dans cette optique que s'inscrit notre travail de thèse, qui vise à apporter aux entreprises les garanties nécessaires afin de pouvoir déployer/réutiliser des fragments de processus pour profiter des avantages du cloud en toute sécurité. Cette problématique nous amène à répondre à cinq questions de recherche principales :

Q1 : Comment automatiser le déploiement des différents fragments sur une architecture cloud en assurant la préservation du savoir faire du processus ? Cette question nous amène à définir en premier lieu les différents types de fragments (sensibles [5] ou pas). En effet, bien que les concepteurs soient capables de définir ces fragments là en utilisant leurs connaissances du processus, automatiser cette tâche, c'est-à-dire être capable de caractériser syntaxiquement de tels fragments s'avère être plus difficile. Ces différents fragments seront une fois identifiés, distribués sur un ensemble de clouds selon leur nature, en se basant sur un ensemble de contraintes générées par notre algorithme d'obfuscation. Cette question est abordée dans le chapitre 3.

Q2 : Comment augmenter et mesurer le niveau d'obfuscation d'une collaboration ? Afin de répondre à cette question, nous proposons des solutions pour complexifier un modèle de processus avant déploiement dans le cloud (dans l'objectif de

3. Business Process As a Service
4. Software As a Service
5. les fragments qui contiennent d'importantes informations quant à la logique de fonctionnement du processus

le protéger contre d'éventuelles coalitions de clouds appartenant à la collaboration). Ceci se fait en se basant sur l'introduction de faux fragments à certains endroits stratégiques du processus. L'approche proposée permet d'optimiser cette complexification tout en répondant à certains besoins fonctionnels/non fonctionnels optimisant la qualité de service. Par ailleurs, cette question nous amène également à étudier les différentes mesures de complexité déjà existantes, et à voir si l'on peut les adapter à notre contexte, puis à proposer des mesures plus adaptées. Ces différentes questions sont également abordée dans le chapitre 3.

Q3 : Comment lutter contre la coalition de clouds ? Ou plus précisément, comment mesurer à quel point le savoir-faire du processus est exposé à des risques de divulgation sur une configuration (tâches/cloud), en tenant compte de la possibilité de coalition de clouds ? Ce niveau de risque est principalement basé sur les niveaux de risques auxquelles sont exposées les tâches sensibles du processus. Nous répondons à cette question en proposant une approche de calcul du risque basée sur des données côté client. Cette approche est détaillée dans le chapitre 4.

Q4 : Comment optimiser le déploiement du processus métier en terme de sécurité et de coût ? Où plutot comment sélectionner la configuration (tâches/cloud) qui permet d'avoir un compromis entre la sécurité procurée et le coût engendré ? Nous répondons à cette question à travers l'introdution d'heuristiques dont l'objectif est de respectivement minimiser le risque de déploiement en respectant un seuil de coût et vice vesra. Cette question est abordée dans le chapitre 5.

1.2 Exemple de motivation

Afin de mieux expliciter les problèmes auxquels sont exposées les entreprises lors du déploiement de leur processus métiers dans le cloud, nous allons utiliser l'exemple décrit dans [GANYG15]. Il s'agit d'un procédé exécuté par une banque en vue de déterminer si la demande de prêt bancaire d'un client sera acceptée ou refusée. Le procédé est défini avec la notation BPMN [KLWL09].

Le processus de prêt bancaire débute par la réception d'une demande de prêt (RDP : Réception Demande Prêt). La demande de prêt du client est par la suite vérifiée par la tâche VDC (Vérifier Dossier Client). En effet, selon l'historique du client et d'autres paramètres (montant du prêt, salaire du client...etc.), la demande de prêt est traitée de différentes manières. Généralement le risque est évalué (ER : Évaluation du Risque) et est quantifié (QR : Quantification du Risque), mais la demande peut également être directement acceptée (DDA : Demande Directement Acceptée) ou refusée (DR : Demande Rejetée). À tout moment la hiérarchie peut intervenir directement (VH : Validation Hiérarchique). Ainsi la décision finale (DF : Décision Finale) est prise en tenant compte de la décision suite au traitement de la requête (DTR : Décision Traitement Requête) ainsi que de la décision de la hiérarchie.

Il est à noter que dans cet exemple, certaines tâches comprennent des informations sensibles quant au processus de l'organisation. Certains *fragments* dits *critiques* comprennent des informations sur le savoir-faire de l'organisation. Dans notre exemple de motivation, le fragment VDC est considérée « critique » car il comporte les informations liées à la prise de décisions au sujet de la demande du prêt bancaire. Respectivement, le fragment DF comprend des informations sur le mode de décision finale (accorder le prêt bancaire ou pas).

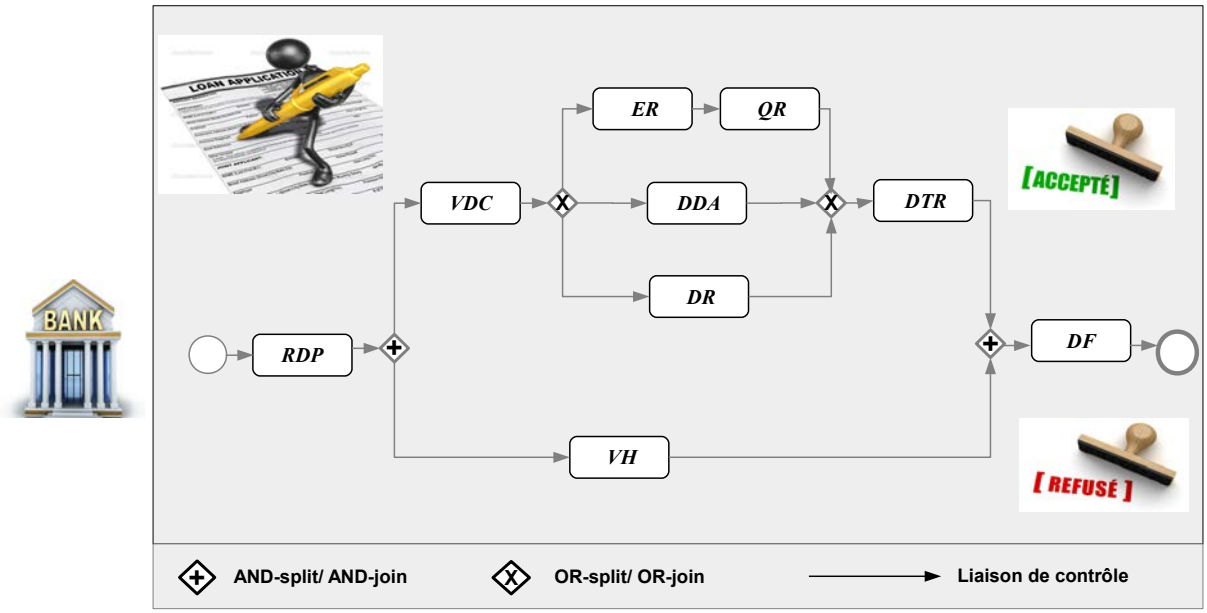


FIGURE 1.1 – Exemple de motivation

Par conséquent, même si la banque est prête à utiliser les ressources du cloud pour profiter de ses avantages, elle doit veiller à ce que sa stratégie pour accepter ou refuser un prêt bancaire ne soit pas divulguée. De la même manière, elle ne veut pas divulguer comment la hiérarchie est intervenue dans le processus ni comment la décision finale fut prise.

1.2.1 Point de départ

Cette thèse est une continuité de travaux effectués précédemment dans l'équipe COAST du Loria. Nous nous sommes principalement appuyés sur les travaux de Fdhila et al. [Fdh11][FYG09], qui ont proposé une méthodologie pour transformer un processus centralisé en une chorégraphie de BP fragments connectés par des messages de synchronisation. Cette chorégraphie décrit l'ensemble des interactions ayant lieu entre les BP fragments impliqués dans l'exécution du processus métier.

Cependant, étant donné que leur méthode n'optimise pas la qualité de service, elle fut améliorée dans [FDG10] pour optimiser l'assignation des services aux activités, après identification des services répondant aux exigences fonctionnelles des activités. Leur solution prend en compte les coûts de communication entre les services, le taux de données que ces services ont besoin d'échanger dans l'orchestration, et enfin un ensemble de contraintes de séparation/colocation imposées par le fournisseur de services.

La figure 1.2 représente notre exemple de motivation décomposé en partitions ((a), (b) et (c)) connectés par des messages de synchronisation, formant ainsi une chorégraphie.

En continuité aux travaux de Fdhila et al.[Fdh11] , Goettelmann et al. [GFG13] ont proposé une méthodologie pour déployer des processus métiers dans le cloud en répondant

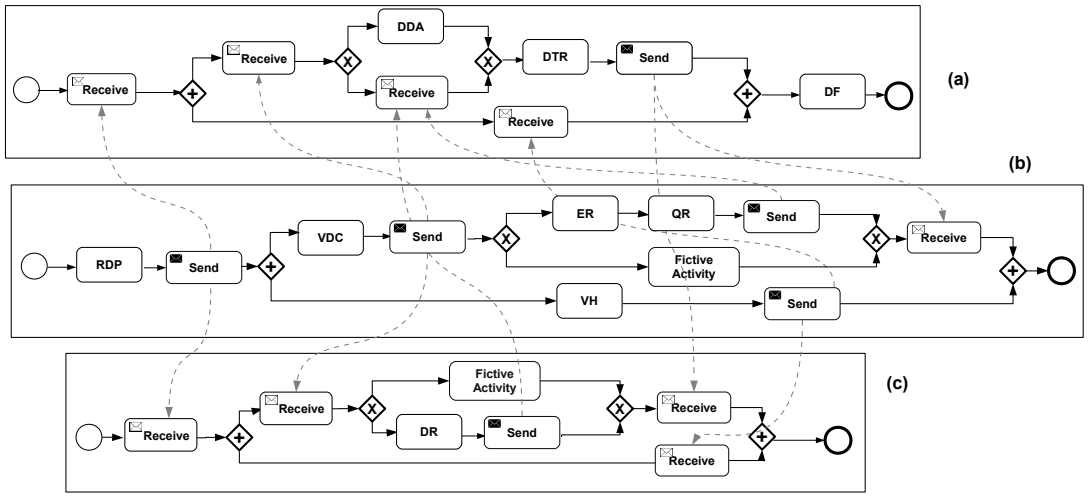


FIGURE 1.2 – Représentation du processus prêt bancaire sous la forme d'une chorégraphie de fragments

à certaines contraintes de sécurité, assurant ainsi la sécurité des données sensibles échangées. Leur méthode de décomposition est basée sur celle de Fhdila et al. [FDG10] avec une extension prenant en compte le contexte cloud. Cependant, à cause des problèmes de sécurité introduits par le cloud, et qui représentent le plus grand obstacle quant à son adoption par les entreprises, Goettelmann et al. [GMG13] ont proposé une méthodologie qui combine les techniques de modélisation et de sélection de clouds permettant un déploiement sécurisé. Il préconisent un ensemble d'exigences à différents niveaux.

(1) Niveau logique : ils proposent un ensemble de solutions pour protéger la logique du processus contre un éventuel cloud malicieux, e.g. séparer la logique des données, ou ajouter des tâches au processus afin de cacher sa réelle structure.

(2) Niveau organisationnel : représente l'ensemble des règles à appliquer pour l'assignation des tâches. Ces règles sont connectées aux exigences définies au niveau logique : séparer deux fragments sur différents clouds, les co-localiser sur le même cloud, ou encore exiger l'exécution de certaines tâches en local.

(3) Niveau informationnel : représente l'ensemble des contraintes qu'un client peut imposer quant à l'exécution de certaines tâches dans certains clouds, e.g. un client peut bannir un cloud pour l'exécution d'une tâche t car il ne lui fait pas confiance, ou au contraire imposer qu'un cloud C déploie une tâche t.

Cette solution est principalement une méthodologie ; les différentes règles pour l'assignation des tâches doivent être introduites par le designer lui même. Notre objectif est d'améliorer cette solution en *automatisant partiellement, voire complètement la décomposition et le déploiement du processus métier dans un environnement cloud sans intervention humaine*. Cette automatisation peut s'avérer être nécessaire lorsque le nombre de tâches du processus est important, i.e. nécessitant un nombre important de contraintes.

Par ailleurs, la solution de Goettelmann et al. [GMG13] permet de minimiser le risque auquel le processus est exposé lors du déploiement dans le cloud, mais ne permet pas de mesurer le niveau de risque effectif auquel il est exposé. Dans ce contexte, ils ont proposé dans [GDG+14] une méthode pour évaluer le risque auquel est exposée une configuration (tâches/cloud), en se basant uniquement sur les données fournis par le fournisseur. Leur solution prend également en compte un ensemble de contrôles à implémenter côté fournis-

seur afin de minimiser les menaces. Il est cependant à noter que ces travaux ne considèrent que les risques externes auxquels sont exposés les processus, en omettant le fait que les clouds puissent eux même être malicieux (en initiant une coalition de clouds), et peuvent ainsi porter préjudice à la préservation du savoir faire.

Le travail présenté dans cette thèse, basé sur les outils et approches de ces travaux précédents, a pour objectif de proposer des solutions qui dépassent les limites de ces travaux. Ceci est fait à travers l'automatisation de la décomposition du processus ainsi que le déploiement des différents fragments sur une architecture cloud. Nous portons attention à la possibilité de coalition de clouds malicieux ayant pour objectif la reconstruction partielle ou complète du processus. L'objectif de notre travail vise à apporter aux entreprises les garanties nécessaires afin de pouvoir déployer/réutiliser des fragments de processus pour profiter des avantages du cloud en toute sécurité.

1.3 Résumé des contributions

Cette partie donne un aperçu général de nos différentes contributions. Ces contributions répondent aux questions liées à la problématique posée dans la section 1.1. Elles couvrent les aspects liés à la sélection et à l'obfuscation des modèles de processus, ainsi qu'au déploiement optimisé de ces processus métiers.

1.3.1 Obfuscation d'un modèle de processus avant déploiement

Afin de permettre aux entreprises d'adopter l'environnement cloud tout en préservant la logique de leur processus, nous proposons deux approches complémentaires basées sur l'obfuscation des processus métiers lors du déploiement dans le cloud. Plus précisément, notre première approche se base sur la nature des fragments à déployer. En effet, nous pensons qu'il y a plus de savoir faire dans certains fragments dit « sensibles ». D'autre part, nous soutenons que chaque fragment sensible a son fragment complémentaire. Si ce dernier est déployé sur le même cloud que le fragment sensible, un éventuel cloud malicieux ou à un potentiel attaquant sera en mesure de découvrir la logique du processus. Ainsi, il devient nécessaire de *séparer ces fragments complémentaires* afin de préserver la logique du processus.

Dans cette optique, une première solution permet de générer un ensemble de contraintes (dites *de séparation*) à respecter lors du déploiement. Ces contraintes, en plus de répondre à un ensemble de besoins fonctionnels et non fonctionnels, assurent que les fragments sensibles complémentaires soient déployés sur des clouds différents, minimisant par conséquent la quantité d'information sensible détenue par chaque cloud. Cette approche sera détaillée dans la section 3.3 du chapitre 3.

Dans la première approche, nous nous sommes basés sur des contraintes de séparation pour minimiser le taux d'informations que détient chaque cloud, afin de complexifier la tâche de reconstruction du processus. Cette solution, bien qu'efficace ne permet que de protéger contre un cloud potentiellement malicieux, qui essayerait de reconstruire le processus à lui seul. Mais, en combinant leur informations, différents clouds sont à même de reconstruire la logique du processus. Ainsi, pour prévenir une coalition de clouds, la seconde approche permet de complexifier la logique du processus avant déploiement. Cette approche se base sur l'introduction de faux fragments à certains points stratégiques, permettant ainsi d'augmenter le nombre de clouds devant conspirer pour reconstruire une

information sensible du processus. Nous soutenons que plus ce nombre est important, et plus il sera difficile pour eux de former une coalition. En effet, convaincre un nombre important de clouds à conspirer est plus difficile que de n'en convaincre que quelques uns.

Cependant, étant donné que l'introduction de faux fragments peut s'avérer coûteux, notre déploiement répondra à certains besoins fonctionnels et non fonctionnels afin d'optimiser au mieux certaines dimensions de QoS (risque, coût...). Cette approche sera détaillée dans la section 3.4 du chapitre 3.

1.3.2 Mesure du niveau d'obfuscation d'un processus métier

Cette contribution se concentre sur la mesure du niveau d'obfuscation obtenu en appliquant nos approches du chapitre 3 (avant et après introduction des contraintes de séparations et des faux fragments). La première partie de notre travail fut de vérifier via un état de l'art si une mesure existante pouvait s'appliquer à notre contexte. Étant donné que nous n'avons trouvé aucune mesure adéquate, nous avons défini nos propres mesures, basées sur le peu de mesures existantes, qui considèrent que plus un processus est complexe, plus il est obfusqué. En effet, nous nous sommes inspirés des méthodes utilisées en génie logiciel pour mesurer la complexité des composants logiciels [HK81][McC76], en mettant l'accent sur le contrôle de flux. De plus, nous nous sommes prioritairement basés sur la mesure « Interface Complexity metric », développée par Cardoso et al. [CMNR06], (elle même adaptée de « Information flow Metric » developpée par Kafura et al. [HK81] pour le domaine du génie logiciel) que nous avons utilisés comme un composant de notre métrique. Cette partie sera détaillée dans le chapitre 3.

1.3.3 Mesure du niveau de risque auquel est exposé un processus métier après déploiement dans le cloud

Cette contribution permet de mesurer le niveau de risque auquel est exposé un processus métier sur une configuration (tâches/cloud) donnée, i.e. à quel point le savoir-faire du processus est exposé à des risques de divulgation. Le calcul du niveau de risque du processus prend en compte l'ensemble des niveaux de risques auxquels sont exposées les tâches sensibles sur tous les chemins les reliant à leur tâche complémentaire. Notre modèle de risque est principalement basé sur les définitions de risque introduites dans [Aus04, May09], qui considèrent le risque comme étant la probabilité qu'une menace exploite accidentellement ou intentionnellement une ou plusieurs vulnérabilités. La vulnérabilité d'une tâche étant elle même reliée au cloud qui la déploie, i.e. plus il est sûr, moins la tâche est vulnérable. De même, nous relions les menaces auxquelles est exposée une tâche sensible à la probabilité de coalition des clouds de la collaboration, i.e. à quel point il est difficile pour eux de former une coalition. L'efficacité de cette approche est présentée à travers un exemple d'application ainsi qu'une simulation qui illustre le comportement de notre métrique. Cette approche sera détaillée dans le chapitre 4 .

1.3.4 Optimiser le déploiement d'un processus métier en terme de sécurité et de coût

Cette contribution permet de déployer les processus métiers dans un contexte cloud en répondant à certains besoins en terme de QoS. En effet, déployer les processus sur différents clouds peut s'avérer être coûteux, et peut porter atteinte à certaines dimensions

de la qualité de service (QoS). Cette partie se concentre donc sur la notion d'optimisation, et plus précisément sur certaines dimensions de la qualité de services à savoir le risque et le coût. Nous tentons à travers les approches proposées de sélectionner les configurations qui répondent au mieux aux besoins des utilisateurs. Ainsi, notre première approche permet de réduire le risque auquel la collaboration est exposée en ne dépassant cependant pas un certain seuil de coût défini par l'utilisateur. De même, la seconde approche tente de sélectionner la configuration qui minimise le coût de déploiement en respectant un seuil de risque. Étant donné que ces méthodes sont des heuristiques, leur efficacité fut vérifiée en implémentant leur solution optimale, à travers l'introduction de modèles d'optimisation pour la réduction du risque et du coût. La comparaison entre ces méthodes (heuristiques et exactes) nous a permis de nous assurer de l'efficacité de nos méthodes. Cette approche sera détaillée dans le chapitre 5.

1.4 Synthèse

Ce chapitre résume la problématique de nos travaux de thèse qui portent sur l'obfuscation des processus métiers avant déploiement dans le cloud.

Tout d'abord, nous avons expliqué que l'introduction du cloud computing et son association aux processus métiers ont introduit de sérieux problèmes de sécurités, qui induisent à plusieurs questions de recherche.

Nous avons par la suite exposé les différents travaux de recherche sur lesquels se base notre travail de thèse, et avons exposé les différentes contributions développées tout au long de cette thèse pour répondre aux problèmes posés.

Dans le chapitre suivant, nous allons introduire les concepts et définitions liés au cloud computing, processus métiers et obfuscation. Nous présenterons ensuite un état de l'art qui illustre les différents travaux réalisés en vue de sélectionner les services du cloud qui répondent au mieux aux besoins des clients. Nous introduirons ensuite les différentes approches proposées afin d'externaliser les processus métiers dans un environnement cloud. Enfin, nous présenterons les approches d'obfuscation introduites dans la littérature et leur effet sur l'externalisation des processus.

Chapitre 2

Concepts et Etat de l'art

Sommaire

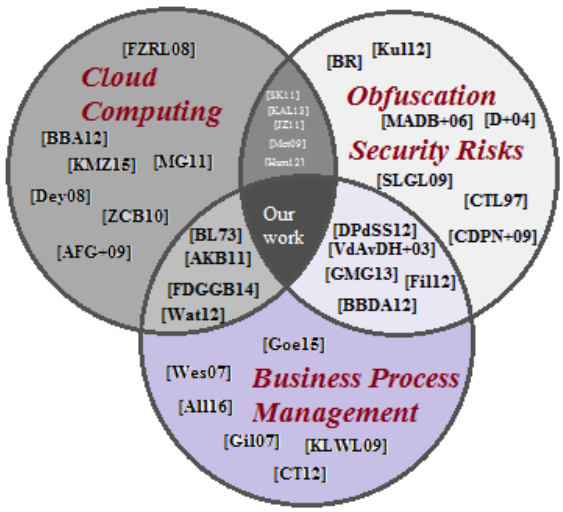


FIGURE 2.1 – Contexte de nos travaux

2.1 Introduction

Le travail de cette thèse se situe à l'intersection de trois grands domaines de recherche à savoir ; le cloud computing, la gestion des processus métiers, ainsi que la gestion des risques de sécurité, à travers l'utilisation de méthodes dites « d'obfuscation ». Nous présentons dans ce chapitre un état de l'art qui porte sur des travaux réalisés dans un contexte similaire au nôtre, en abordant ces trois grands domaines de recherche comme illustré dans la figure 2.1.

Nous commençons par introduire le concept du Cloud Computing ainsi que les problèmes de sécurité auxquels son adoption fait face. Nous présentons par la suite les processus métiers ainsi que les problèmes rencontrés quant à leur externalisation dans un contexte cloud. Enfin, nous présenterons les méthodes d'obfuscation proposées dans la littérature ainsi que les solutions qu'elles apportent quant à une externalisation dans un contexte cloud.

2.2 Concepts de base

2.2.1 Cloud Computing

Historiquement, les ressources disponibles en termes de puissance, stockage, calcul, etc n'étaient pas suffisantes pour répondre aux besoins du marché. Avec l'avènement du cloud computing, ces ressources sont devenues abondantes en termes de disponibilité, et bon marché en terme de coût [KMZ15]. En effet, elles sont fournies sous forme de services publics qui peuvent être loués et publiés par des utilisateurs tiers, via internet à la demande [ZCB10].

Cependant, l'idée principale qu'apporte le cloud computing n'est pas récente. Dans les années 1960, John McCarthy envisageait des aménagements informatiques qui seraient fournis au public comme des services [Par66]. Le terme « cloud » a été utilisé dans plusieurs contextes dans les années 1990, mais n'a été associé au fait d'offrir des services à travers internet qu'en 2006 [ZCB10]. Depuis, beaucoup de définitions ont été associées au cloud computing [BYV+09][FZRL08]. Parmi ces définitions, nous retenons celle du NIST (The National Institute of Standards and Technology) [MG11].

Le cloud computing est un modèle permettant l'accès omniprésent à un réseau partagé de ressources informatiques configurables (réseaux, serveurs, stockage, applications et services, par exemple) qui peuvent être rapidement provisionnées et diffusées avec un minimum d'efforts de gestion, ou d'interactions avec le fournisseur de service.

2.2.1.1 Architecture du Cloud Computing

L'architecture cloud est représentée par trois niveaux [Vie09] :

- **Infrastructure As a Service (IaaS)** est le niveau le plus bas, il vise à fournir l'infrastructure informatique. Au lieu d'acheter des logiciels, des serveurs ou du matériel réseau, les utilisateurs peuvent en bénéficier en tant que service entièrement externalisé, qui est généralement facturé en fonction de la quantité de ressources consommées. Des exemples de fournisseur IaaS sont Salesforce [Sal], Google doc [Goob] et zoho [Zoh].

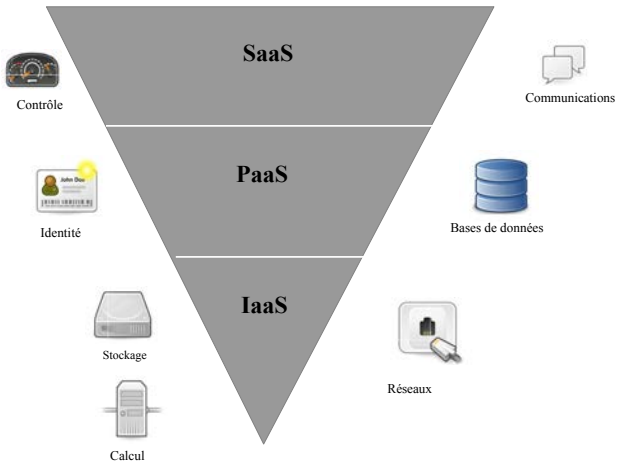


FIGURE 2.2 – Niveaux de service cloud

- **Platform As a Service (PaaS)** représente le second niveau, fournit le matériel et les plates-formes nécessaires permettants aux developpeurs de construire leurs programmes. Comme exemples d'offres PaaS, nous retrouvons Google App Engine [gooa] pour le développement d'applications Web et Microsoft SQL Azure [Mic] pour les bases de données.

- **Software As a Service (SaaS)** représente le niveau le plus haut et probablement la forme la plus populaire de cloud computing. SaaS réfère essentiellement aux logiciels et services hébergés sur des serveurs qui sont fournis en tant que services. Certaines utilisations initiales de SaaS incluaient les vidéos conférences, le courrier électronique et les systèmes de gestions de contenu. Les applications SaaS sont fournies à travers le web, pouvant être ainsi accessibles de n'importe quel ordinateur sans aucune pré-installation requise [Dey08].

Contrairement aux applications logicielles traditionnelles qui nécessitent un achat initial, les applications SaaS offrent généralement des prix basés sur des abonnements. En outre, le fournisseur gère l'application et l'utilisateur paie via une API Web. Les exemples les plus éminents de ce type de service sont Amazons Elastic compute Cloud (EC2) [Ama], Saccis Symphony [Sav] and RackSpace [Rac].

Il est à noter que cette classification n'est pas suffisante lorsque les processus métiers sont pris en compte. Un autre niveau d'abstraction au dessus de SaaS est introduit et est représenté par un quatrième modèle émergeant, i.e. Business Process as a Service (BPaaS) (voir figure 2.3). BPaaS ne signifie pas seulement que le logiciel est fourni à l'utilisateur en tant que service, mais peut signifier aussi fournir le flux logique et de contrôle du processus métier (que l'utilisateur veut exécuter) en tant que service.

Les différents modèles de cloud cités précédemment offrent l'opportunité de composer des services d'une grande variété de fournisseurs afin de créer ce qu'on appelle une *Syndication de clouds*. Ces syndications sont essentiellement des fédérations de fournisseurs de clouds dont les services sont agrégés ensemble, formant ce qu'on appelle « une

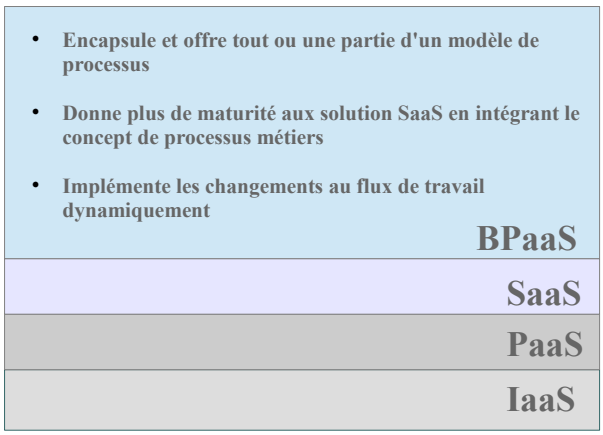


FIGURE 2.3 – Introduction du niveau BPaaS

architectures multi-clouds [6] ». Parmi ces services, au niveau SaaS, certains sont considérés comme des services BPaaS (*Business Process As a Service*) [BBA12]. BPaaS permet de créer de bout-en-bout des processus métiers, qui sont généralement syndiqués, i.e. réunis avec d'autres services externes (provenant de divers fournisseurs SaaS). Il offre par ailleurs des services d'entreprise pré embarqués ou conditionnés qui peuvent à leur tour être réutilisés par différentes applications, fournisseurs ou processus métiers. Cependant, même si beaucoup de recherches sont faites autour du cloud, le secteur des processus métiers y reste vaguement défini [SG12].

Il est à noter que beaucoup considèrent que BpaaS n'est pas un autre mot pour exprimer les services SaaS qui traitent les processus métiers, même si BPaaS tend à être considéré comme une forme spéciale de SaaS dans certains cas. Étant donné la confusion existant entre SaaS et BPaaS, différentes définitions lui ont été attribuées [IBM11] [Cha13]. Nous avons cependant décidé de retenir la définition de Accorsi [Acc11], qui ne considère pas BPaaS comme un service distinct de SaaS, mais plutôt comme un sous ensemble. Il a de ce fait défini BPaaS comme :

"Un modèle spécial SaaS dans lequel les fournisseurs de cloud fournissent des méthodes pour la modélisation, l'utilisation, la personnalisation et l'exécution (distribuée) des processus métiers."

2.2.1.2 Modèles de déploiement

Quatre types de modèles de déploiement sont généralement considérés pour le cloud [ZCB10] [MG11] :

- **Cloud public** est destiné au grand public, il permet un accès à des ressources gérées par une entreprise, une université ou une organisation gouvernementale. Il est le plus

6. Est l'utilisation de différents services de plusieurs clouds dans une seule architecture hétérogène.

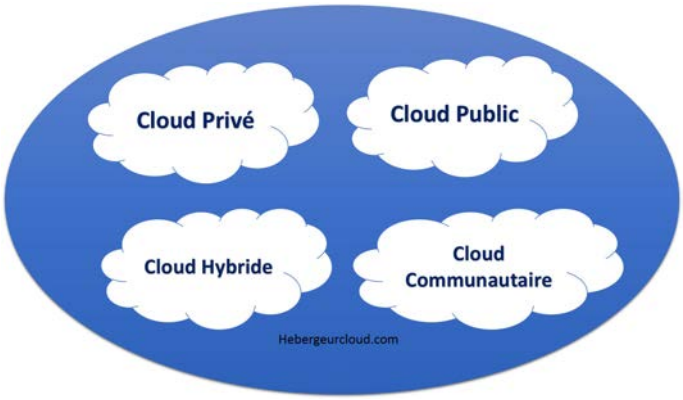


FIGURE 2.4 – Modèles de déploiement

intéressant en terme de coût, mais manque de contrôle sur les données, le réseau et les paramètres de sécurité, ce qui peut entraver son efficacité.

- **Cloud privé** est exclusivement destiné à une personne/organisation. Il offre le plus haut degré de contrôle de performance, fiabilité et sécurité. Il ne permet cependant pas d'économiser sur les coûts d'investissement initiaux.

- **Cloud hybride** est une combinaison du cloud public et privé permettant ainsi de répondre aux limites de chacun.

- **Cloud communautaire** est la possibilité pour plusieurs entités ou membres d'organisations ayant les mêmes besoins d'utiliser une seule et unique solution Cloud. Dans ce cas, le Cloud est partagé au sein de plusieurs structures (sociétés, filiales, regroupement d'entreprises, groupes métiers etc.) et sa gestion est assurée soit en interne, soit en externe.

2.2.1.3 Les acteurs du cloud

Traditionnellement dans les configurations informatiques, les principaux acteurs se résument aux consommateurs et aux serveurs. Chaque acteur est une entité (une personne ou une organisation) participant à une transaction/processus qui exécute des tâches dans le cloud computing [MG11]. Les consommateurs utilisent, possèdent, entretiennent et mettent à jour les systèmes tandis que les fournisseurs s'occupent de la vente, de l'installation, de l'octroi de licences, du conseil et de la maintenance [MLB+11]. Le cloud computing modifie le rôle des différents intervenants en introduisant de nouveaux acteurs. En effet, étant donné qu'il offre différents services (IaaS,PaaS,SaaS) générant ainsi différents niveaux de contrôle, sécurité et configuration, il devient nécessaire d'avoir un intermédiaire entre les clients et les serveurs [HT12]. Dans l'architecture cloud, cet intermédiaire est connu sous le nom de **Broker** [MG11]. Ainsi, les différents acteurs du cloud peuvent être résumés comme suit : (voir figure 2.5)

- **Consommateur de cloud** Le NIST (National Institute of Standards and Technology) définit le consommateur de cloud comme étant une personne ou organisation qui maintient une relation commerciale avec un fournisseur cloud en utilisant ses

services [MG11]. Les consommateurs de cloud peuvent être classés en quatre catégories : consommateurs SaaS (Infrastructure as a Service), consommateurs PaaS (Platform Service), consommateurs SaaS (Software as a Service), et consommateurs XaaS (Anything as a Service) [ALIJ13]. Ainsi, selon ses besoins, le consommateur souscrit à un service proposé par un fournisseur de cloud. Ce consommateur peut être une personne, une entreprise, ou encore un fournisseur cloud qui fait appel à un autre fournisseur pour un service.

- **Fournisseur de cloud** Le NIST définit le fournisseur de cloud comme une organisation ou une entité mettant à disposition des consommateurs de cloud des services basés sur les SLA (Service Level Agreement) [MG11]. Un fournisseur de cloud peut être lui même consommateur d'un autre cloud.
- **Broker** Le NIST définit le broker comme une entité qui négocie les relations entre les consommateurs et les fournisseurs de cloud. En effet, il agit comme intermédiaire entre eux, permettant ainsi d'aider les consommateurs dans la sélection des services adéquats, répondants à leur besoin d'une part. D'autre part, il permet d'aider les fournisseurs dans l'implémentation des contrôles nécessaires afin d'offrir un niveau de sécurité optimal. Le NIST catégorise les services du Broker en trois catégories : [MG11]

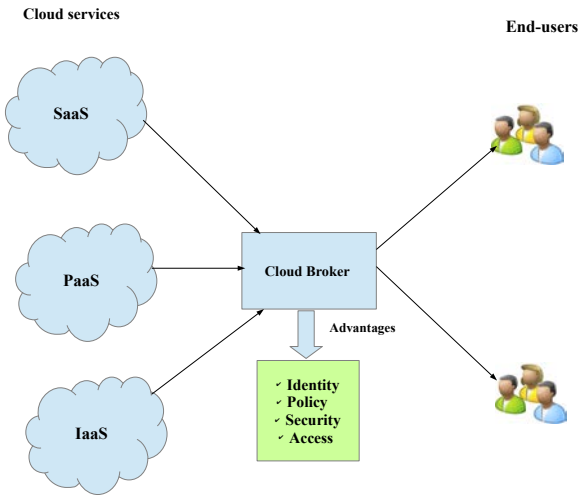


FIGURE 2.5 – les acteurs du cloud

  - **Intermédiaire** un cloud Broker améliore un service donné en améliorant certaines capacités spécifiques. Cette amélioration peut se traduire par une meilleure gestion d'accès aux services du cloud, la gestion des identités, une sécurité renforcée...etc.
  - **Agrégation** le broker combine plusieurs services générant ainsi de nouveaux services, plus adaptés aux besoins des consommateurs. Il assure aussi la sécurité des données transitant entre le consommateur et le fournisseur du cloud.
  - **Arbitrage** Permet de sélectionner le fournisseur offrant le service répondant aux mieux aux besoins du consommateur parmi un ensemble de fournisseurs de

services.

2.2.1.4 Synthèse

Cette partie introduit les différents concepts de base liés à l'environnement cloud. Outre la complexité de son architecture, la possibilité d'agréer ses services en formant une architecture multi-cloud complexifie davantage son utilisation. En effet, sélectionner le service qui répond au mieux aux besoins de l'utilisateur parmi un large choix peut s'avérer être compliqué.

Notre travail de thèse se situe par rapport à cette architecture cloud au niveau *brocker*. Nous nous définissons comme intermédiaire entre les consommateurs de cloud (i.e. les entreprises dans notre cas) et les fournisseurs cloud. Dans ce contexte, nous tentons de proposer les solutions adéquates à adopter afin de sélectionner les services BPaaS qui répondent au mieux à leurs besoins, i.e. profiter des avantages du cloud en terme de temps et de coût pour la construction et l'automatisation des processus métiers, en répondant aux problèmes de sécurité induits par la complexité de son architecture.

Nous introduisons dans ce qui suit la notion de processus métiers, ainsi que les différents concepts relatifs à la bonne compréhension des approches proposées au cours de cette thèse.

2.2.2 Processus métiers

Un processus métier peut être vu comme une collection d'activités exécutées et coordonnées afin de produire un résultat bien spécifique, répondant aux besoins d'un client ou d'un marché [Sys04]. Un processus est donc un ordre spécifique d'activités à travers le temps ayant un début, une fin, ainsi que des entrées/sorties clairement définis. Chaque processus métier est affecté à une seule organisation, mais peut interagir avec d'autres procédés métiers appartenant éventuellement à d'autres organisations. Chaque processus métier a un objectif et est affecté par des éventements externes ou internes, ou se produisant dans d'autres processus [Ko09].

Business processes

A business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations [Wes07] .

2.2.2.1 Fragment de Processus

Plusieurs définitions ont été attribuées à un fragment de processus. Vanhatalo et al.[VVL07] considèrent un fragment de processus comme un sous graphe connecté d'un graphe de processus. Un graphe de processus contient des nœuds, qui représentent des activités à réaliser, et des arêtes qui représentent des contrôles de dépendances entre eux [EUL09][LR00].

Un fragment de processus peut être créé de deux manières différentes [SLM+10] :

L'approche top-down dans cette approche, un fragment est créé à partir d'un proces-

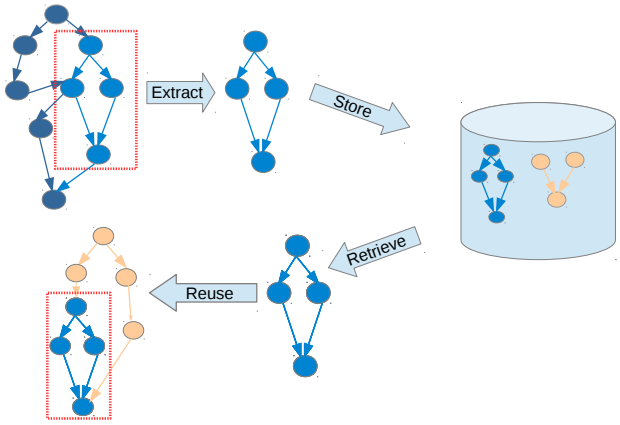


FIGURE 2.6 – Fragments de processus

sus existant. Il est créé en extrayant des structures connectées de ce processus. Le fragment résultant est de ce fait un sous-graphe d'un graphe de processus.

L'approche bottom-up dans cette approche, un fragment est construit de zéro. Il est conçu pour implémenter un ensemble de besoins et n'est donc pas un sous-graphe d'un graphe de processus existant.

Il est à noter que nous nous basons au cours de cette thèse sur la définition de [VVL07], qui considère un fragment de processus comme un sous-graphe connecté d'un graphe de processus déjà existant. Un sous graphe peut contenir une ou plusieurs tâches.

Nous considérons aussi que certains fragments de processus sont *critiques* en comparaison à d'autres fragments. Un fragment est critique s'il comporte des informations sensibles permettant à un éventuel attaquant d'extraire des informations sur le savoir faire de l'organisation à qui il appartient. Cette notion de fragment sensible sera détaillée dans le chapitre 3.

2.2.2.2 Gestion des processus métiers (BPM)

La gestion des processus métiers (BPM : Business Process Management) étant devenue un domaine de recherche reconnu dans la discipline des technologies de l'information en générale, et des systèmes d'informations en particulier, a pour objectif d'accroitre l'efficacité des processus organisationnels à travers de nouvelles stratégies de gestion et d'amélioration des performances d'une entreprise. Ceci est possible en analysant et adaptant régulièrement les processus, workflows et procédures en fonction des contraintes identifiées pour développer les performances ou la qualité de service [VDATHW03]. En effet, le BPM comprend les méthodes, les techniques et les outils pour appuyer la conception, la mise en œuvre, la gestion et l'analyse des processus opérationnels. L'objectif étant d'obtenir une meilleure vue globale de l'ensemble des processus métiers de l'entreprise et de leurs interactions dans le but de les optimiser et dans la mesure du possible, de les automatiser.

2.2.2.3 Classification des standards de BPM

L'un des aspects les plus important de la gestion des processus (BPM) est la modélisation de processus. En effet, il existe plusieurs langages de modélisation permettant de fournir la syntaxe et la sémantique appropriés, afin de spécifier les exigences des processus métiers. L'objectif étant de supporter l'automatisation de leur vérification, validation et simulation [LS07].

Ryan et al. [KLWL09] les classifient comme suit :

- **Standards graphiques** Permet d'exprimer les processus métiers, leurs flux et transitions schématiquement. Parmi ces modèles nous retrouvons BPMN [7] et YAWL [8].
- **Standards d'exécutions** Permet d'informatiser le déploiement et l'automatisation des processus métiers. Un exemple de ces standards est YAWL et XPDL.
- **Standards d'échanges** Facilite l'import/export des processus en facilitant la portabilité des données (e.g. XPDL, XML).
- **Standards de diagnostic** Offre des capacités de surveillance permettant d'auditer les processus des compagnies, mais aussi d'identifier les éventuels goulot d'étranglements se produisant, e.g. BPQL (Business Process Query Language), BPRI(Business Process Runtime Interface).

Parmi les langages utilisés au cours de cette thèse nous retrouvons :

- **BPMN (Business Process Modelling Notation)** : définit par la BPMI [9], BPMN est considéré comme un standard graphique basé sur des organigrammes semblables aux diagrammes d'activités UML. Il comprend également d'autres concepts tel que des passerelles ou évènements complexes. La version 2.0 inclut la sémantique d'exécution le rendant donc exécutable [OMG13]. La force de BPMN réside dans le fait qu'il fournisse une notation intuitive, et donc fournit un standard qui est facilement compréhensible aussi bien pour les utilisateurs techniques que pour les professionnels [Goe15]. La figure 2.7 illustre notre exemple de processus de prêt bancaire en BPMN.
- **RPST (The Refined Process Structure Tree)** est introduite par Vanhatalo. et al. [VVK09]. Cette structure permet de décomposer un modèle de processus en une hiérarchie de blocs canoniques uniques et modulaires.

  Un block canonique est un fragment de BP avec une seule entrée et une seule sortie (SESE : Single Entry/Single Exit). L'entrée peut être représentée par une tâche (dans le cas d'un flux séquentiel), ou une passerelle divisant le flux en différents flots alternatifs (dans le cas d'un $X(OR)$), ou en flots concurrents (dans le cas d'un AND). La sortie d'un block canonique commençant par une passerelle $X(OR)$ split est une passerelle $X(OR)$ join (bloc de décisions). De même, si l'entrée du block est une passerelle AND split, la sortie sera une passerelle AND join (bloc de synthèses). Ainsi, un bloc canonique peut être vu comme un fragment se situant entre des

7. Business Process Modelling Notation.
8. Yet Another Workflow Language.
9. Business Process Management Initiative : était un consortium d'entreprises spécialisées dans le marché des produits pour la gestions des processus. Elle fut fusionnée en 2005 avec l'Object Management Group (OMG).

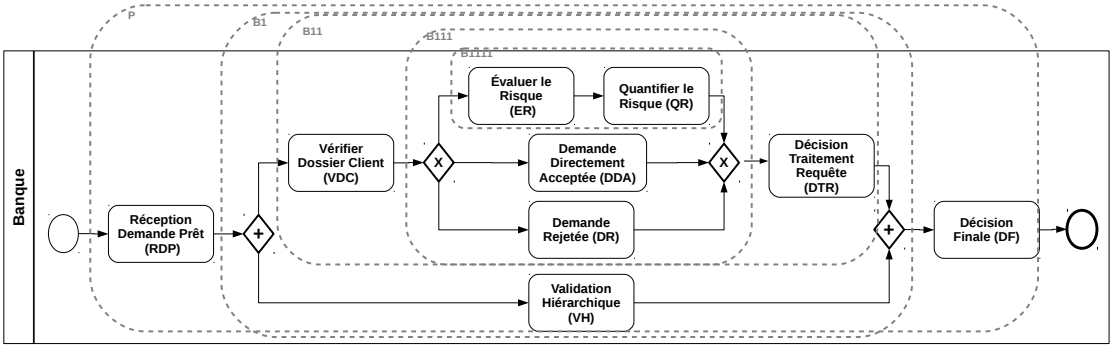


FIGURE 2.7 – Processus de prêt bancaire en BPMN

passerelles complémentaires. Les différents blocs canoniques du processus ne doivent pas se chevaucher [10].

L'arbre *RPST tree* de tout processus est composé de l'ensemble de tous ses blocs canoniques SESEs. Chaque bloc SESE ayant un parent (à l'exception de la racine), un ou plusieurs fils (à l'exception des feuilles), un frère droit (qui le précède dans la liste des blocs de son parent) et un fils gauche (qui le suit dans cette liste). La racine de l'arbre représente le graphe en entier (figure 2.8).

La construction de cet arbre passe par les étapes suivantes [PVV10] :

- **Étape 1** : Cette étape consiste en l'extraction de l'ensemble des blocs canoniques SESE à partir du modèle de processus. L'avantage de la décomposition canonique réside dans le fait qu'elle permet d'obtenir une décomposition unique et modulaire, i.e. tout changement effectué au niveau d'un bloc canonique n'aura d'effet qu'en local (le bloc parent et les blocs frères restent inchangés).
- **Étape 2** : Cette étape permet de construire l'arbre R-PST à partir de l'ensemble des blocs canoniques. Cette construction se fait de bas (à partir des feuilles) en haut (la racine). Chaque bloc "fils" est rattaché au plus petit bloc canonique le comprenant (considéré donc comme son parent).

La représentation de notre exemple de motivation de la section 1.1 à travers cette structure est illustré dans la figure 2.8.

2.2.2.4 Synthèse

Nous avons introduit dans cette partie la notion de processus métiers et de gestion des processus métiers (BPM : Business Process management) et avons présenté les langages utilisés au cours de cette thèse pour leur modélisation.

Bien qu'il existe plusieurs langages dans la littérature, nous nous somme principalement basés sur BPMN pour la représentation de nos processus métiers. Ce choix se justifie par la richesse des notions qu'il comprend (e.g. passerelles et évènements complexes, etc.) ainsi

10. Deux fragments F_1 et F_2 se chevauchent si $F_1 \cap F_2 \neq \emptyset$, $F_1 \backslash F_2 \neq \emptyset$,$F_2 \backslash F_1 \neq \emptyset$. Donc F_1 et F_2 se chevauchent s'ils sont imbriqués ($F_1 \subseteq F_2$, ou $F_2 \subseteq F_1$) ou disjoint ($F_1 \cap F_2 = \emptyset$)

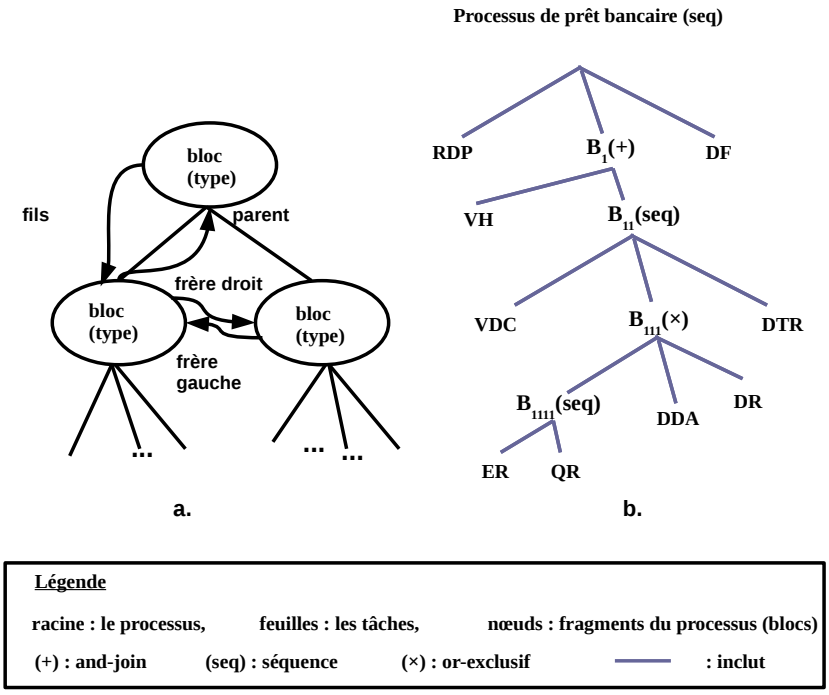


FIGURE 2.8 – (a) : R-PST Concept abstrait, (b) : Processuss de prêt bancaire en R-PST

que la facilité de compréhension qu'offre sa représentation. Il est cependant à noter que nous considérons dans cette thèse que les processus métiers sont structurés [11] [PGBD10].

Nous avons également utilisé au cours cette thèse la structure RPST (Refined Process Structured Tree) pour la décomposition des processus métiers. Sa notion de blocs canoniques permet de faciliter l'analyse et la reconstruction des processus métiers et s'adapte avec nos hypothèses de modélisations (qui seront introduites dans le chapitre 3).

Nous introduisons dans ce qui suit le troisième axe de cette thèse, qui concerne l'obfuscation des programmes en général et des processus métiers en particulier.

2.2.3 Obfuscation : concept

L'obfuscation peut être vue comme l'action de transformer un programme C en un autre programme O(C), ayant les mêmes fonctionnalités mais étant plus complexe (plus difficile à comprendre), afin d'éviter l'ingénierie inversée [BGI+12][BR].

Formellement, Colleberg définit l'obfuscation comme suit [CTL97] :

- $P \xrightarrow{\tau} P'$ est la transformation d'un programme P en un programme P'. Elle est considérée comme étant une obfuscation si P et P' ont le *même comportement*. Plus

11. Un processus structuré est un processus où pour chaque passerelle ouvrante $(x)OR$ split correspond une passerelle fermante $(x)OR$ join, et respectivement pour chaque passerelle ouvrante AND split, correspond une passerelle fermante AND join.

précisément, une transformation est considérée comme obfuscation si :

- Si P ne se termine pas ou se termine avec une erreur, alors P' se termine avec une erreur.
- Sinon, P' retourne les mêmes résultats que P.

L'obfuscation du code est largement plus répandue que l'obfuscation des processus métiers. En effet, il existe différentes méthodes pour l'obfuscation du code qui dépendent directement du type de transformations opérée sur le code, dans le but de le protéger (sémantiquement et lexicalement). Nous la prenons donc comme référence.

2.2.3.1 Obfuscation du code

Comme introduit précédemment, le rôle de l'obfuscation est de compliquer la compréhension d'un programme. Elle peut opérer à différents niveaux et peut prendre différentes dimensions ([CTL97] [Wro02]).

i) Obfuscation de la structure Concerne le code source ou les structures binaires. Cette méthode supprime les informations pertinentes (tel que le nom des classes, champs et opérateurs) sans changer le comportement du code, mais de façon à rendre plus difficile pour un attaquant la compréhension des fonctionnalités implémentées dans les différentes parties du code.

ii) Obfuscation des données L'obfuscation de données peut être classifiée en trois grandes catégories :

- Obfuscation de l'encodage : Est réalisée en modifiant la représentation des méthodes et des variables (ex : inverser les valeurs logique TRUE et FALSE pour les variables booléennes).
- Obfuscation par agrégation : C'est à dire fusionner les données indépendantes et dissocier les données dépendantes.
- Obfuscation sur l'ordre : Augmenter la distance physique (dans le code) entre deux éléments logiquement connectés.

iii) Obfuscation du contrôle Peut être classifiée en trois catégories :

- Obfuscation du calcul : C'est à dire apporter des modifications à la structure du flux de contrôle (ex : ajouter des conditions qui n'impacteront pas le comportement du programme).
- Obfuscation par agrégation : C'est à dire diviser et fusionner des fragments de codes.
- Obfuscation sur l'ordre : modifier l'ordre des blocs, boucles ou expressions.

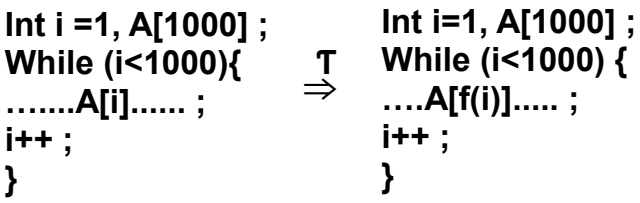


FIGURE 2.9 – Exemple d'obfuscation de code

Obfuscation préventive Est la prévention des décompilateurs et des débogueurs.

La figure 2.9 illustre l'obfuscation d'un code en réordonnant les éléments d'un tableau. Ceci est réalisé par une fonction de codage $f(i)$, qui mappe le *ième* élément dans le tableau original à sa position dans le nouveau tableau réordonné. Il est cependant à noter que ceci n'est qu'une manière d'obfusquer, et qu'il en existe plusieurs autres (comme expliqué dans la section 2.2.3.1).

2.2.3.2 Mesure de l'obfuscation d'un programme

La mesure de l'obfuscation d'un programme représente *la qualité* de l'obfuscation apportée par une transformation. Colleberg et al. [CTL97] la classifient selon trois critères : (1) à quel point la transformation ajoute de confusion au programme. (2) à quel point il est difficile de déobfusquer le programme.(3) Quels sont les frais introduits par cette opération d'obfuscation. Les mesures *puissance, résistance et coût* permettent de répondre respectivement à ces questions.

i) Puissance (Potency) Un programme est *puissant* s'il arrive à dérouter un éventuel attaquant du réel fonctionnement du code original. En d'autres termes, la puissance mesure à quel point le code obfusqué est plus difficile à comprendre que le code original. Colleberg et al. [CTL97] considèrent qu'un code est plus difficile à comprendre lorsqu'il a une plus grande complexité. Ils présentent un ensemble d'actions pouvant être utilisées pour le calcul de la complexité d'un programme [McC76] [Ovi93]. Cette mesure est généralement *à minimiser*, mais dans le contexte de l'obfuscation elle doit être *maximisée*. Ainsi, la puissance d'un code est mesurée selon la complexité de sa version originale et sa version obfusquée comme suit :

$$T_{pot}(P) \equiv E(P')/E(P) - 1 \tag{2.1}$$

où $E(P)$ et $E(P')$ sont les complexités du programme avant et après obfuscation respectivement.

La transformation est considérée *puissante* si $T_{pot}(P) > 0$

ii) Résistance Représente le niveau de résistance d'une transformation à une attaque d'un déobfuscateur, elle peut être mesurée à travers deux paramètres :

- *L'effort du programmeur :* Représente le temps nécessaire pour la construction d'un déobfuscateur automatique capable de réduire le niveau de puissance du programme.
- *L'effort du déobfuscateur :* Le temps nécessaire pour que le déobfuscateur réussisse effectivement à réduire le niveau de puissance de la transformation.

Ainsi, la résistance est calculée comme suit :

$$T_{res} = Resilience(T_{Deobfuscator\,effort}, T_{Porgrammer\,effort}) \tag{2.2}$$

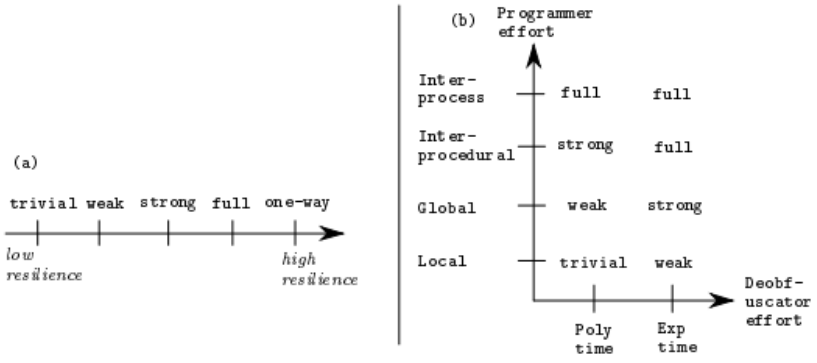


FIGURE 2.10 – Résistance d'une obfuscation (source :[CTL97])

La figure 2.10-(a) représente les différentes valeurs de la résistance. Elle peut aller d'une résistance légère à forte. La forte résistance est généralement dûe à des transformations dites *à sens unique* qui ne peuvent jamais être annulées. Ceci est dû au fait de supprimer du programme des informations nécessaires à l'humain mais qui n'entravent pas l'exécution correcte du programme.

La figure 2.10-(b) représente la niveau de résilience effectif selon les efforts du programmeur et du déobfuscateur. Les efforts du déobfuscateur sont classifiés comme temps polynomial ou exponentiel tandis que ceux du programmeur dépendent directement de l'ampleur de la transformation, i.e. il est plus facile de construire des contremesures contre une obfuscation qui affecte qu'une petite partie d'une procédure, qu'une obfuscation qui affecte un problème en entier. Ainsi, les efforts du programmeur vont de local (n'affecte qu'un seul bloc basique) à inter processus (affecte l'interaction entre les processus).

iii) Coût Représente le cout d'une transformation. Il est classifié en $< gratuit, peu\ coûteux, coûteux, très$

$$T_{cost}(P) \equiv \begin{cases} gratuit & si\ P'\ requiert\ O(1)\ plus\ de\ ressource\ que\ P \\ peu\ coûteux & si\ P'\ requiert\ O(n)\ plus\ de\ ressource\ que\ P \\ coûteux & si\ P'\ requiert\ O(n^p),\ p>1,\ plus\ de\ ressource\ que\ P \\ très\ coûteux & si\ P'\ requiert\ exponentiellement\ plus\ de\ ressource\ que\ P \end{cases} \tag{2.3}$$

Ainsi, la *qualité* d'une obfuscation peut être mesurée en utilisant les trois critères introduits ci-dessus comme suit :

$$T_{qual}(P) = (Tpuiss(P), T_{res}(P), T_{coût}(P)) \tag{2.4}$$

2.2.3.3 Obfuscation des processus métiers

L'obfuscation des processus métiers est inspirée de l'obfuscation du code. En effet, son objectif est de préserver le savoir faire contenu dans les modèles de processus. Elle peut être réalisée de différentes manières complémentaires [GMG13] :

i) Retenir les données sensibles et la logique en interne Une façon simple de préserver le savoir faire des organisations est de garder en interne les fragments sensibles qui contiennent beaucoup d'informations pertinentes.

ii) Séparer la logique du processus en différents BP fragments Une autre solution intuitive serait de diviser le processus en sous-fragments, et de distribuer ces fragments sur différents clouds. De ce fait, chaque cloud n'aura qu'une vue partielle de la logique générale du processus. Ceci est en quelques sortes relié à l'approche *Aggregation obfuscation* dans l'obfuscation du code (control obfuscation, section iii)). Pour garder la sémantique initiale du processus, ces différents fragments seront associés ensemble afin de former une collaboration.

iii) Ajout de logique non fonctionnelle Ajouter du code inutile dans un programme est une manière utilisée par les programmeurs pour l'obfusquer. Ceci est relié à l'approche *Computation obfuscation* utilisée dans l'obfuscation du code (control obfuscation, section iii)). Dans le même objectif, des BP fragments inutiles peuvent être ajoutés.

iv) Obfusaction des données L'obfusaction des données est largement utilisée pour l'obfusaction des programmes. En effet, la cryptographie est utilisée pour obfusquer les données. Cependant, pour exécuter une tâche, les données doivent être en claire (non cryptées). Une autre solution pouvant être utilisée avec des données en claire est *l'anonymization*, qui consiste à supprimer certaines informations pertinentes pouvant divulguer d'éventuelles informations personnelles. Dans notre contexte, elle permet de cacher (partiellement) le lien entre la logique et les données.

v) Séparer la logique des données L'idée est de sauvegarder la logique et ses données correspondantes à différents endroits. De ce fait, un cloud analysant la logique ainsi que les données stockées, ne peut extraire de liens ou d'informations pertinentes. Cette idée est directement reliée à la solution *Ordering obfuscation* dans l'obfuscation du code (data obfuscation, section ii)).

vi) Distribuer les instances sur plusieurs clouds Distribuer les instances du processus sur plusieurs clouds permet de minimiser la quantité de données par cloud. Par conséquent, aucun cloud n'aura assez d'informations pour reconstituer la logique du processus. (Le nombre de cas nécessaires pour reconstruire un BP modèle est facilement calculable [VdAvDH+03]).

2.2.3.4 Les mesures de complexité des processus métiers

L'évolution des processus métiers ainsi que l'intégration du concept de gestion des processus métiers (BPM) a vu leur intégration pour la mise en œuvre et l'intégration des

systèmes d'information à grande échelle. Il devient donc nécessaire de comprendre comment les erreurs peuvent être évitées, comment l'entretien peut être facilité, ou comment la qualité des processus peut être améliorée [CMNR06]. Dans ce contexte, il est prouvé que la complexité est un déterminant de la probabilité d'erreur d'un processus métiers. Cependant, mesurer la complexité des modèles de processus est un nouveau domaine de recherche avec encore peu de contributions. Par conséquent, beaucoup de mesures émanant de disciplines voisines tentent d'être adaptées au contexte processus métiers pour le calcul du niveau de complexité.

Cardoso et al.[CMNR06] affirment que les métriques conçues pour le calcul de la complexité des programmes écrit en C++, JAVA, etc peuvent être adaptées pour l'analyse et l'étude de certains caractéristiques des processus métier tel que la complexité, la compréhension et la maintenance. En effet, ils considèrent qu'il existe une forte analogie entre les programmes et les processus métiers ; un processus métiers modélisé en BPEL par exemple peut être vu comme un programme traditionnel partitionné en modules ou fonctions (i.e. des activités), qui prend des paramètres en entrée et retournes des sorties. Ainsi, ils proposent un ensemble de métriques principalement utilisées pour mesurer la complexité du code, et tentent de l'adapter au contexte des processus métiers. Ces mesures peuvent être résumées comme suit :

i) La métrique LOC (Number of Line of Code) est considérée comme l'une des mesures les plus répandues pour l'analyse du code [Kal90]. Elle se base sur le simple calcul du nombre de lignes de code d'un programme. Cette métrique considère que la taille d'un programme peut être utilisée comme prédicateur de certaines de ses caractéristiques telles que l'occurrence des erreurs et sa facilité de maintenance. Par conséquent, en tenant compte de la taille d'un processus, une métrique $M1$ peut être dérivée en se basant sur le nombre d'activité d'un processus.

$$M1 : NOA = Nombre\ d'activités\ d'un\ processus \tag{2.5}$$

Cependant, étant donné que cette métrique ne prend en compte que le nombre d'activités pour calculer la complexité d'un processus, elle fut réadaptée afin de prendre en compte les éléments de flux de contrôle. Il est à noter que cette nouvelle métrique n'est adaptée que pour les processus dits « structurés ».

$$M2 : NOAC = Nombre\ d'activités\ et\ d'éléments\ de\ flux\ de\ contrôles \tag{2.6}$$

Par ailleurs, puisque certains langages permettent la construction de processus non structurés, une troisième métrique fut proposée et permet de compter le nombre de *split* et *join* en plus du nombre d'activités.

$$M3 : NOAJS = le\ nombre\ d'activités,\ split\ et\ join\ d'un\ processus \tag{2.7}$$

ii) La métrique Interface Complexity (IC) Proposée par Cardoso et al.[CMNR06], cette métrique est adaptée de la métrique de Henry et Kafura.[HK81] qui se base sur l'impacte du flux d'informations sur la structure d'un programme. En effet, Henry et al.[HK81] suggèrent d'identifier le nombre d'appels à un module (le flux d'informations entrants *fan-in*) et le nombre d'appel à partir d'un module (le flux d'informations sortants *fan-out*) pour mesurer la complexité d'un programme PC

$$PC = length * (fan - in * fan - out)^2 \tag{2.8}$$

Ainsi, en considérant un module comme une activité, et les fan-in et fan-out comme ses entrées/sorties, la complexité IC d'un modèle peut être calculée comme suit :

$$PC = length * (nomberofinputs * numberofoutputs)^2 \quad (2.9)$$

Il est à noter que la taille d'une activité (length) est calculée selon sa nature. Si c'est une activité *boite noire* sa taille est considérée égale à 1. Dans le cas contraire, la taille de l'activité est calculée sur le principe de la connaissance de son code source, en utilisant des métriques traditionnelles pour le calcul de la complexité logicielle e.g. la métrique LOC.

iii) Complexité des processus métiers basée sur la mesure de Halstead Cette mesure est basée sur la mesure de Halstead [Hal77] qui se base sur la mesure de la compréhension d'un programme pour quantifier sa complexité. En effet, sa mesure comprend un ensemble de primitives pouvant être dérivées du code source :

- $n1$ = le nombre unique d'opérateur (si, tant que...etc.).
- $n2$ = le nombre unique d'opérande (variables ou constantes).
- $N1$ = le nombre d'occurrences de l'opérateur.
- $N2$ = le nombre d'occurrences de l'opérande.

Ainsi, Cardoso et al.[CMNR06] ont adapté ces primitives au concept de processus métiers en considérant $n1$ comme le nombre unique d'activités split, join et flux de contrôle, $n2$ comme le nombre unique de données manipulées par les activités du processus. Par ailleurs, ils proposent la métrique nommée *Halstead-based Process Complexity (HPC)* pour estimer la taille, le volume et la difficulté des processus comme suit :

- Taille du processus : $N = n1 * log2(n1) + n2 * log2(n2)$.
- Volume du processus : $V = (N1 + N2) * log2(n1 + n2)$.
- Difficulté du processus : $D = (n1/n2) * (N2/n2)$.

iv) La métrique CFC (Control flux complexity) Inspirée du nombre cyclomatique de McCabe [McC76] qui se base sur le flux de contrôle d'un graphe pour mesurer la complexité, la métrique CFC proposée par Cardoso [Car05] prend en compte le nombre de décisions dans le flux de contrôle pour le calcul de la complexité d'un modèle. En effet, étant la dérivée de la métrique de McCabe qui représente le nombre de chemins linéairement indépendants d'un programme, i.e. à travers le calcul du nombre de décisions binaires (pour les structures if) plus 1, et du nombre de décisions non binaires (comptabilisés comme n-1) avec n étant le nombre de résultats possibles, la métrique CFC est construite comme tenant compte des éléments suivants :

- *AND-split* puisque toutes les transitions sortant du AND doivent être exécutées et seront considérées comme un seul résultat, chaque AND-split ajoute la valeur 1 à la métrique CFC.
- *XOR-split avec n transitions sortantes* puisque une seule transition sera considérée parmi n, il est important de prendre en compte les n possibilités. Ainsi, chaque XOR-split ajoute n à la métrique CFC.

- *OR-split avec n transitions sortantes* au moins une seule transition et au maximum n des transitions sortantes du OR-split seront considérées. Par conséquent, chaque OR-split ajoute $2^n - 1$ à la métrique.

L'avantage de cette métrique est qu'elle permet de mesurer la complexité relative des processus, en plus de sa simplicité d'utilisation. Cependant, son inconvénient réside dans son impossibilité à mesurer la complexité des données ; elle ne se base que sur le flux de contrôle. En effet, cette métrique ne prend en compte que le nombre de décisions d'un modèle et donc ne donne que peu d'informations concernant sa structure.

Plusieurs autres métriques ont été proposées dans la littérature pour la complexité logicielles et furent par la suite adaptées au concept des processus métiers [SW03][WHH79]. Cependant, aucune de ces métrique ne prend en compte l'environnement cloud pour le déploiement et l'exécution des processus métiers lors du calcul de la complexité des modèles de processus. Par conséquent, aucune d'entre elles ne peut être appliquée directement à notre contexte.

2.2.3.5 Synthèse

Nous avons introduit dans cette partie les différentes notions liées à l'obfuscation d'une manière générale. Nous avons ensuite présenté les différentes dimensions que peut prendre l'obfuscation du code, et avons passé en revue les différents critères sur lesquels repose la mesure de la qualité de l'obfuscation apportée. Nous avons également présenté les différentes méthodes d'obfuscation des processus métiers existantes dans la littérature, qui reposent principalement sur l'obfuscation du code. Enfin, nous avons passé en revue les différentes mesures proposées ayant pour objectif de calculer leur complexité.

Nous nous appuyons au cours de cette thèse sur l'obfuscation basée sur la *séparation de la logique des données*. Nous tentons de préserver la logique de fonctionnement d'un processus en le subdivisant en plusieurs fragments, évitant ainsi de concentrer la logique du processus sur un seul et même fragment.

Concernant la mesure de complexité, nous nous basons sur la métrique *IC Interface Complexity* pour la proposition de notre propre métrique, qui selon nous s'adapte mieux à notre contexte. Cette métrique sera introduite et justifiée dans le chapitre 3.

Nous introduisons dans ce qui suit un état de l'art sur les problèmes de sécurité liés au cloud computing ainsi que sur les différents travaux effectués autour la sélection des services du cloud, l'externalisation, la décomposition et l'obfuscation des processus métiers.

2.3 Etat de l'art

2.3.1 Problèmes de sécurité liés au Cloud Computing

L'introduction du cloud computing a ouvert la porte à de nombreux défis et problèmes de sécurité. En effet, la complexité de son architecture a amené de sérieux problèmes de sécurité auxquels les fournisseurs doivent constamment faire face. Beaucoup de menaces résultantes de l'adoption du cloud sont énumérées dans [LGSS12]. Elles visent à mettre en garde les utilisateurs quant aux problèmes résultants de l'adoption du cloud. Selon une

enquête menée par l'IDC [12] en 2008 [Gen08], les principaux défis qui empêchent les organisations d'adopter le cloud computing concernent les problèmes de sécurité. Subashini et al. [SK11] présentent les principaux enjeux reliés au différents modèles du cloud, et se sont concentrés sur les défis fondamentaux en matière de sécurité : la sécurité du stockage des données, la sécurité de la transmission des données, la sécurité des applications et la sécurité reliée aux ressources des tierces parties. Jamil et al.[JZ11] présentent certaines menaces provenant d'internet qui affectent également le cloud computing, telle que l'attaque DDOS, et les attaques par injection SQL.

Dans le contexte des processus métiers, de nouveaux risques de sécurité ont vu le jour suite à l'adoption du cloud, risques que les entreprises n'ont jamais eu à traiter par le passé. Goettelmann et al. [GMG13] ont listé certains de ces risques comme suit [Eur09] :

- **Politique et risque organisationnels** : Des problèmes de dépendance et de perte de conformité peuvent se poser. En effet, il se peut que le changement de fournisseur soit trop cher, et que donc une entreprise se retrouve dépendante du fournisseur. Ou encore il est possible qu'il modifie les termes et les conditions de départ d'un service, provoquant ainsi la perte de conformité aux exigences de sécurité.

- **Risques techniques** : Comme par exemple une suppression inefficace des données, qui résulte donc en leur disponibilité pour d'éventuelles parties malveillantes en dehors du cycle de vie spécifiée dans la politique de sécurité.

- **Risques légaux** : Le changement des loies appliquées peut augmenter l'exposition à des risques de divulgation par exemple.

Par ailleurs, ils proposent un ensemble d'objectifs/besoins devant être réalisés afin d'assurer un déploiement sécurisé des processus métiers [GMG13]. Ces besoins peuvent être résumés comme suit :

- **Préservation du savoir faire** : Les entreprises ont besoin de garanties quant à leur savoir-faire, i.e. stratégie de prise de décision qui représente leur réelle valeur commerciale.

- **Confidentialité des données** : Étant donné que l'entreprise manipule les données privées de ses clients, elle a besoin de certaines garanties quant à leur protection e.g. les données qui sont supposées être supprimées le sont réellement.

- **Vérification de la confiance** : L'entreprise doit être en mesure de pouvoir vérifier si la confiance placée dans un cloud est bien méritée, i.e. s'il respecte réellement les garanties qu'il est supposé offrir.

2.3.1.1 Synthèse

Nous nous concentrons au cours de cette thèse sur la premier besoin *Préservation du savoir faire* lors d'un déploiement cloud. Dans cette optique, nous introduisons dans le chapitre 3 nos différentes approches qui visent à assurer aux entreprises les garanties nécessaires quant à la préservation de leur valeur commerciale.

12. International Data Corporation est une société américaine de recherche, d'analyse et de conseil, spécialisée dans les technologies de l'information, les télécommunications et le développement de logiciels

2.3.2 Sélection des services du cloud

L'évolution des technologies et l'expansion d'internet ont permis une croissance exponentielle quant au nombre de services offerts par les fournisseurs. Par conséquent, il devient souvent difficile pour les clients de sélectionner le service adéquat parmi l'ensemble des services offerts pour utilisation, ou à des fins de composition. En effet, il est crucial de sélectionner les services qui, en plus de répondre aux besoins fonctionnels du client, répondent à certains besoins non fonctionnels (cout, sécurité, risque, performances...), rendant ainsi le choix pour le client plus difficile.

Cependant, la nature dynamique du cloud implique des changements occasionnels et consciemment planifiés ; qui exposent la sélection des services du cloud à différents challenges. En effet, Garcia et al. [GGS13] soutiennent qu'il est important de mettre à jour les tableaux comportant les caractéristiques des ressources disponibles, étant donné que la politique des prix de la plupart des fournisseurs est déterminée par l'offre et la demande. Il affirment qu'il est aussi important pour un broker d'avoir toutes les informations mises à jour quant à la disponibilité des services afin d'offrir une sélection optimale. D'autres problèmes liés à la politique de sécurité des services peut engendrer des dépendances et conflits entre les services, rendant ainsi plus compliquée la composition de services [Str10].

Il est à noter qu'il existe une forte relation entre la sélection des services du cloud et des services web. En effet, les services SaaS sont principalement des applications web, les services PaaS sont fournis pour le développement de services Web et sont généralement accessibles via des interfaces Web. Les services IaaS offrent des environnements virtuels pour le déploiement de plates-formes et peuvent être surveillés par des outils de surveillance Web. Cependant, étant donné la nature du cloud, il n'est pas possible d'appliquer directement les méthodes de sélection des Web Services pour les services du cloud pour les raisons suivantes [SDH+14] :

1) Groupe d'utilisateurs ciblés : Les utilisateurs du cloud diffèrent selon les services ciblés. En effet, les services PaaS et IaaS sont souvent utilisés par des entreprises tandis que les services SaaS s'adressent à la fois aux entreprises et aux utilisateurs individuels.

2) Politiques de paiement plus flexibles : Dépendent directement du processus de consommation des services, e.g. payer pour ce qu'on utilise. Ceci rend le choix de l'utilisateur plus difficile.

3) Critères d'évaluation des performances plus importants : Certains critères qui évaluent les performances des services cloud ne sont pas pris en compte par les services Web e.g. l'habilitée à transférer des donnée d'un fournisseur de service à un autre.

Plusieurs approches ont été proposées dans la littérature autour de la sélection des services du cloud , que ce soit pour la sélection d'un seul service [SSZK12][ZZZ09] [MTG11] ou une composition de services [QRD14] [KSP12]. Parmi ces méthodes, nous retrouvons le travail de Gater et al. [GLGB13] qui propose un framework pour la sélection de services basés sur des critères fonctionnels et non fonctionnels. Les requêtes des utilisateurs sont exprimées comme un modèle de processus orné d'annotations exprimant leurs préférences et leurs exigences. Goettelmann et al. [GDGG14] ont proposé une solution pour équilibrer le coût du déploiement d'un processus métier dans un environnement cloud, avec le niveau de sécurité procuré par ces serveurs cloud. Leur solution est basée sur l'utilisation d'un brocker qui joue le rôle d'intermédiaire entre le client et le serveur. En effet, ce brocker prend en compte les besoins de l'utilisateur en terme de sécurité, analyse les différentes offres disponibles et réalise une évaluation du risque. Ainsi, en se basant sur cette éva-

luation de risque en plus des exigences fonctionnelles et des coûts, le brocker sélectionne les offres clouds pour le déploiement du processus métier en question. Après décomposition, les différentes configurations (tâches/cloud) sont sélectionnées en se basant sur une heuristique introduite dans [FDG10].

2.3.2.1 Synthèse

Nous avons introduit dans cette partie les différents problèmes de sécurités résultant de l'environnement cloud, ainsi qu'un ensemble de solutions existantes dans la littérature qui permettent de sélectionner des services du cloud, aussi bien pour une exécution simple que pour une composition de services.

Étant dans un contexte processus métiers où les informations transmises reflètent le savoir-faire de l'organisation, ces solutions ne peuvent être appliquées. En effet, lors du déploiement et de l'exécution des processus métiers dans un environnement cloud, il est primordial de sélectionner les services qui d'une part répondent au mieux à la politique de sécurité de l'organisation qui déploie le processus, mais offrent aussi un niveau de sûreté acceptable minimisant les fuites d'informations.

Bien que la solution de Goettelmann et al.[GDGG14] s'applique au même contexte que le nôtre, elle ne permet une gestion du risque que du côté cloud, i.e ne repose que sur les données du cloud pour la sélection des services qui répondent au mieux aux besoins de l'utilisateur en terme de sécurité. Par conséquent, si le cloud s'avère être malicieux, les données communiquées pourraient être incorrectes. Contrairement à leur approche, nous nous basons sur des données côté client pour la sélection des services du cloud. Nos méthodologies de sélection sont détaillées dans le chapitre 5.

2.3.3 Business Process Outsourcing (BPO)

2.3.3.1 Les objectifs de BPO

Traditionnellement, l'externalisation est l'abréviation de « l'utilisation de ressources externes » [GMLH15]. En effet, cela consiste à déléguer des tâches d'une entreprise à un prestataire indépendant appelé également « tierce partie ». En général, l'externalisation est considérée comme une pratique courante dans les entreprises grâce à ses importantes stratégies commerciales [RBBA16]. Dans le paradigme BPO, l'entreprise externalisant le processus est appelée « consommateur de service » et le service mettant en œuvre le processus métier externalisé est appelé « fournisseur de service ».

L'externalisation des processus métiers prend de plus en plus d'ampleur suite aux avantages qu'elle procure. BPO permet aux organisations d'adopter des solutions souples axées sur les services, impliquant généralement la sous-traitance d'opérations spécifiques des processus métiers à un service tiers [KS07] [WHFL04]. Le Groupe Gartner [13] la définit comme la délégation d'un ou de plusieurs processus métiers à un fournisseur externe, qui à son tour possède, administre et gère le processus sélectionné sur la base d'un critère de performance défini et mesurable. En effet, il s'agit d'un contrat dans lequel une entreprise cède une partie de son activité interne à une autre entreprise [MA04]. Les tâches transfé-

13. Gartner Inc. est une entreprise américaine de conseil et de recherche dans le domaine des techniques avancées. Elle mène des recherches, fournit des services de consultation, tient à jour différentes statistiques et maintient un service de nouvelles spécialisées (source : Wikipédia)

rées sont généralement non stratégiques et non essentielles, mais nécessaires pour le bon fonctionnement d'une entreprise.

Dans la littérature, plusieurs travaux se sont intéressés à l'externalisation des processus métiers. En effet, Arnold et al. [Arn00] présentent un modèle d'externalisation composé de quatre éléments : les sujets qui sous-traitent, les objets sous traités, les partenaires de sous-traitance et enfin la conception de la sous traitance. Les sujets qui sous-traitent représentent les entreprises qui prévoient d'externaliser (ou non) leurs processus métiers. Les objets sous-traités sont les processus ou parties de processus externalisés. Les partenaires de sous traitance représentent toutes les entreprises auxquelles seraient éventuellement externalisées des objets, la conception de sous traitance représente la stratégie d'externalisation (quels objets externaliser) [Arn00].

En effet, ayant directement évolué de l'externalisation des systèmes d'informations, qui selon les définitions les plus répandues représentent « *La contribution des fournisseurs externes aux ressources physiques et/ou humaines associées à des éléments entiers ou spécifiques de l'infrastructure informatique de l'organisation* » [LV92], le BPO s'en différencie par le fait que le sous traitant contrôle toutes les questions liées au processus métiers, aux ressources humaines ainsi qu'aux technologies.

Les plus gros avantages offerts par BPO sont représentés par une réduction de coûts, une habilité à se concentrer sur les parties importantes des processus métiers, permettre de faire appel à une expertise externe et une agilité à répondre aux éventuelles demandes de changements des clients [LOM+14]. Cependant, externaliser les processus peut induire à de sérieux risques de sécurité [WBWK08], qui peuvent se traduire par l'exploitation des parties sous-traitantes d'informations pertinentes au sujet des processus des entreprises clientes (les entreprises qui externalisent leurs processus) [ACR05]. Ce type de risque est considéré comme étant stratégique car il est causé délibérément par les entreprises sous traitantes. D'autres types de risques peuvent être considérés comme par exemple une défaillance dans les opérations au niveau du sous traitant. Ce type de risque n'est pas considéré comme délibéré mais est dû à la complexité des opérations à effectuer au niveau du sous-traitant, la séparation géographique entre l'entreprise cliente et le sous traitant ou encore la différence existante entre l'environnement de l'entreprise et du sous-traitant [ACR05]. Par conséquent, la décision de confier une partie du processus à un sous traitant est généralement prise en tenant compte de plusieurs facteurs en utilisant des modèles de décision multi-critères [LWF96].

2.3.3.2 Approches de décomposition et d'externalisation des processus métiers dans le cloud

Dans un contexte cloud considéré comme un environnement partagé, l'externalisation des processus métiers reste un sujet sensible pour les analystes du marché, les entreprises et les experts. Ceci est du au fait de la capacité du cloud à fournir des ressources informatiques qui sont dynamiquement évolutives, virtualisées et accessible à travers des services sur internet. Cependant, très peu de travaux ont été effectués autour de l'externalisation des processus métiers dans le cloud computing. En effet, à cause de ses problèmes de sécurité (comme introduit dans la section 2.3.1), les entreprises hésitent encore à l'adopter, malgré tous les avantages qu'il offre.

Par ailleurs, étant dans un contexte où différents clients accèdent aux mêmes plateformes et aux même informations, utiliser la décomposition permet de se prévenir contre toute éventuelle partie malicieuse. En effet, la décomposition des processus métiers per-

met de découper un processus d'une manière progressive en activités moins granulaires [CST10]. Elle contribue à la conception modulaire de larges systèmes, à la réutilisation de ses parties et à sa compréhension globale. Ainsi, les modèles de processus - qui traduisent les informations sur le fonctionnement d'une organisation- peuvent être décomposés en différents sous-processus avant déploiement dans le cloud. Ceci permet de réduire le taux d'information déployé sur chaque serveur cloud, réduisant ainsi la possibilité qu'un cloud puisse reproduire la logique du processus à lui seul.

Nous présentons dans ce qui suit un ensemble de travaux de décomposition et d'externalisation ayant pour objectif de sécuriser le processus lors du déploiement cloud.

Duipmans et al. [DPdSS14] ont proposé une solution pour décomposer un processus selon les différentes structures le composant (e.g. une activité simple, séquence d'activité, passerelles ou boucles). Ils proposent des solutions qui considèrent toutes les allocations possibles, i.e. locale ou sur le cloud. Leur décomposition est conduite par une liste de distribution, dans laquelle les activités du processus d'origine sont marquées avec leurs emplacements de distribution souhaités, en plus d'ajouter des restrictions de données afin de s'assurer que les éléments de données restent à l'intérieur d'un certain emplacement (en local ou sur cloud). Les résultats de leurs expérimentations ont indiqué que le comportement du processus lui-même n'est pas modifié par la transformation, et aucune information provenant du processus original n'est perdue pendant la décomposition.

Fdhila et al. [FYG09] ont proposé une solution pour transformer un processus métier de centralisé en plusieurs sous-processus ayant un comportement équivalent, qui interagissent entre eux à travers l'échange de messages asynchrones sans aucun contrôle centralisé. Leur méthode de décomposition est réalisée à travers les quartes étapes suivantes :

1. *Construction des tables de dépendances (DCDT Direct Control Dependency Table et DDDT Direct Data Dependency Table)* : Permettent de résumer toutes les dépendances de contrôle et de données entre les activités.

2. *Construction des tables TCDT (Transitive Control Dependency Tables)* : Cette étape considère le critère de décomposition pris en compte (i.e. selon les services, fournisseurs, organisations...). Par exemple, les activités invoquant le même service seront réunis dans le même sous processus dans le cas où le critère service est sélectionné. Ainsi, et afin de sauvegarder une part de la sémantique initiale dans chaque sous processus, les dépendances de contrôles transitives entre les activités invoquant le même services seront déduites à partir de la table DCDT. Par conséquent, les sous-processus seront construits et ils contiendront ces activités reliées à travers des dépendances transitives.

3. *Construction des sous-processus* : Les différents sous processus seront construits à partir des tables TCDT.

4. *Interconnections des sous-processus* : Cette étape connecte les différents sous-processus en se basant sur les dépendances de contrôles et de données pour la sauvegarde de la sémantique initiale du processus.

Bien que ces deux solutions soient adaptées pour la décomposition des processus métiers, elles ne proposent cependant aucune méthode pour formellement démontrer le niveau de sécurité apporté par leur approches.

Dans cette optique, dans la continuité des travaux de Fdhila et al. [Fdh11], Goettelmann et al. [GFG13] ont proposé une solution pour la décomposition des processus métiers dans

le but d'un déploiement cloud. Leur méthode de décomposition est basée sur celle de Fdhila et al. [FYG09] et prend en compte les niveaux de sécurité requis des tâches lors de la sélection des clouds adéquats. En effet, les tâches qui requièrent le plus de sécurité sont déployées sur les clouds de confiances. De plus, leur solution prend en compte un ensemble de contraintes de séparation et de co-localisation, qui imposent de séparer ou de co-localiser certaines tâches afin dé répondre aux exigences de sécurité imposées par l'architecture cloud.

Dans la même optique, Povoa et al. [PdSPdP14] ont proposé une approche pour décomposer les processus métiers partiellement dans le cloud en gardant une partie en local, tout en prenant en compte le coût, les performances et les contraintes sur les données afin d'assurer leur sûreté. Leur méthode est composée de quatre étapes (1)*Lifting* : un processus métier est transformé en modèle intermédiaire en tenant compte des variables et des flux de données entre les activités. Ce modèle intermédiaire nommé *(Graph-based Workflow Model(GWM))* est basé sur des graphes directs et permet de représenter des processus métiers définis en différents langages (BPMN 2.0, Another Workflow Language (YAWL),Web Services Choreography Description Language (WS-CDL),...etc.).(2) *Sélection* : cette étape permet d'obtenir la représentation la plus optimale du processus métier en modèle intermédiaire. Le calcul de la localisation optimale des activités et données associées (sur cloud ou en local) est effectué en se basant sur les politiques de sécurité définies, les coûts ainsi que les performances (temps de réponse, débit). (3) *Décomposition* : le processus métier représenté sous forme de modèle intermédiaire est décomposé en sous-processus selon les résultats obtenus dans la phase sélection (quelles activités mettre sur cloud ou en local) en appliquant un ensemble de contraintes qui permettent au processus décomposé d'adopter le même comportant que le processus original(4) *Grounding* : cette phrase permet de transformer les sous processus du modèle intermédiaire en sous processus représenté en langage de modélisation de processus (e.g. BPMN, YAWL.....etc.).

Similaire à l'approche de Goettelmann et al.[GFG13], Watson et al. [Wat12] ont proposé une méthode pour automatiquement déterminer les configurations (tâches/cloud) en se basant sur un ensemble d'exigences de sécurité. Ces exigences sont principalement basées sur le modèle *Bell-LaPadula multi-level access control* [BL73]. Leur méthode a démontré que la nécessité de transfert de données entre les clouds soulève de potentielles violations de sécurité qui doivent être résolues lors du choix des configurations. Contrairement à Goettelmann et al.[GFG13], les processus pris en compte ne comprennent que des tâches séquentielles et la donnée en entrée d'une tâche est toujours la sortie de la tâche précédente. Par ailleurs, les patrons de contrôle et les dépendances de données complexes ne sont pas traités.

Rekik et al. [RBBA15] proposent une solution pour sélectionner les tâches du processus à déployer sur le cloud en se basant sur leur niveau de dégradation des performances. En effet, ils estiment qu'il est plus efficace pour les entreprises d'externaliser ces tâches là. Cette solution est basée sur un outil de suivi qui permet d'identifier ces activités. Ils ont défini par la suite un modèle de décision utilisant le processus de hiérarchie analytique (AHP), pour aider les experts dans la tâche fastidieuse de sélectionner les activités appropriées à externaliser. Leur modèle de décisions prend en compte le coût de déploiement ainsi que la nature des tâches (apporte un avantage concurrentiel ou pas).

Par ailleurs, ils proposent dans [RBBA16] un framework d'externalisation nommé « Business Process Outsourcing to the cloud (BPO2C) » qui assiste les entreprises et les experts IT quant à la décision d'externaliser leurs processus. Ce framework couvre le cycle de vie d'externalisation des processus qui va des motivations de l'entreprise à externali-

ser, à l'identification des fragments considérés « externalisables ». En effet, ce framwork sélectionne les meilleurs fragments de processus ainsi que les clouds les plus appropriés permettant de minimiser les coûts et temps d'exécutions. Par ailleurs, ils permettent également de minimiser les risques auxquels ils s'exposent lors de l' externalisation.

2.3.3.3 Synthèse

Nous avons introduit dans cette partie les différents travaux liés à la décomposition et l'externalisation des processus métiers. Bien que beaucoup d'entre eux proposent des approches permettant de décomposer et d'externaliser des processus métiers dans un contexte cloud, ils ne s'adaptent pas tout à fait à notre contexte. En effet, ils ne prennent pas en compte la sécurité lors de leur déploiement. Par ailleurs, les travaux de Goettelmann et al.[GFG13] se rapprochent fortement de notre contexte ; leur décomposition prend en compte un ensemble de paramètres de sécurité ainsi que des contraintes de séparation/co-localisation lors de la sélection tâche/cloud. Cependant, leur solution ne prend pas en compte l'automatisation de la génération des contraintes de séparation, i.e. leurs contraintes sont introduites manuellement. De plus, ils ne prennent pas en compte lors du déploiement la nature des tâches, i.e.*sensibles ou pas*, ni la possibilité d'une coalition entre les clouds de la collaboration pour reconstituer le processus. Les différentes contributions proposées dans cette thèse tentent de répondre à ces limites en automatisant d'une part la génération des contraintes pour le déploiement des processus métiers, et d'autre part en tenant compte de la possibilité de coalition de clouds malicieux à des fins de reconstruction du processus.

Nous présentons dans la partie suivante le concept d'obfuscation en général, et plus précisément l'obfuscation des processus métiers dans le but d'un déploiement cloud *plus sécurisé*.

2.3.4 Approches d'obfuscation

2.3.4.1 Obfuscation du code

Plusieurs travaux ont été proposés dans la littérature et peuvent être classifiés selon ce qu'ils ciblent à obfusquer (structure, données, etc.)[CTL97][MADB+06] [CDPF+14][SLGL09] . Dans ce contexte, Ceccato et al. [CDPN+09] ont conduit un ensemble d'expérimentations afin de mesurer la puissance d'une obfuscation réalisée en utilisant la méthode *identifier renaming*. Cette méthode consiste à supprimer les informations pertinentes (par exemple remplacer des noms significatifs par des noms aléatoires dénués de sens) en utilisant deux types d'attaquants (novices et experts), et deux types d'attaques (compréhension et modification du programme). Ce type d'obfuscation cible la structure du code. Les résultats obtenus démontrent que la méthode réduit l'efficacité des attaques en augmentant le temps (le doublant) nécessaire afin d'obtenir une attaque réussie. D'autre part, ils démontrent aussi que les attaquants « experts » doivent fournir plus d'efforts afin de réussir leurs attaques.

Afin de se prévenir des décompilateurs et débogeurs, plusieurs travaux ont été introduit dans la littérature [BS05]. Dans ce contexte, Linn et Debray.[LD03] proposent une solution basée sur des *octet indésirables* introduits au milieu des autres instructions. Ainsi, tout désassembleur interprètera ces octets comme un début d'instructions, risquant ainsi de désassambler incorrectement les instructions qui succèdent ces octets indésirables.

Il est cependant à noter qu'il existe beaucoup de travaux effectués sur l'obfuscation de java Bytecode alors qu'il reste un sérieux manque sur l'obfuscation du code en C/C++. En effet, bien que C/C++ soient des languages très populaires dans le monde du developpement logiciels, e.g. dans les communités open source et systèmes embarqués, ils n'ont aucun mécanisme de sécurité supplémentaire telle que la protection contre les débordements des mémoires tampons [Cap12].

2.3.4.2 Obfuscation des processus métiers

Contrairement à l'obfuscation du code, très peu de travaux ont été effectués autour de l'obfuscation des processus métiers dans le cloud. En effet, externaliser les processus métiers dans le cloud apporte un avantage considérable pour les entreprises externalisantes (i.e. profiter des avantages du cloud) mais aussi pour les entreprises qui utilisent ces processus/ou parties de processus externalisés. Aussi, le cloud leur donne la possibilité de composer des services d'une grande variété de fournisseurs. Ainsi, en réutilisant des services déjà existants pour la création de leurs processus métiers, les entreprises économisent sur les coûts d'infrastructures mais aussi sur les délais de livraison. Cependant, comme déjà introduit, profiter des avantages du cloud expose à de sérieux problèmes de sécurité (section 2.3.1). Bentounsi et al. [BBDA12] soutiennent qu'il est important de cacher la provenance des fragments déployés dans le cloud afin de de préserver l'activité commerciale de l'organisation à qui appartient les fragments externalisés. En effet, étant donné un environnement partagé entre plusieurs utilisateurs, leurs données et informations privées sont exposés à des risques de divulgation. À cet effet, ils proposent un modèle d'anonymat pour les fragments considérés comme pertinents (i.e. lier ce type de fragment à un utilisateur donne directement une idée de l'activité commerciale de cet utilisateur) en générant des vues anonymes de ces fragments. Ces vues anonymes jouent le rôle de base de données et sont composées d'un ensemble de fragments clones des fragments dits « pertinents ». Ainsi, ne connaissant pas le nombre exact de clients l possédant les k clones, ils définissent la probabilité de découvrir la provenance du fragment comme k/l.

Duipmans et al. [DPdSS12] ont proposé une solution pour déployer les processus métiers dans le cloud en se basant sur une transformation et décomposition en sous processus, qui seront par la suite déployés partiellement dans le cloud en gardant la liberté de déployer les tâches sensibles et peu coûteuses à un niveau local. Bien que leur solution apporte un certain niveau de sûreté (grâce au déploiement des tâches sensibles localement), ils n'ont proposé aucune méthode pour formellement démontrer le niveau de sécurité apporté par leur décomposition.

Fill et al. [Fil12] ont proposé un ensemble de transformations directement dérivées de l'obfuscation du code, qu'ils classifient comme suit : (1) *transformation* représentative : qui consiste à cacher ou supprimer certaines informations dans le modèle obfusqué e.g. supprimer les identifiants et les types d'éléments en ne gardant simplement que les relations entre ces éléments.(2) *transformation de structure* : qui consiste à modifier la structure et la complexité des modèles afin de cacher leur fonctionnement. Ils proposent de cacher l'exécution de certaines tâches A et B par exemple, en les incorporant dans un sous processus sub_1. Ainsi, seule l'exécution du processus sub_1 sera visible sans divulguer son contenu, cachant ainsi l'exécution des tâches A et B. Ils proposent aussi d'intégrer les mêmes parties d'un processus dans différents sous-processus. Ainsi, l'appel à cette même partie se fera durant l'exécution du processus en exécutant différents sous-processus, donnant ainsi l'illusion d'exécuter différentes parties.(3) *transformation de données* : cette transforma-

tion se base sur les méthodes appliquées pour la préservation de la vie privée dans les fouilles de données. Elle consiste en l'obfuscation d'attributs d'activités e.g. obfusquer le temps d'exécution d'une activité en l'arrondissant, ou obfusquer le coût d'exécution en donnant une valeur x représentant le coût et appartenant à un intervalle.(4) *transformation sémantique* : basée sur les propriétés des ontologies formelles, cette transformation utilise les hiérarchies de subsomption pour anonymiser les informations, en remplaçant les valeurs des attributs avec une valeur moins spécifique dans un sens plus généralisé. Par exemple, représenter l'attribut « client » par sa valeur généralisée « personne » permet de l'anonymiser.

2.3.4.3 Synthèse

Nous avons présenté dans cette partie les différentes solution proposées dans la littérature pour l'obfuscation du code en premier lieu et des processus métiers en second lieu. Seul peu de travaux se sont consacrés à l'obfuscation des processus métiers et la plupart d'entre eux ne tiennent pas compte de l'environnement cloud lors de l'obfuscation de leur processus.

Nous proposons au cours de cette thèse une solution permettant d'obfusquer un processus dans un environnement cloud, i.e. cacher sa logique de fonctionnement en le décomposant en plusieurs fragments qui seront déployés sur différents clouds. Notre méthodologie est détaillée dans le chapitre 3.

2.4 Conclusion

Dans ce chapitre, nous avons introduit les trois grands concepts de cette thèse, à savoir le cloud computing, les processus métiers et l'obfuscation. En effet, cette thèse peut être considérée comme se situant à l'intersection de ces trois grands domaines.

Après introduction des différents concepts nécessaires à la compréhension de nos travaux, nous avons présenté un état de l'art autour des travaux effectués pour d'une part sécuriser l'environnement cloud et d'autre part sélectionner ses meilleurs services selon les besoins de l'utilisateur.

Nous avons ensuite présenté un ensemble de travaux effectués autour de l'externalisation des processus métiers d'une manière générale, et plus particulièrement autour de leur externalisation dans un environnement cloud. Nous avons également présenté le peu de travaux effectués autour de l'obfuscation des processus métiers. Seul très peu d'entre eux prennent en compte l'environnement cloud lors de l'obfuscation. Une partie de ce chapitre fut dédiée au différentes mesures de complexités introduites dans la littérature. Bien que la plupart de ces mesures furent proposées pour mesurer la complexité logicielles, bon nombre d'entre elles furent adaptées pour la mesure de la complexité des processus métiers, mais ne s'adaptent cependant pas directement à notre contexte.

Le prochain chapitre est dédié à notre première contribution qui consiste à obfusquer un modèle de processus en augmentant sa complexité. Dans un premier temps, on éclate le processus en fragments de processus en séparant les fragments sensibles dans différents clouds. Dans un second temps, cette solution est améliorée par l'introduction de *faux fragments* à certains endroits *stratégiques*, afin de réduire la probabilité d'une éventuelle coalition initiée par un cloud malicieux.

Deuxième partie

Contributions

Chapitre 3

Obfuscation d'un processus métier

Sommaire

3.1 Introduction

L'introduction du cloud computing et son alliance à la technique de gestion des processus métiers (BPM) a apporté de nombreux avantages stratégiques et financiers, ayant encouragé les entreprises à l'adoption du cloud. Cependant, étant donné que les processus métiers expriment le savoir-faire des organisations, i.e. certains fragments considérés *critiques* comprennent d'importantes informations sur les stratégies adoptées par les entreprises, il est nécessaire de fournir aux entreprises les garanties nécessaires quant à la protection de ces informations. En effet, pour que les entreprises envisagent de déployer leurs fragments dans un environnement cloud, elles doivent être sûres que leurs activités stratégiques et commerciales seront préservées.

Nous présentons dans ce chapitre la première contribution de cette thèse, qui concerne l'obfuscation des processus métiers dans le but d'un déploiement plus sécurisé dans un environnement cloud. L'obfuscation étant considérée comme l'action de transformer un programme en un autre programme ayant les mêmes fonctionnalités, mais étant plus complexe à comprendre. Plus précisément, nous présentons dans ce chapitre deux approches d'obfuscation complémentaires. Mais avant de présenter ces techniques d'obfuscation, nous présentons dans la section 3.2 l'architecture générale sur laquelle sont basées ces approches. La section 3.3 introduit notre première approche à travers un algorithme d'obfuscation basé sur la notion de *fragment critique*. Cet algorithme est par la suite amélioré dans la section 3.4 à travers l'introduction de faux fragments à certains endroits considérés *stratégiques*.

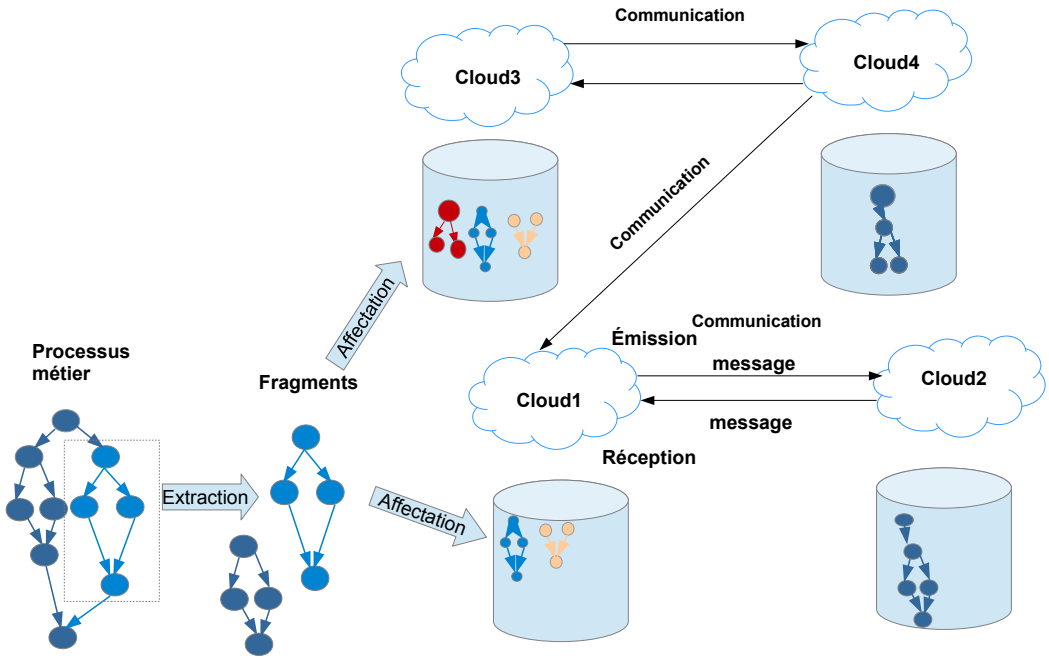


FIGURE 3.1 – décomposition du processus en fragments et déploiement sur plusieurs clouds

3.2 Approche globale

Avant de présenter notre approche d'obfuscation, nous introduisons dans ce qui suit les principes ainsi que l'architecture sur lesquels nous nous sommes basés au cours de cette thèse, pour l'obfuscation des processus métiers dans le but d'un déploiement dans un environnement cloud.

3.2.1 Déploiement d'un processus comme une collaboration de fragments dans un multi-cloud

Plusieurs moyens d'obfuscation ont été proposés dans la littérature (voir section 2.2.3.3). Parmi ceux-ci, nous nous sommes intéressés dans cette thèse à la méthode qui **sépare la logique du processus en différents BP fragments** pour exécuter le processus comme une collaboration de fragments. En effet, en décomposant le processus en plusieurs sous-processus ou fragments, et en les déployant dans un environnement multi-cloud [14] (formant ainsi une collaboration de clouds)(voir figure.3.1), chaque cloud n'aura qu'une vue partielle de la logique générale du processus. Cette approche empêche par conséquent tout éventuel cloud malicieux de reconstruire à partir des informations se trouvant à son niveau, la logique du processus.

Cependant, et comme soulevé dans la question **Q1** de la section 1.1, automatiser l'extraction de ces fragments, c'est-à-dire être capable de les caractériser syntaxiquement s'avère être une tâche difficile. Ainsi, l'objectif primaire de notre contribution est d'identifier ces fragments sensibles pour être capable de les assigner à différents clouds en maintenant par ailleurs la logique générale du processus.

Ce déploiement permettra de minimiser le taux d'informations sensibles détenue par chaque cloud, les empêchant ainsi de reconstruire la logique générale du processus.

3.2.2 Architecture générale

Avant d'entrer dans les détails de notre approche, nous présentons dans ce qui suit, l'architecture sur laquelle est basée notre méthode de déploiement de processus métiers dans un contexte cloud (figure 3.2). Cette dernière clarifie le rôle des différents acteurs participant à son déploiement, à savoir le client, le brocker et le serveur cloud.

Le client est responsable de l'identification et de la conception du modèle de processus, qui sera transmis conjointement avec les besoins en terme **d'exigences fonctionnelles** et **de sécurité** au brocker. **Le brocker** se basera par la suite sur cet ensemble de besoins et les différentes descriptions de services qu'il aura au préalable reçu des serveurs cloud, pour la **sélection** du service qui correspond au mieux à chaque tâche en termes d'exigences fonctionnelles et de QoS. **Les serveurs** cloud quant à eux sont responsables de la description des offres et de la mise en œuvre du processus.

- **Le client :** Dans le contexte de déploiement et d'exécution des processus métiers dans un environnement cloud, le client est responsable de l'initiation du processus en publiant une version initiale du modèle, accompagnée de l'ensemble de ses exigences fonctionnelles et non fonctionnelles. En effet, comme illustré dans la figure 3.2, le

14. un ensemble de clouds qui collaborent afin d'exécuter le processus

client est chargé de la modélisation du processus à travers un outil de modélisation (e.g. BPMN) et la description des différentes exigences relatives à son déploiement.

En plus des contraintes de sécurité imposées a priori par le client, des exigences de sécurité seront obtenues à travers un algorithme d'obfuscation (section 3.3). Cet algorithme génèrera automatiquement un ensemble de contraintes additionnelles auxquelles devra répondre la collaboration de fragments obtenue suite au déploiement du processus. Par ailleurs, certaines incohérences entre les contraintes imposées par le client et celles générées par l'algorithme d'obfuscation peuvent exister (voir section 3.3.2.4).

Ces contraintes de sécurité permettent la sélection d'une configuration (affectation des tâches aux clouds), et peuvent être considérées en quelques sortes comme des exigences s'appliquant au niveau logique [15] étant donné qu'elles impactent le flot de contrôle du processus. Parmi ces exigences, nous listons dans ce qui suit une liste représentative de ces règles :

- **Séparation de fragments** : Cette règle impose la séparation de deux fragments f_i et f_j sur différents clouds. Son objectif est de diviser les connaissances et les informations au sujet du processus, pour éviter d'avoir un trop grand taux d'informations sur un seul et même cloud. Elle est formellement définie comme suit :

$$separate(t_i, t_j) = \begin{cases} 1 & si\, C(t_i) \neq C(t_j) \\ 0 & sinon \end{cases}$$

  Où $C(t_i)$ et $C(t_j)$ sont les clouds qui déploient les tâche t_i et t_j respectivement.
- **Co-localisation de fragments** : Cette règle impose l'exécution de deux fragments sur le même cloud dans le but d'améliorer l'intégrité des données. Ceci est possible en groupant sur le même cloud dans lequel on a une forte confiance, les tâches ayant une certaine valeur, i.e. contenant des informations importantes. Elle est formellement définie comme suit :

$$colocate(t_i, t_j) = \begin{cases} 1 & si\, C(t_i) \neq C(t_j) \\ 0 & sinon \end{cases}$$

  Où $C(t_i)$ et $C(t_j)$ sont les clouds qui déploient les tâche t_i et t_j respectivement.
- **Rétention de fragments en local** : cette règle impose de garder certains fragments localement. En effet, le client peut considérer que l'environnement cloud est trop dangereux pour certaines tâches de son processus.
- **Niveaux de confiance des clouds** Cette règle impose un certain niveau de confiance pour les clouds qui déploient le processus. Plus précisément, le client peut considérer que certaines tâches importantes doivent être déployées sur des clouds de confiance, tandis que d'autres peuvent l'être sur des clouds moins sûrs (mais aussi moins coûteux).

L'algorithme d'obfuscation quant à lui comprend l'ensemble des étapes permettant la transformation d'un processus P, en un processus $O(P)$ avec une structure (flots

15. Les contraintes logiques peuvent aussi être appelées contraintes fonctionnelles car elles impactent le flot de contrôle du processus. Elles peuvent modifier la logique du processus en introduisant des tâches additionnelles et/ou flots de contrôle

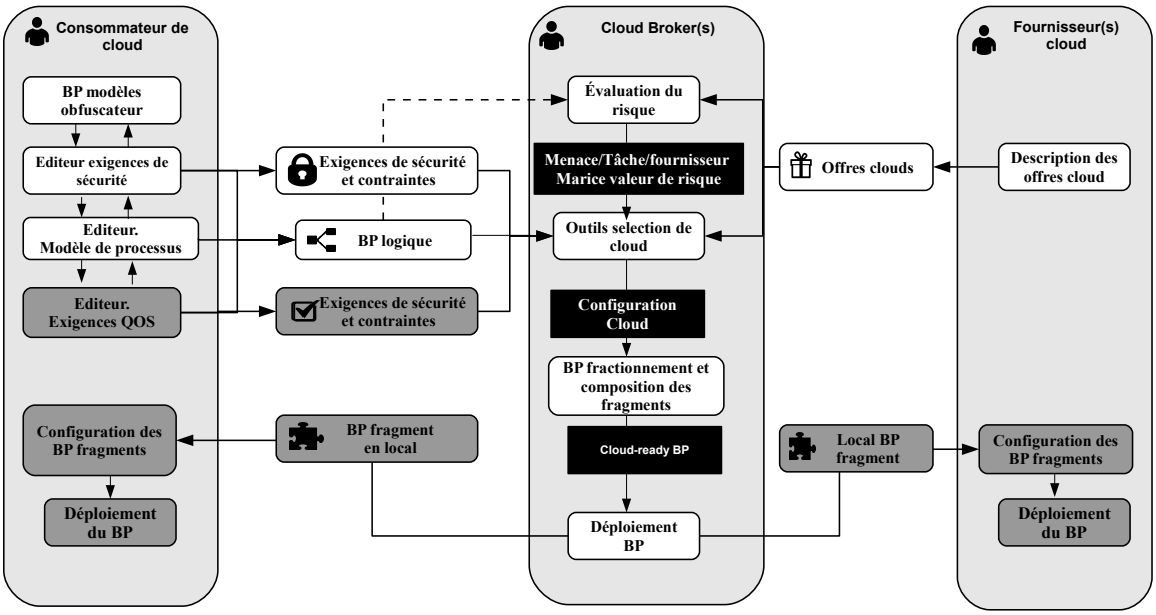


FIGURE 3.2 – Architecture générale

de contrôle) plus complexe et plus sécurisée pour un déploiement cloud. Cet ensemble d'étapes permet la génération de *contraintes de sécurité*, qui permettront de protéger la logique du processus dans un environnement multi-cloud.

- **Le brocker** Comme introduit dans la section 2.2.1.3, le rôle principal du brocker est d'être un intermédiaire entre les fournisseurs et les consommateurs du cloud. Essentiellement, il permet la sélection des configurations adéquates à partir des informations (modèle, contraintes fonctionnelles et non fonctionnelles) reçues du client. Dans notre architecture, le brocker permet d'assigner les fragments de processus (BP) aux différents serveurs clouds après une évaluation du risque, obtenue à partir des besoins formulés en terme de sécurité. Cette assignation peut être vue comme un problème d'optimisation multi-critères tentant de maximiser les performances et de minimiser les coûts et les risques de sécurité. Ainsi, les résultats d'obfuscation introduits dans ce chapitre représentent un ensemble de contraintes QoS/sécurité qui seront en entrée d'un problème d'optimisation résolu par le brocker, qui permettra l'assignation des BP fragments aux différents clouds de la collaboration. Il est à noter que lors de l'assignation des différentes tâches, le brocker vérifie le cohérence de l'ensemble des contraintes, i.e. plus précisément la cohérence entre les contraintes manuellement générées par le client et automatiquement générées par l'algorithme d'obfuscation.

  Par ailleurs, afin de sauvegarder la logique générale du processus, les différents liens reliant les fragments seront sauvegardés à travers l'utilisation de messages de synchronisations *send* et *receive* [Fdh11], établissant ainsi un lien de communication entre les différents clouds.

- **Le serveur cloud** Le rôle du serveur cloud est d'offrir les différents services permettant l'exécution des processus métiers. Ainsi, étant donné que les serveurs ont généralement accès à l'implémentation du process (ou au moins à une partie de celui-ci), il devient nécessaire d'implémenter les contrôles adéquats pour les différents problèmes de sécurités qui peuvent se poser (intégrité, confidentialité,...etc). Il est cependant à noter que généralement, les services décrits par les fournisseurs de clouds se focalisent principalement sur les besoins fonctionnels, i.e. les contrats entre les consommateurs et les fournisseurs ne se résument qu'à la considération des performances et des coûts, mettant de côté les besoins en termes de sécurité. En effet, seul peu de travaux prennent en compte la dimension sécurité lors d'un déploiement cloud [Clo13] [GDG+14].

3.3 Approche d'obfuscation basée sur la séparation des tâches sensibles

Cette section introduit notre première contribution qui consiste en l'obfuscation des processus métiers en se basant sur la séparation des tâches sensibles. Ainsi, nous introduisons d'abord cette notion de tâches sensibles et présentons ensuite notre algorithme d'obfuscation, qui décrit les différentes étapes nécessaires à la protection du savoir du processus lors d'un déploiement dans un environnement multi-cloud. Enfin, nous appliquons notre approche sur l'exemple de motivation introduit dans le chapitre 1.

3.3.1 Tâches sensibles

Nous introduisons dans ce qui suit les types de tâches que nous considérons comme sensibles, et qui représentent le pilier sur lequel se base notre approche d'obfuscation. Nous considérons ces tâches comme celles qui comprennent le plus de savoir-faire sur la logique du processus ; ce sont celles où sont prises les décisions et où sont faites les synthèses.

Aussi, après analyse de plusieurs processus métiers provenant de différentes sources dans la littérature [Bri13] [DLRM+13] [Fdh11] [LMS14] [Wes12], nous caractérisons ces fragments sensibles comme suit :

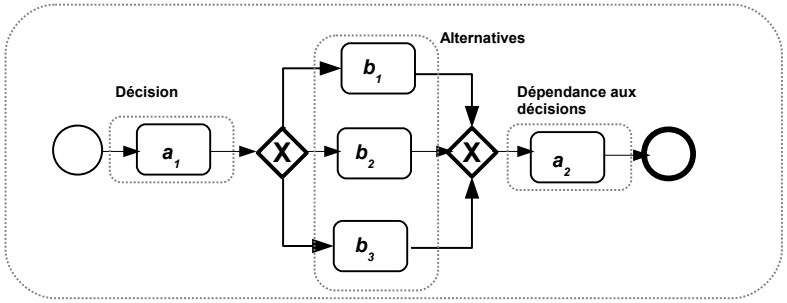


FIGURE 3.3 – Processus de prise de décisions

- **Decisions** Suite à notre analyse, nous avons établi une relation entre les décisions et les passerelles ouvrantes $X(OR)$ *split* qui déclenchent différentes alternatives dans le

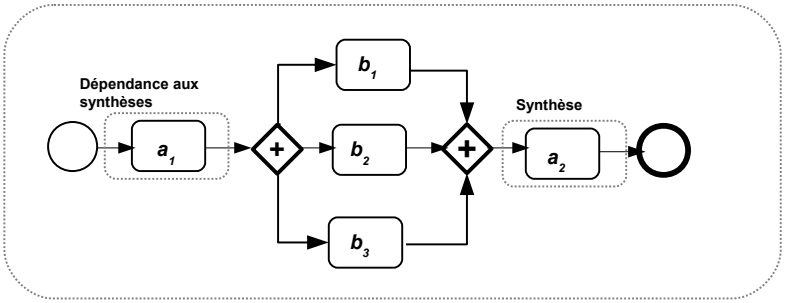


FIGURE 3.4 – Processus représentant une synthèse

processus. En effet, le lien établi suggère que les décisions sont prises dans les activités précédant ces passerelles. Ainsi, ces activités sont appelées *tâches de décision.*

- **Synthèses** Notre analyse nous a également permis d'établir un lien entre les synthèses et les passerelles fermantes AND join qui synchronisent différentes alternatives du processus. Cette relation suggère que les synthèses sont faites dans les activités qui suivent ces passerelles. Ainsi, ces activités sont appelées *tâches de synthèses.*

- **Dépendance de décisions**

  Notre analyse a révélé que les décisions prises dans les activités précédant les passerelles $(X)OR$ split, sont souvent complétées par des actions dans les activités qui suivent la passerelles fermantes $(X)OR$ join. De ce fait, nous considérons qu'il existe une dépendance entre ces deux fragments considérés comme **complémentaires**. Cette dépendance est appelée *dépendance de décision* et est illustrée dans la figure 3.3.

- **Dépendance de synthèses** Respectivement, nous avons conclu que les synthèses faites dans les activités qui suivent les passerelles AND join, sont généralement préparées dans les activités qui précédent la passerelles AND split. De ce fait, nous considérons qu'il existe une dépendance entre ces deux fragments dits **complémentaires** appelée *dépendance de synthèse* (voir figure.3.4).

- **Fragments alternatifs** Notre analyse a également révélé que les décisions peuvent souvent être découvertes en analysant les différentes alternatives provenant d'une passerelle $(X)OR$ split. En analysant et comparant ces différents flots, un attaquant potentiel peut obtenir d'importantes informations concernant les décisions stratégiques des organisations. De cet fait, nous considérons qu'il existe entre ces différentes alternatives une dépendance appelée *fragments alternatifs* (figure 3.3).

3.3.2 Algorithme d'obfuscation

Comme introduit précédemment, le rôle de notre algorithme d'obfuscation n'est pas d'affecter directement une tâche à un cloud, mais de générer des contraintes de séparations/niveaux de confiance, permettant ainsi de déployer le processus en toute sécurité.

L'algorithme prendra donc en entrée un processus métier, et génèrera à partir des tâches sensibles de ce dernier, l'ensemble des contraintes à appliquer [NGY+18].

Notre algorithme d'obfuscation (figure 3.5) peut se résumer à travers les étapes suivantes :

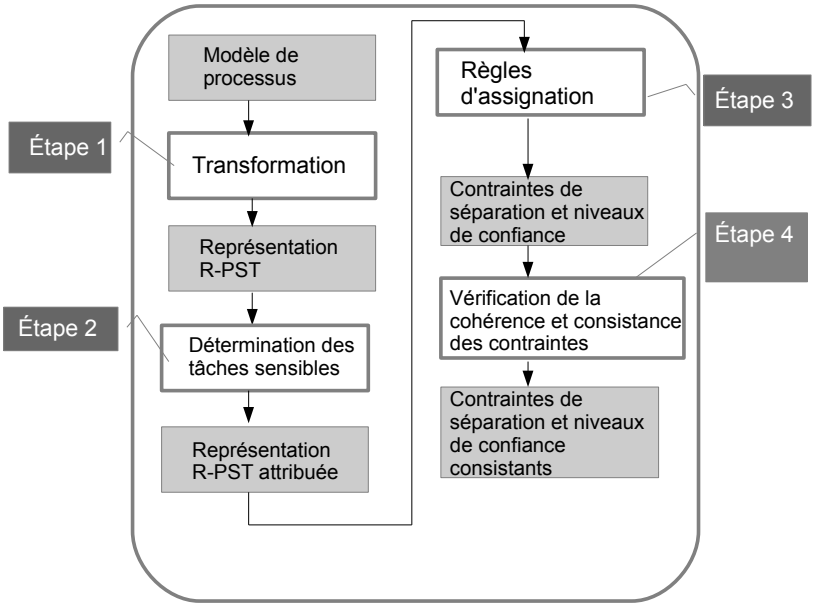


FIGURE 3.5 – Algorithme d'obfuscation par séparation de tâches sensibles

- **Étape 1 :** La première étape consiste à transformer notre modèle de processus en arbre R-PST en utilisant la méthode *Refined Processus Structure Tree* (section.2.2.2.3). Ceci se fait en transformant dans un premier temps le processus sous forme de blocs canoniques, qui nous permettent par la suite la construction de notre arbre R-PST.
- **Étape 2 :** La seconde étape consiste à déterminer d'abord les tâches sensibles en se basant sur les hypothèses introduites dans la section 3.3.1. Ensuite, un ensemble d'attributs pour la représentation de ces tâches sensibles (décisions et synthèse) sont ajoutés à notre arbre en utilisant la technique de *grammaire attribuée* [Knu90].
- **Étape 3 :** Cette étape consiste à appliquer un ensemble de règles (voir section 3.3.2.2) sur les fragments *critiques*, qui permettent de générer l'ensemble des contraintes ainsi que les niveaux de confiance nécessaires à un déploiement sécurisé.
- **Étape 4 :** La dernière étape consiste à vérifier la cohérence entre les règles générées par l'obfuscateur et les règles générées par le designer, qui peuvent être conflictuelles.

Nous détaillons dans ce qui suit ces étapes.

3.3.2.1 Construction de l'arbre R-PST attribué

Comme introduit précédemment, la décomposition canonique permet d'obtenir une décomposition unique et modulaire. En effet, utiliser la structure R-PST tree nous permet

de simplifier la caractérisation et la visualisation des décisions et synthèses introduites dans la section 3.3.1. Nous utilisons dans ce qui suit la technique des *grammaires attribuées* pour décorer la structure R-PST tree. L'utilisation d'attributs pour représenter nos *Décisions*, *Synthèse*, $fragments\ alternatives$, *Dépendance de décisions*, *Dépendance de synthèses* simplifie considérablement leur caractérisation.

La figure 3.6 représente la décomposition d'un processus en arbre R-PST à travers les différents blocs canoniques qui le composent. Chaque noeuds de l'arbre est soit un fragment soit un bloc. Les nœuds blocs diffèrent selon leur type comme suit :

- **Type de nœud :** un bloc est étiqueté avec un type :

$$type : \mathbf{node} \rightarrow \begin{cases} \text{“}X\text{”}, \text{si c'est un bloc de decision} \\ \text{“} + \text{”}, \text{si c'est un bloc de synthèse} \\ \text{“}seq\text{”}, \text{si c'est un bloc de sequence} \\ \text{“}autre\text{”}, \text{sinon} \end{cases}$$

- **Criticité d'un fragment :** l'attribut criticité exprime si un fragment est une *décision* ou bien une *synthèse*.

$$criticality : \mathbf{node} \rightarrow \begin{cases} \text{“}d\text{”}, \text{s'il s'agit d'une décision} \\ \text{“}s\text{”}, \text{s'il s'agit d'une synthèse} \\ \text{“}autre\text{”}, \text{sinon} \end{cases}$$

  Ses valeurs sont définies comme suit :

  - **Fragment décision :** Dans le modèle R-PST tree, le frère gauche d'un nœud de type = "X" est un fragment *décision*. Sa criticité aura donc la valeur "d". Soit n un nœud d'un arbre R-PST, $type(n) = \text{”}X\text{”} \rightarrow criticité(frère\_gauche(n)) = \text{”}d\text{”}$
    Dans la figure 3.6 -(c), le fragment F2 est un fragment *décision*.
  - **Fragment synthèse :** Dans le modèle R-PST tree, le frère droit d'un nœud de type = "+" est un fragment *synthèse*. Sa criticité aura donc la valeur "s". Soit n un nœud d'un arbre R-PST, $type(n) = \text{”}+\text{”} \rightarrow criticité(frère\_droit(n)) = \text{”}s\text{”}$
    Dans la figure 3.6 -(c), le fragment F9 est un fragment *synthèse*.
  - **Dépendance de décision** Dans un modèle R-PST tree, il existe une *dépendance de décisions* entre le *frère_gauche* et le *frère_droit* d'un nœud de type "X". Soit n un nœud d'un arbre R-PST,

$$type(n) = \text{“}X\text{”} \rightarrow dépendance\ de\ décision(\ frère\_gauche(n),\ frère\_droit(n))$$

    Dans la figure 3.6 -(c), il existe une dépendance de décision entre les fragments F2 et F8.
  - **Dépendance de synthèse** Dans un modèle R-PST tree, il existe une *dépendance de synthèse* entre le *frère_gauche* et le *frère_droit* d'un nœud de type

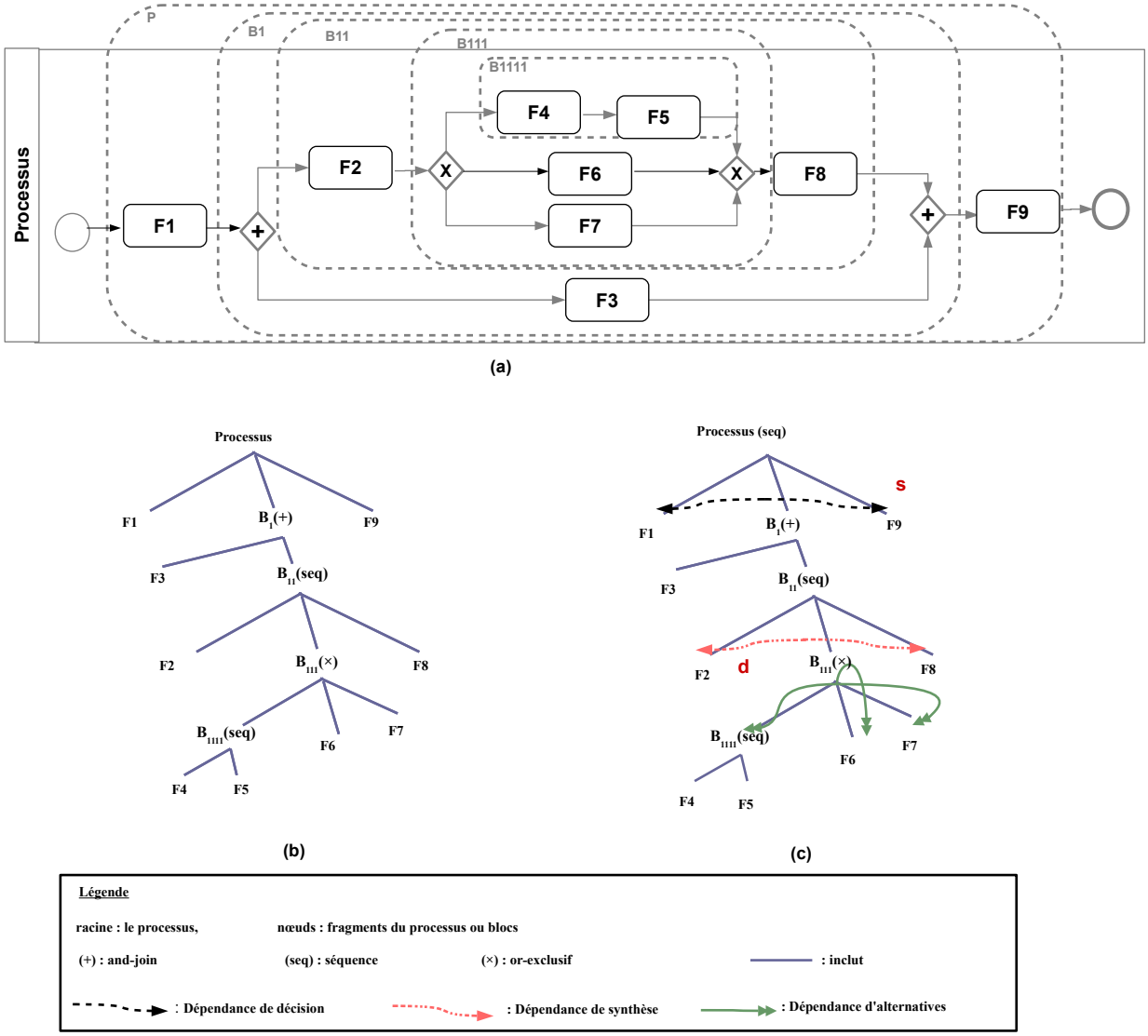


FIGURE 3.6 – (a) : processus décomposé en blocs canoniques (b) : Processus en R-PST, (c) : Processus R-PST décoré

"+". Soit n un nœud d'un arbre R-PST,

$$type(n) = \text{“} + \text{”} \rightarrow \textit{dépendance de synthèse}(frère\_gauche(n), frère\_droit(n))$$

Dans la figure 3.6 -(c), il existe une dépendance de synthèse entre les fragments F1 et F9.

– **Dépendance d'alternatives** Dans un modèle R-PST tree, il existe une *dépendance d'alternatives* entre tous les nœuds fils d'un nœud de type "X". Soit

n un nœud d'un arbre R-PST,

$$type(n) = \text{“}X\text{”} \rightarrow d\acute{e}pendance\ d'alternatives(fils(n)).$$

Dans la figure 3.6 -(c), il existe une dépendance d'alternatives entre les fragments F6, F7 et le bloc B_{1111}.

3.3.2.2 Règles pour l'assignation des activités de décisions et des activités synthèses

En accord avec l'approche générale introduite précédemment, nous introduisons cet ensemble de règles, permettant de répondre à un ensemble de besoins non fonctionnels (en termes de sécurité prioritairement), qui prennent en compte la nature des tâches.

- **règle 1 (décisions (d)) :** Cette règle concerne les fragments sensibles de type *décisions*, i.e. les fragments qui précèdent les passerelles ouvrantes $X(OR)$. Étant donné leur sensibilité, il est primordial de les assigner à des clouds ayant un niveau de confiance considéré comme acceptable. Ainsi, nous proposons d'assigner ces fragments décisions à des clouds ayant un niveau de confiance au moins égal à l_1 [16].

- **règle 2 (synthèses (s)) :** Respectivement, étant donné un fragment de type *synthèse* qui comporte d'importantes informations, nous proposons de l'assigner à un cloud ayant un niveau de confiance au moins égal à l_2.

- **règle 3 (Dépendance de décisions) :** Il s'agit d'assigner à différents clouds le fragment décision et le fragment qui complète la décision. En effet, comme expliqué précédemment (section 3.3.1), ces fragments sont considérés comme complémentaires, i.e. il existe une dépendance entre eux, permettant à un cloud les déployant simultanément de reconstruire la logique du processus. Il est donc nécessaire de les séparer.

- **règle 4 (Dépendance de synthèses) :** Il s'agit d'assigner à différents clouds le fragment qui succède la passerelle fermante AND et le fragment qui précède la passerelle ouvrante AND. En effet, identiquement aux fragments décisions, les fragments synthèses sont complémentés par les fragments précédents la passerelle ouvrante AND. Par conséquent, déployer ces fragments complémentaires sur différents cloud est primordial.

- **règle 5 (Fragments alternatifs) :** Nous assignons à différents clouds les différents choix sur lesquels se base une prise de décision, dans le but d'éviter qu'un cloud puisse deviner la stratégie de prise de décisions à partir de ces différents choix. Ainsi, il est primordial de déployer sur différents clouds les fragments alternatifs du processus en question.

16. Niveau de confiance de cloud considéré suffisant pour le déploiement d'une tâche sensible de type décision

3.3.2.3 Contraintes de séparation et niveaux de confiance

Les différentes contraintes (séparations/niveaux de confiances) sont obtenues à partir des règles d'assignation appliquées aux fragments sensibles (section 3.3.2.2) et peuvent être résumées comme suit :

1. $\forall F_i \in F$ [17], *si criticité*$(F_i) = d \vee$ *criticité*$(F_i) = s$ alors $(Niv_{confiance}(c_i)) > l$

   Une contrainte sur le niveau de confiance $(Niv_{confiance}(c_i)$ [18]$) > l$ est assignée à tout fragment critique ayant la valeur « d » ou « s » . En effet, étant des fragments sensibles, le processus de déploiement doit leur assurer que les clouds qui les déploient aient un certain niveau de confiance. Ces niveaux de confiance sont calculés selon une référence globale pour tous les serveurs de cloud [GDG+14].

2. $separate(f_i, f_j)$ si f_i et f_j sont complémentaires

   une contrainte de séparation est définie pour toutes les tâches complémentaires dépendantes. En effet, en appliquant la règle 3 pour les décisions et la règle 4 pour les synthèses, nous obtenons les contraintes permettant d'éviter de déployer trop d'informations pertinentes sur le même serveur cloud.

3. $separate(f_i, f_j)$ si f_i et f_j sont des alternatives

   une contrainte de séparation est définie pour chaque couple (f_i, f_j) appartenant aux différentes alternatives à travers l'application de la règle 5. Respectivement, cette règle permet également d'éviter d'avoir beaucoup d'informations sur le même cloud.

3.3.2.4 Cohérence des contraintes

L'objectif de cette étape est de vérifier la cohérence entre les contraintes automatiquement générées ainsi que les contraintes définies par le designer. En effet, il est primordiale d'éviter la contradiction de ces contraintes e.g. $separate(f_i, f_j)$ et $colocate$ [19](f_i, f_j) (il est impossible de séparer et co-localiser deux tâches en même temps).

En théorie, et grâce à nos hypothèses de modélisation, et plus précisément le principe SESE (Single Entry/Single Exit), il ne devrait exister aucune incohérence entre les contraintes générées automatiquement.

Cependant, étant donné que le designer peut prendre certaines décisions stratégiques et définir des contraintes manuellement, il est nécessaire de vérifier que ces contraintes soient en accord avec les contraintes générées par l'obfuscateur, i.e. vérifier qu'il n'y a pas de contradictions entre les deux types de contraintes (manuellement et automatiquement générées).

Par exemple, le designer ayant la possibilité de conserver un fragment sensible ou juste une partie de ce fragment à son niveau, certaines incompatibilités quant aux contraintes en relation avec les niveaux de confiances ainsi que les contraintes de séparation peuvent se poser.

- **Inconsistances des contraintes sur les niveaux de confiances :** Dans le cas où l'obfuscateur et le designer proposent différents niveaux de confiances pour le même fragment.

17. F est l'ensemble des fragments du processus
18. c_i représente le cloud qui déploie la tâche i
19. Déployer les tâches sur la même platforme (localement ou sur un serveur cloud).

Par exemple pour un fragment F_i, les niveaux de confiances proposés sont comme suit :

$$\begin{cases} trust\_level(F_i)_{designer} > l_1 \\ \wedge \\ trust\_level(F_i)_{obfuscateur} > l_2 \\ \wedge \\ l_1 < l_2 \end{cases}$$

- **Inconsistances des contraintes de séparation :** Peut se poser dans le cas où l'outil propose une contrainte $separate(f_i, f_j)$ pour deux tâches, alors que le designer préconise plutôt de les co-localiser ($colocate(f_i, f_j)$) pour des raisons de sécurité (i.e. garder en local des fragments considérés critiques), ou encore pour répondre à des besoins en terme de coûts ou de performances.

  – **En cas de conflit** Il est préférable que le choix final revienne au designer. En effet, nous considérons que le designer reste le plus à même à fixer les contraintes à employer, compte tenu de sa connaissance du processus et de la sensibilité des différentes tâches. Ainsi, lorsque l'outil propose une contrainte $separate(f_i, f_j)$ alors que le designer propose une contrainte $colocate(f_i, f_j)$, la contrainte « colocate » sera sélectionnée.

3.3.3 Application de l'algorithme à l'exemple de motivation

Nous appliquons dans ce qui suit notre méthode d'obfuscation par séparation de tâches sensibles à l'exemple de motivation 1.1 décrit dans le chapitre 1. Ainsi la première étape consiste à extraire les blocs canoniques et à construire l'arbre R-PST comme introduit dans la section 3.3.2.1.

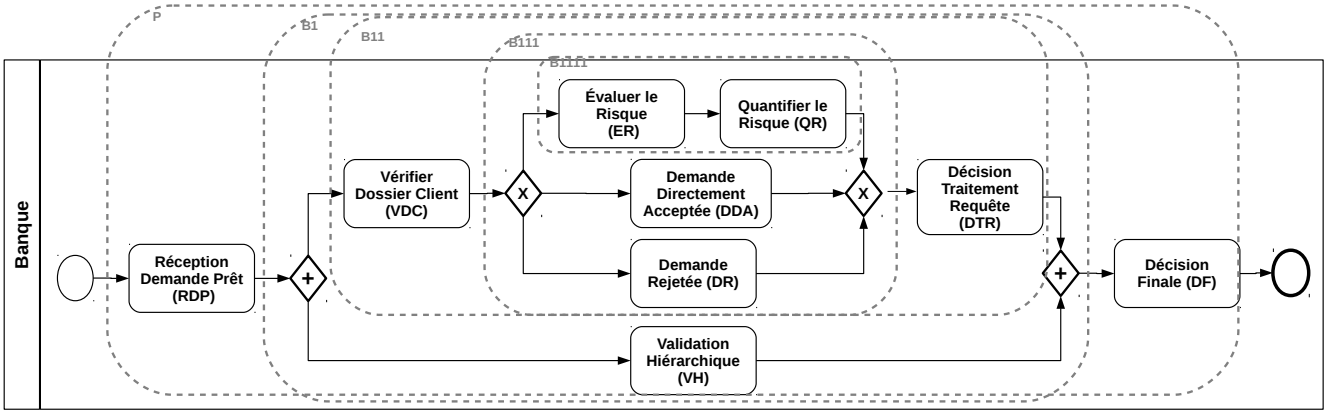


FIGURE 3.7 – découpage du processus de *prêt bancaire* en blocs

L'extraction des blocs canoniques de notre exemple de motivation retourne la figure 3.7. Comme illustré, les différents blocs du processus n'ont qu'une seule entrée et qu'une seule sortie respectant ainsi le principe SESE. Par ailleurs, en se basant sur la technique de construction d'arbre introduite dans la section 2.2.2.3, couplée à la technique de grammaires attribuées (introduite dans la section 3.3.2.1), nous obtenons l'arbre R-PST de la figure 3.8.

La figure 3.8-b, represente l'arbre R-PST après introduction des différents attributs pour caractériser les fragments sensibles. Ainsi, les criticités des tâches *VDC* et *DTR* sont

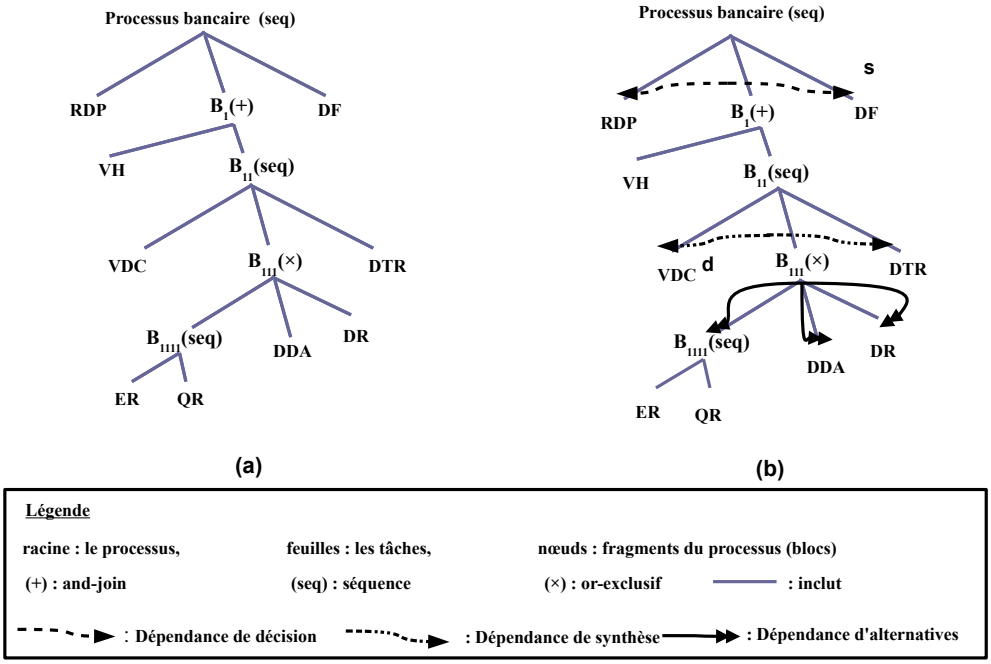


FIGURE 3.8 – (a) : Processus de prêt en R-PST, (b) : Processus de prêt R-PST décoré

respectivement « d » et « s », correspondant aux décisions et aux synthèses. Par ailleurs, les fragments *VDC-DTR* et *RDP-DF* représentent « les dépendances de décision » et « de synthèse » respectivement. Les fragments B_{1111}, DAA et DR quant à eux représentent les différentes alternatives.

3.3.3.1 Génération des contraintes et des niveaux de confiance

Après représentation de notre processus sous forme d'arbre R-PST attribué, la prochaine étape consiste à générer les contraintes et les niveaux de confiance à partir des règles d'assignation de la section 3.3.2.2.

Ainsi, nous obtenons sur notre exemple de motivation les contraintes suivantes :

- $trust\_level(DF) > l_1$. DF est une synthèse, elle doit par conséquent être déployée sur un cloud avec un important niveau de confiance l_1
- $trust\_level(VDC) > l_2$. VDC est une décision, elle doit aussi être déployée sur un cloud avec un bon niveau de confiance l_2
- $separate(RDP, DF)$. Séparation des tâches complémentaires RDP et DF sur différents clouds.
- $separate(VDC, DTR)$. Séparation des tâches complémentaires VDC et DTR sur différents clouds.
- $separate(B_{1111}, DDA)$. Séparation de la tâche DDA de son alternative représentée par le bloc B_{1111}.

- $separate(B_{1111}, DR)$. Séparation de la tâche DR de son alternative représentée par le bloc B_{1111}.
- $separate(DDA, DR)$. Séparation des tâches alternatives DDA et DR.

3.3.3.2 Cohérence des contraintes

Comme expliqué précédemment, certaines incohérences entres les règles générées par l'obfuscateur et celle énoncées par le designer peuvent se poser. Dans notre exemple de motivation, le designer peut décider de maintenir en local VDC et DTR générant ainsi la contrainte $colocate(VDC,DTR)$, qui est inconsistante avec $separate(VDC,DTR)$.

Dans ce cas là, $separate(VDC, DTR)$ sera supprimée car nous considérons que les contraintes générées par le designer priment sur celle de l'obfuscateur. Par ailleurs, il est à noter que le même cas peut s'appliquer à d'autres contraintes de séparation.

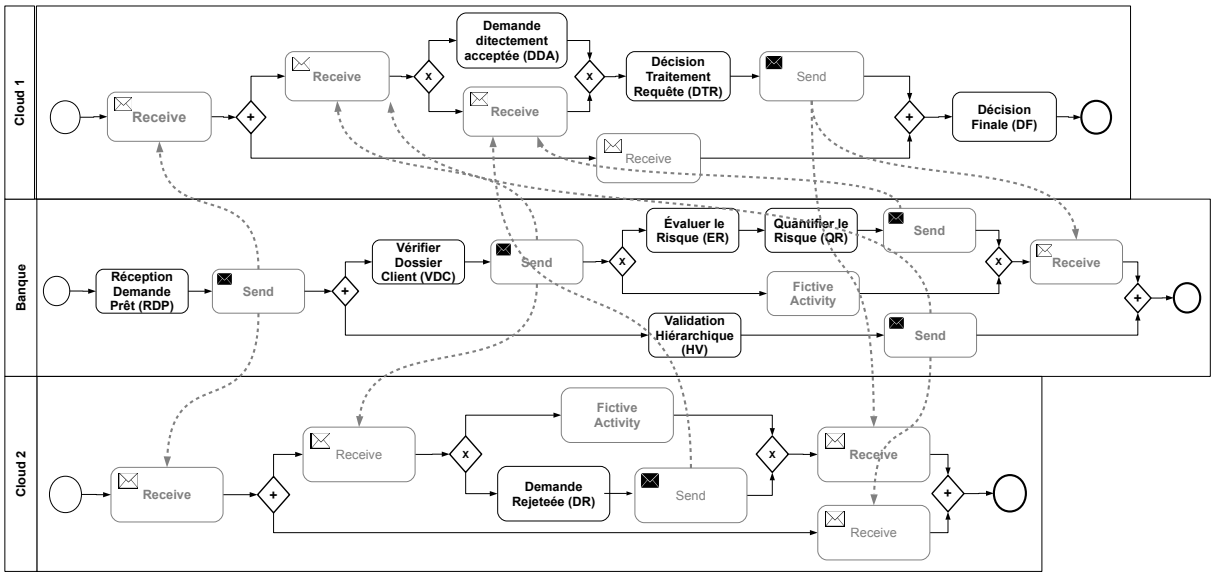


FIGURE 3.9 – Représentation du processus prêt bancaire sous la forme d'une collaboration de fragments de processus

Ainsi, après génération des contraintes et la vérification de leur consistances, plusieurs déploiements du processus sont possibles (figures.3.9 et 3.10). Ces déploiements répondent au mêmes contraintes mais diffèrent selon les configurations tâches/clouds e.g. la tâche VCD est déployée en local dans le partitionnement de la figure.3.9 tandis qu'elle est déployée sur le cloud 1 dans le second partitionnement. Cependant, nous pouvons remarquer que dans les deux cas, elle est toujours séparée de sa tâches complémentaire DTR.

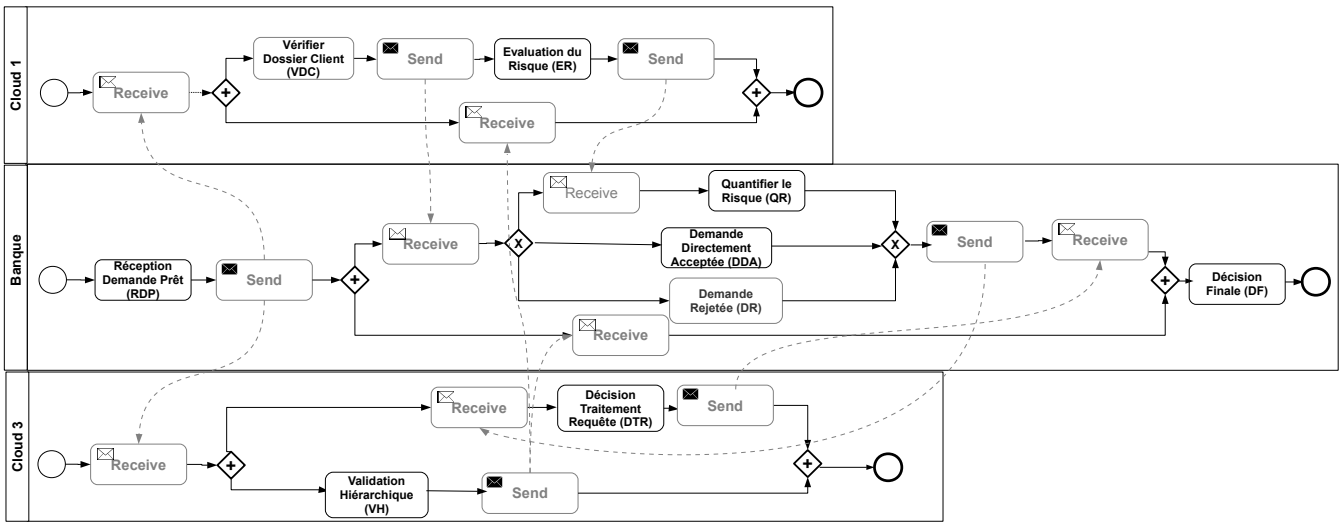


FIGURE 3.10 – Autre exemple de collaboration représentant le processus de prêt bancaire

3.4 Approche d'obfuscation basée sur l'introduction de fausses tâches

Comme introduit précédemment, l'objectif de cette thèse est d'aider les designers et les entreprises au déploiement sécurisé de leur processus ou une partie de leur processus dans le cloud. La méthode d'obfuscation introduite précédemment assure d'avoir au niveau de chaque cloud juste une partie du processus. Chaque partie n'étant pas suffisante pour reconstruire les informations sensibles sur la logique du processus.

Cependant, bien qu'efficace contre un seul cloud qui essayerait seul de découvrir la logique du processus, cette solution ne protège pas contre un ensemble de clouds, qui en combinant leurs informations pourraient deviner une partie de la logique de fonctionnement du BP (même si la décomposition du processus complique la conspiration entre clouds).

En effet, si les clouds qui déploient les fragments sensibles (i.e. *décisions, dépendance de décisions, synthèses, dépendance de synthèses et les fragments alternatifs*) arrivent à collaborer, violant ainsi nos contraintes de séparations introduites dans 3.3.2.3, ils seront en mesure de reconstruire le processus, en partie ou totalement. Comme illustré dans la figure.3.11, si les clouds 1 à 5 (qui déploient d'importantes informations) arrivent à collaborer, ils seront en mesure de reconstruire une partie de la logique du processus. Leur collaboration peut s'établir à travers l'ensemble des clouds qui les séparent. En effet, si C_1 arrive à convaincre C_2 de conspirer avec lui, qui lui même arrive à convaincre C_3 et ainsi de suite jusqu'à C_5, cet ensemble de clouds pourra reconstruire la logique du processus.

Dans cette optique, l'objectif de l'approche développée ici est de complexifier au maximum la tâche des clouds malicieux qui tenterait de communiquer avec les clouds qui déploient les tâches complémentaires aux leurs, dans le but de reconstruire la logique du processus (voir figure 3.11).

Pour ce faire, nous présentons dans cette section notre approche d'ajout de faux fragments à certains endroits stratégiques du processus, ayant pour objectif d'augmenter la distance qui sépare les tâches complémentaires. Nous présentons par la suite nos différentes règles d'assignation de ces faux fragments, et enfin appliquons cette méthode à notre exemple de motivation du chapitre 1.

Il est cependant à noter que l'approche développée ici, bien qu'efficace pour augmenter la difficulté d'une coalition et la retarder au maximum, ne peut l'empêcher de se produire.

3.4.1 Approche de complexification par ajout de fausses tâches

L'approche de complexification développée dans ce chapitre consiste à insérer à certains points stratégiques de la collaboration de faux fragments, avec l'objectif d'augmenter le nombre et la taille des chemins séparant deux tâches complémentaires (voir 2.2.3.1). Plus précisément, le but de cet ajout est d'augmenter l'écart qui sépare les clouds abritant ces tâches complémentaires, i.e. le nombre de clouds sur le chemin séparant ces tâches. En effet, nous considérons que plus grand est cet écart, moins il est probable qu'un cloud malicieux puisse rapidement construire le chemin menant à son cloud complémentaire.

Pour introduire ces faux fragments d'une manière optimale, il est primordial de savoir quand les considérer, où les insérer et comment les déployer dans un environnement multi-cloud.

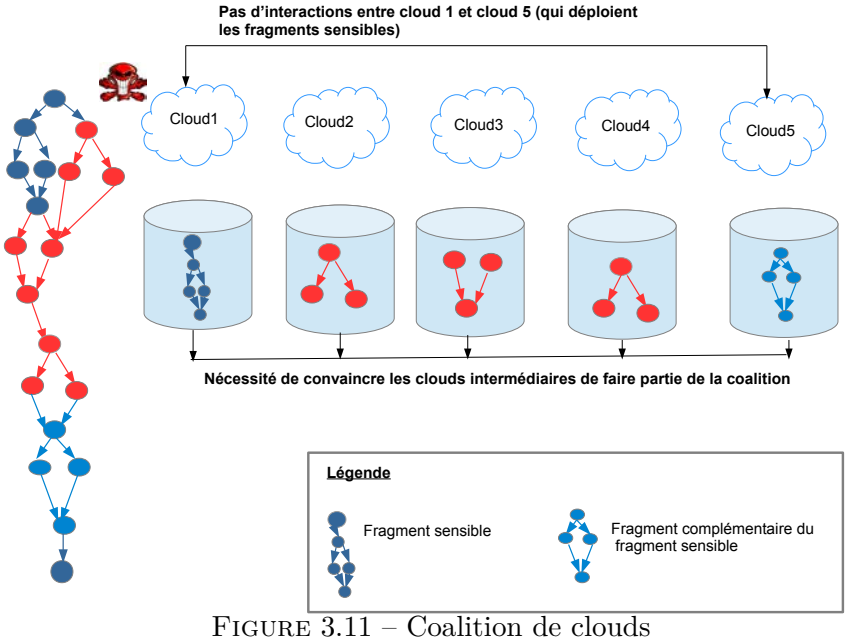


FIGURE 3.11 – Coalition de clouds

Avant d'introduire les étapes de notre approches de complexification, nous commençons par introduire les notions suivantes qui sont nécessaires à la compréhension de notre méthode.

- **Faux fragment** : Un faux fragment est une tâche ou en ensemble de tâches ne modifiant pas la logique de fonctionnement du processus et ne lui ajoutant aucune valeur.
- **Chemin** : Nous définissons un chemin p_j reliant deux tâches sensibles complémentaires i et i' comme l'ensemble des clouds exécutant des tâches sur le chemin logique j séparant i de i'. Dans la figure.3.11, le chemin qui relie C_1 à C_5 est $C_1 - C_2 - C_3 - C_4 - C_5$.

Il est à noter que notre nouvel algorithme d'obfuscation est une extension de l'algorithme introduit dans l'approche précédente (figure 3.5, section 3.3). En effet, cet algorithme est composé des mêmes étapes que celles introduites dans la section3.3.2, étendue par deux étapes supplémentaires :

- **Introduction de faux fragments** : Cette étape consiste à introduire de fausses tâches à certains endroits stratégiques, dans le but d'augmenter l'écart et le nombre de chemins existants entre les clouds qui déploient les fragments sensibles.
- **Règles de déploiement** : Cette étape consiste à introduire un ensemble de règles pour le déploiement du processus métier. Ces règles ont pour objectif d'assurer une concordance entre les nouvelles contraintes générées suite à l'introduction des faux fragments et les contraintes obtenus suite aux règles introduites dans la section.3.3.2.2.

Ainsi, en introduisant ces étapes, nous obtenons l'algorithme représenté dans la figure.3.12.

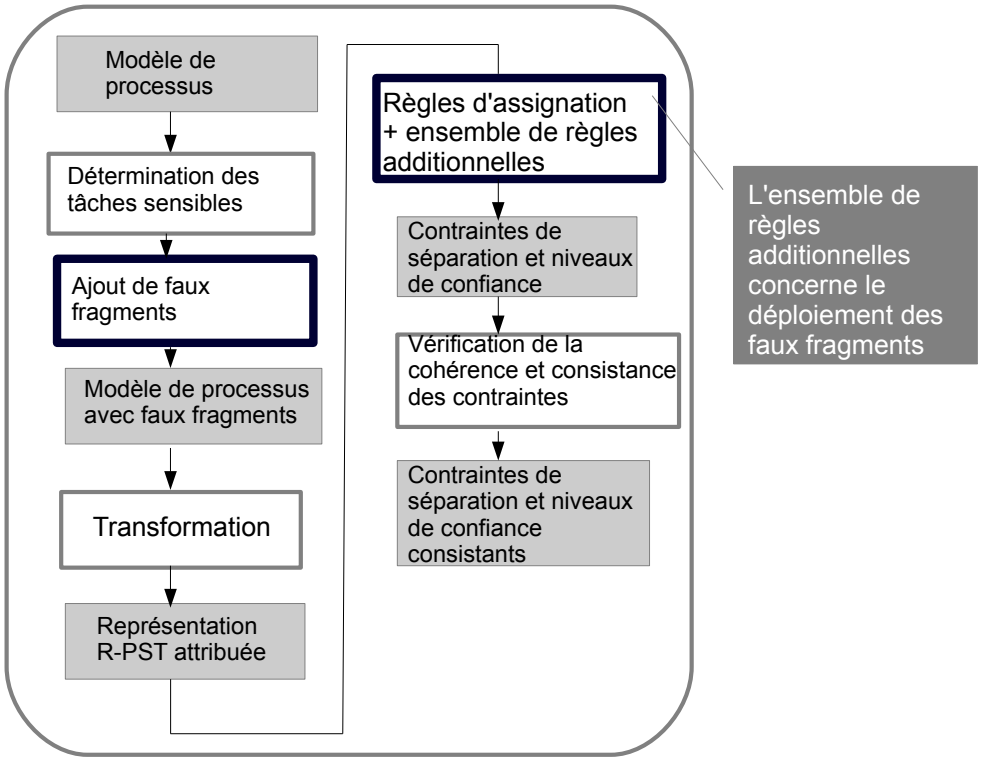


FIGURE 3.12 – Algorithme d'obfuscation avec ajout de faux fragments

Étant donné que les autres étapes de l'algorithme sont les mêmes que celles introduites précédemment, nous introduisons dans ce qui suit seulement les étapes supplémentaires.

3.4.1.1 Introduction de faux fragments

Comme introduit précédemment, afin d'assurer une complexification optimale, il est important de savoir à quel moment considérer ces faux fragments, i.e. est-il préférable de les ajouter au processus initial (avant décomposition), ou est-il plutôt préférable de les ajouter directement aux fragments obtenus après la première phase d'obfuscation. Notre choix s'est porté sur l'introduction des faux fragments dans le processus initial pour les raisons suivantes :

- Opérer au niveau logique permet aux designers de garder un certain contrôle.
- Puisque la construction d'une coalition est directement reliée à l'interaction des clouds qui déploient les fragments sensibles, elle peut être gérée à travers un ensemble de patrons de conception. En effet, nous introduirons dans ce qui suit, un ensemble de patrons et de règles de déploiement des faux fragments, permettant de réduire autant que possible la probabilité d'une coalition.

Par ailleurs, nous partons du principe que les fragments les plus sensibles sont ceux à protéger en priorité, en raison de l'importance des informations qu'ils comprennent. En effet, si un cloud qui déploie l'un de ces fragments sensibles s'avère être malicieux, il pourrait accéder à des informations importantes, portant ainsi préjudice au savoir-faire du processus. Par conséquent, il est primordial d'augmenter la difficulté de ces clouds malicieux à former une coalition en complexifiant leurs interactions.

Dans cet objectif, nous proposons dans ce qui suit un ensemble de patrons de conception permettant d'augmenter le nombre et la taille des chemins les séparant, incrémentant par conséquent l'écart entre eux.

- ***Introduction de faux fragments*** : Nous introduisons de faux fragments au début et à la fin de chaque chemin séparant une passerelle ouvrante *(X)OR-split* (respectivement *AND-Split*) et sa passerelle fermante *(X)-OR join* (respectivement *AND-join*) afin d'augmenter la protection des *décisions* (respectivement des *synthèses*)(voir les figures 3.13 et 3.14).
- ***Introduction de faux chemins*** Introduire une fausse alternative entre chaque *(X)OR-split* et son correspondant *(X)OR-join* (figure 3.13).

Il est à noter que l'introduction de ces faux fragments n'a pas d'effet sur la sémantique de la collaboration ; ils peuvent être introduits n'importe où. Cependant, l'avantage de ces patrons de conception réside dans le fait d'assurer un déploiement plus optimal. En effet, cela nous permet d'obtenir des chemins plus longs à moindre coût, i.e. nous introduisons moins de fragments en suivant ces patrons qu'en les introduisant de manière aléatoire. De plus, cela nous permet de réduire au mieux le risque de coalition, i.e. insérer les fragments à des endroits considérés *stratégiques*.

La figure 3.15 représente notre exemple de motivation de la figure 1.1 après insertion des faux fragments.

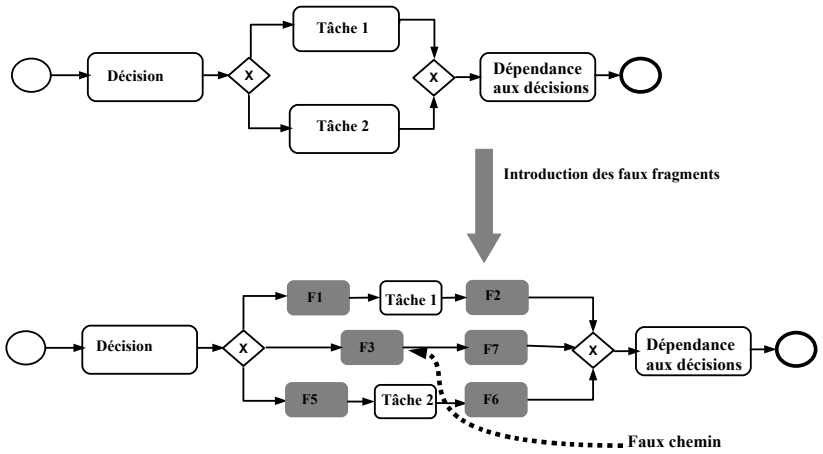


FIGURE 3.13 – Bloc décision après introduction des faux fragments

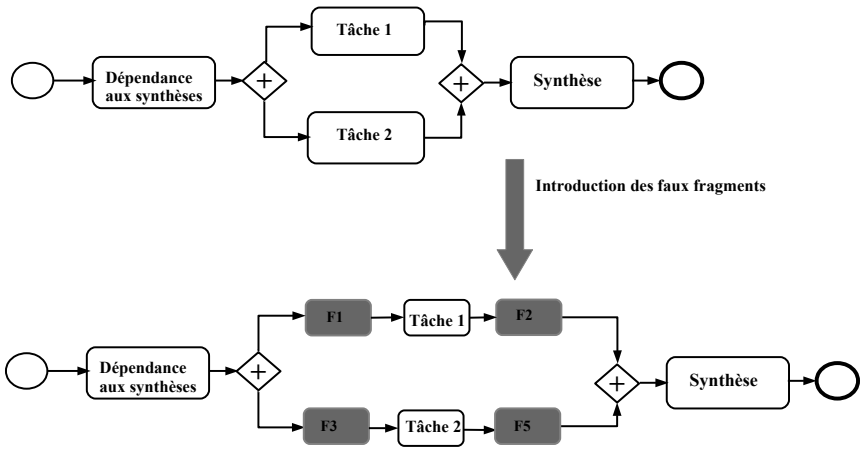


FIGURE 3.14 – Bloc synthèse après introduction des faux fragments

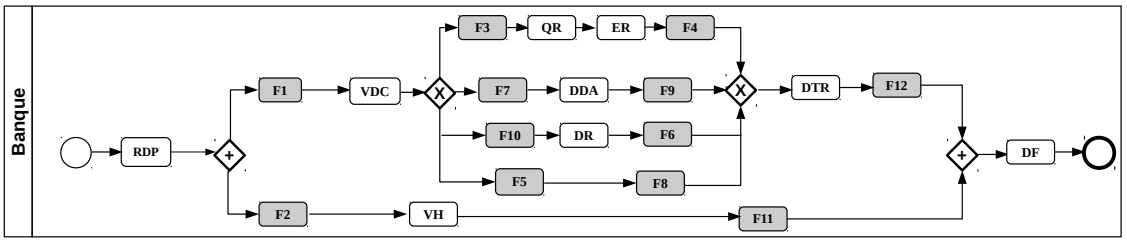


FIGURE 3.15 – Collaboration correspondante au processus de prêt bancaire avec l'introduction de faux fragments

3.4.1.2 Règles d'assignation des faux fragments

L'introduction des faux fragments comme expliqué précédemment permet d'allonger la taille et le nombre des chemins reliant les tâches sensibles. Par ailleurs, ils donnent également lieu à de nouvelles contraintes de séparation qui doivent être cohérentes avec nos

contraintes introduites dans la section 3.3.2.3. Également, comme introduit dans la section 1, le but de notre approche est d'augmenter le niveau d'obfuscation des collaborations en répondant à certains besoins fonctionnels/non fonctionnels (sécurité, coût, risque, performances....) [GANYG15] qui peuvent être en contradictions avec les nouvelles contraintes introduites par les faux fragments. Dans cette optique, nous proposons de :

- Dans un premier temps :
  - Séparer les faux fragments et les fragments sensibles sur différents clouds.
  - Déployer autant que possible les faux fragments sur différents clouds.
- Dans un second temps, en cas de contraintes fonctionnelles et non fonctionnelles (ex : un certain niveau de sécurité est exigé sans dépasser un coût donné), privilégier autant que possible :
  - L'assignation des faux fragments se situant sur un même chemin séparant deux fragments complémentaires à différents clouds, i.e. éviter d'avoir sur un même cloud les faux fragments qui relient deux tâches complémentaires.
  - L'assignation du fragment sensible et du faux fragment qui le précède ou qui le suit à deux clouds différents.

Ainsi, déployer ces faux fragments en tenant compte de ces contraintes assure de diminuer le risque de coalition à moindre coût.

3.4.2 Application à l'exemple de motivation

En appliquant notre méthode d'obfuscation couplée à la complexification par ajout de fausses tâches (section 3.4), nous obtenons la figure 3.16. En comparaison au déploiement des figures 3.9 et 3.10 qui ne se basent que sur les tâches sensibles pour le déploiement du processus, ce déploiement permet de cacher les interactions directes qui lient les fragments sensibles, en augmentant la taille des chemins qui les séparent.

Par exemple, les clouds qui déploient les tâches sensibles VDC et DTR sont reliés à travers une interaction directe dans les figures 3.9 et 3.10, tandis qu'il n'existe aucune interaction directe les reliant dans la figure 3.16, i.e. les clouds qui les déploient doivent conspirer avec ceux se trouvant sur le chemin les séparant, diminuant par conséquent la probabilité de fuite d'informations.

3.5 Validation

Cette section présente les aspects et les choix technologiques liés à implémentation de nos approches introduites dans ce chapitre.

Le but de ces implémentations est de tester d'une part la faisabilité de nos hypothèses de départ, i.e. les fragments sensibles sont localisés *avant* et *après* les passerelles *X(OR)-split, X(OR)-join, AND-split, AND-join* ainsi que d'importantes informations peuvent être exploitées à partir des différentes altératives.

D'autre part, ces implémentations nous permettent de vérifier l'efficacité des méthodes d'obfuscation, en mesurant les niveaux d'obfuscation obtenus lors du déploiement des processus métiers dans le cloud.

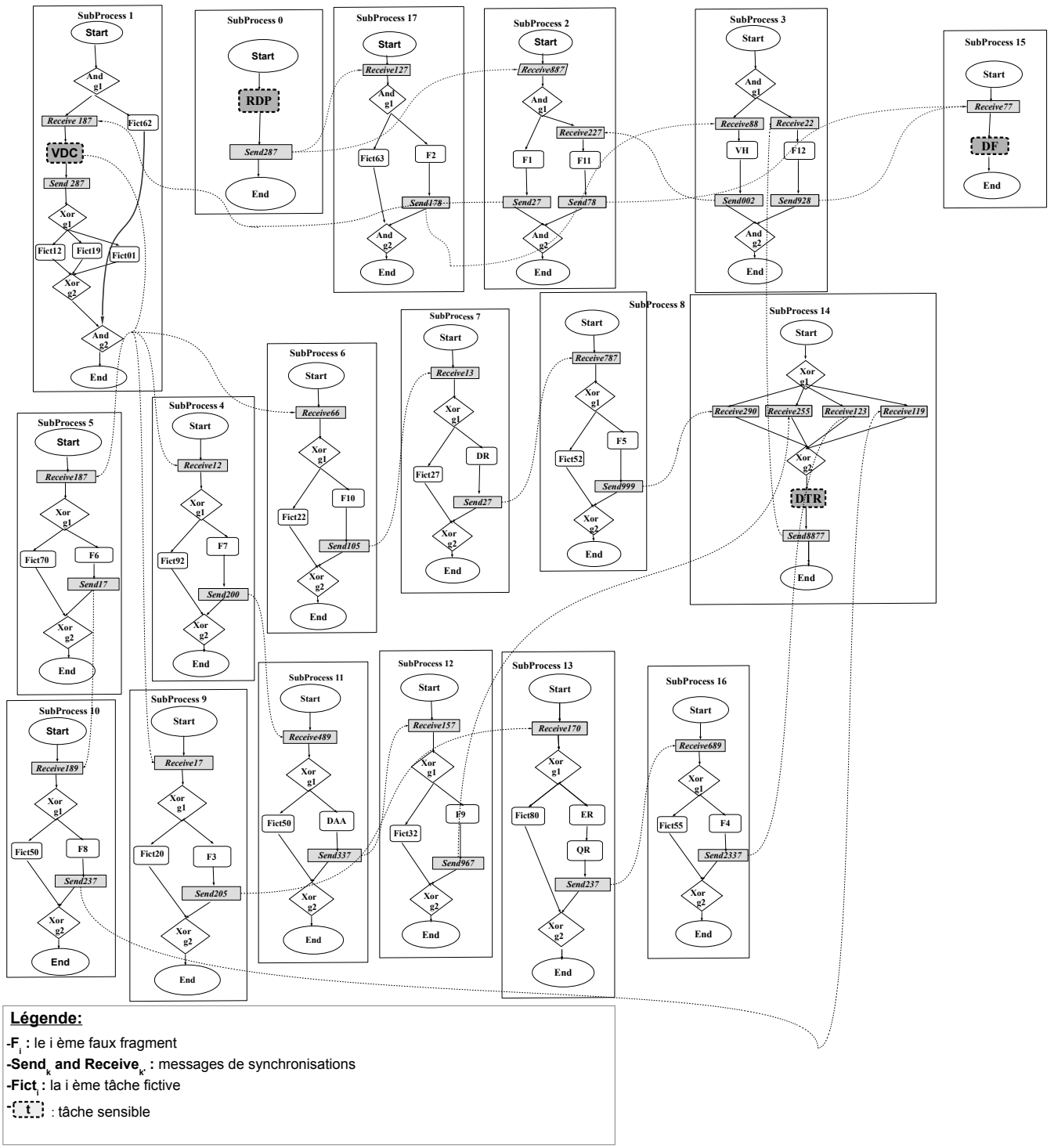


FIGURE 3.16 – Déploiement du processus de prêt bancaire avec faux fragments intégrés

Pour ce faire, nous considérons dans un premier temps un ensemble d'exemples de processus métiers pour vérifier nos hypothèses de départ. Nous introduisons et justifions par la suite une première métrique ayant pour objectif de calculer le niveau d'obfuscation obtenu après l'application de notre algorithme. Enfin, nous présentons une seconde métrique ayant pour but de mesurer l'effet de l'introduction des faux fragments sur la difficulté de

reconstruction du processus.

3.5.1 Positionnement des tâches sensibles

Afin de vérifier l'exactitude de nos hypothèses de départ, nous avons analysé différents processus métiers à partir de différentes sources dans la littérature [Bri13] [DLRM+13] [Fdh11] [LMS14] [Wes12]. Notre analyse a révélé qu'une large majorité des processus (environs 75%) vérifient nos hypothèses ; à savoir que la séparation des tâches sensibles permet de préserver la logique du processus. Pour les 25% restants, notre algorithme est considéré comme n'ajoutant pas de valeur particulière par rapport à un déploiement aléatoire.

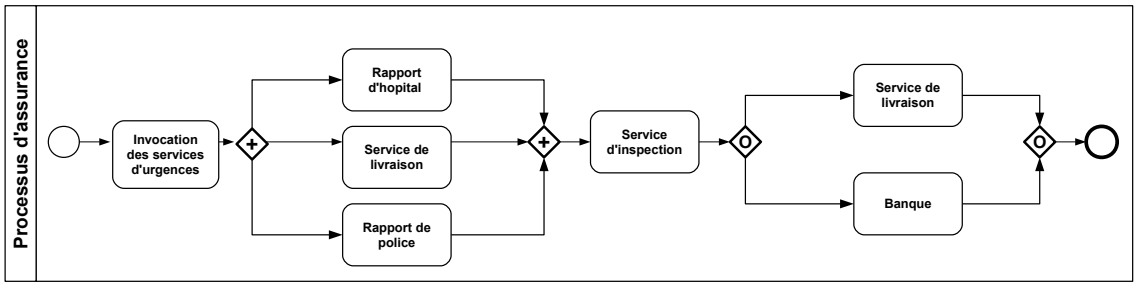


FIGURE 3.17 – Processus validant nos hypothèse

La figure 3.17 représente un exemple de processus où les tâches sensibles permettent à un attaquant tierce de deviner la logique du processus à partir de leur contenu. Ce processus représente un procédé exécuté par une compagnie d'assurance en vue de déterminer si la réclamation de sinistre d'un souscripteur est remboursable [Fdh11]. Ce processus débute en invoquant les services d'urgences (SU) afin d'obtenir plus de détails au sujet de l'incident. Après obtention des détails nécessaires, la compagnie d'assurance invoque l'hôpital, la police ainsi que le service de livraison (qui récupère les documents de réclamation de chez le client). Ainsi, après obtention des rapports de police et d'hôpital, la compagnie d'assurance invoque le service d'inspection, qui décidera si la demande du client sera accordée ou pas. Dans le cas d'une réponse positive, la banque et le service de livraison seront invoqués. Dans le cas contraire, seul le service de livraison sera invoqué pour informer le client de la décision finale.

Les tâches *invocation des services d'urgences* et *service d'inspection* sont considérées comme des tâches sensibles vu l'importance des informations qu'elles révèlent. Par ailleurs, en plus d'être sensibles, ces tâches sont considérées comme étant complémentaires (i.e. en cas de déploiement sur le même cloud, permettent de reconstruire une partie ou la totalité de la logique du processus). Par conséquent, appliquer notre algorithme à ce processus permet de protéger le savoir-faire et la stratégie de prise de décisions de la compagnie d'assurance.

Il est cependant à noter que dans certains cas, les fragments qui précédent/suivent les passerelles $X(OR)/AND$ ne sont pas considérés comme étant *sensibles* car ils ne comprennent pas d'informations vraiment pertinentes, rendant ainsi leur séparation inutile quant à la protection de la logique du processus. Cette séparation n'aura aucun effet sur le fonctionnement du processus.

La figure 3.18 représente l'un de ces cas [DLRM+13]. Le processus décrit une compagnie qui stocke ses produits dans différents entrepôts : Amsterdam et Hambourg. Ainsi, à la

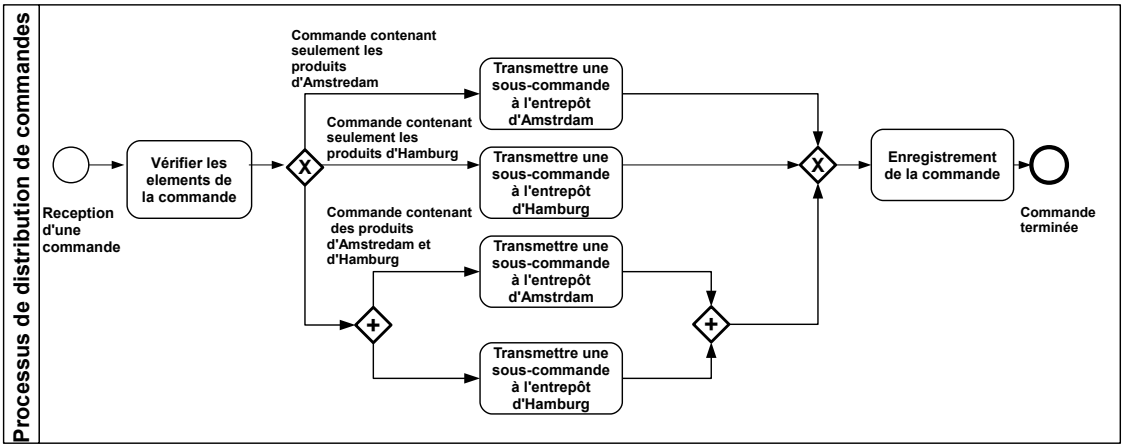


FIGURE 3.18 – Processus ne validant pas nos hypothèses

réception d'une commande, cette dernière est transférée aux entrepôts : si certains produits sont disponibles au niveau de l'un des entrepôts, une sous-commande est alors envoyée.

Par conséquent, à travers cette analyse, nous concluons que nos hypothèses et la plausibilité de notre approche sont globalement confirmées.

3.5.2 Métriques pour le calcul des niveaux d'obfuscation

Les résultats obtenus dans la section précédente bien qu'encourageants, ne peuvent être considérés complets. En effet, étant dans l'incapacité d'analyser un grand nombre de processus ou de comparer les collaborations générées par notre méthode avec un grand spectre de collaborations, nous avons comparé les niveaux d'obfuscation de collaborations issues du même modèle de processus, mais dans des configurations différentes, générées par nos algorithmes et générées de façon aléatoire.

Pour ce faire, nous avons introduit un ensemble de métriques, qui furent utilisées pour évaluer les niveaux d'obfuscation obtenus à travers les différentes approches proposées.

La première métrique a pour but de mesurer le niveau de complexité de la collaboration à travers les différents messages (entrants et sortants) échangés entre les différents clouds qui la composent. Cependant, puisque cette complexité est indépendante de la position des fragments sensibles (ne se base que sur leur séparation pour le calcul de la complexité, en ignorant la distance qui les sépare), nous introduisons une seconde métrique ayant pour objectif de calculer l'effort devant être entrepris par un cloud malicieux afin de reconstruire la logique du processus. cet effort est principalement basé sur la distance qui sépare deux fragments complémentaires, i.e. plus ils sont séparés et plus la reconstruction est difficile.

3.5.2.1 Complexité d'un modèle de processus

Étant donné que nous n'avons trouvé aucune métrique dans la littérature qui convienne directement à notre contexte, nous avons défini notre propre métrique, basée sur le peu de mesures de complexité existantes pour les modèles de processus, considérant que plus la logique d'un processus est complexe, plus il est obfusqué.

i) Facteurs d'obfuscation Étant donné que le rôle de l'obfuscation est de complexifier la structure et la compréhension d'un processus, nous avons considéré dans notre métrique les facteurs suivants : nombre de fragments, nombre de messages entrants des fragments et nombre de messages sortants des fragments.

- **Nombre de fragments** : Le nombre de fragments d'une collaboration est défini comme étant le nombre de fragments qu'elle comprend.

  Dans notre approche, un fragment est assigné à un fournisseur cloud, augmentant ainsi le nombre de clouds de la collaboration dans le cas d'un processus avec un nombre important de fragments. Ce facteur est motivé par le fait que dans un environnement multi-cloud, plus il y a de fragments appartenant à la collaboration, et plus il sera difficile pour un fournisseur cloud d'organiser l'ingénierie inverse de la collaboration (comme introduit dans la section 3.4.1).

  Par ailleurs, étant dans un environnement multi-cloud, le nombre de fournisseurs a un impact direct sur le nombre de connexions potentielles entre eux. En effet, dans le cas où n fournisseurs appartiennent à la collaboration, ils seront dans la possibilité de générer $n \times (n-1)/2$ connexions, augmentant ainsi la complexité du processus. Par conséquent, nous considérons que plus nous avons de fournisseurs dans la collaboration ; plus il est probable que le nombre de connexions soit important, complexifiant ainsi la structure du processus et la possibilité de coalitions.

  Il est cependant à noter que ce choix est également basé sur l'état de l'art de la complexité du logiciel et du code en général, et des processus métiers en particuliers. En effet, le nombre de fragments peut être comparé à la longueur d'un processus ou d'une activité complexe. La métrique *Interface Complexity Metric* de Cardoso et al.[CMNR06] considère cette valeur comme étant le nombre d'activités d'un processus, tandis que la métrique *the Information Flow Metric* ([HK81]) suggère que cette valeur soit corrélée à la taille du module en question e.g. à travers le nombre de lignes de code.

- **Nombre de messages entrants d'un fragment (mess_in)** :

  Nous considérons que le nombre de messages en entrées d'un fragment affecte le nombre de combinaisons possibles avec les autres fragments. En effet, plus il a de messages en entrées, plus il est probable que le fragment soit connecté à plusieurs autres fragments, et plus il sera difficile de retrouver son rôle dans la collaboration. De plus, il est à noter que ces éventuelles connexions augmentent le nombre de chemins de la collaboration.

  Ce choix est également corroboré par l'état de l'art. Plus particulièrement, notre métrique est inspirée de celles de Kafura.[HK81] et Cardoso.al [CMNR06], où la notion de *fan_in* (nombre de flux entrants dans un module) correspond à notre notion de nombre de message entrants (mess_in) d'un fragment. Par ailleurs, notre définition est aussi inspirée de la complexité cyclique de McCabe [McC76], qui stipule que le nombre de chemins augmente en concordance avec le nombre de messages. Enfin, cette notion est également inspirée de la notion de *couplage* [AKC01] qui mesure le taux de connexion d'une activité avec les autres activités du process, i.e. plus elles sont connectées et plus le processus est complexe.

- **Nombre de messages sortants d'un fragment (mess_out)** :

Comme pour les messages entrants, nous considérons que plus un fragment a de messages sortants, plus il est probable qu'il soit connecté à plusieurs autres fragments, et plus il sera difficile de retrouver son rôle dans la collaboration.

Les métriques *Information Flow* et *Interface Complexity* gèrent également une notion de *fan_out* (nombre de flux sortants d'un module), correspondant à notre notion de nombre de messages sortants (mess_out) d'un fragment. Il est à noter que cette notion de mess_out est également liée à la notion de *couplage* des modules logiciels qui mesure le taux de connexion d'une activité.

On remarquera cependant qu'augmenter le nombre de fragments *NoF* peut être perçu comme contradictoire avec l'augmentation du nombre de messages entrants et sortants. En effet, l'augmentation du nombre de fragments permet une meilleure répartition des messages, réduisant ainsi les nombres de *mess_in* et *mess_out* de certains fragments. Cependant, dans un contexte purement analytique, nous considérons le processus tel quel, sans considération d'une meilleure conception. Par conséquent, les facteurs introduits et discutés ci-dessus permettent et contribuent à une bonne évaluation de la complexité de la collaboration, en se basant sur notre propre analyse corroborée par l'état de l'art.

ii) Métrique d'obfuscation d'une collaboration multi-cloud En se basant sur les facteurs d'obfuscations introduit ci-dessus, nous proposons la métrique suivantes qui permet de calculer le niveau de complexité d'une collaboration en tenant compte de l'environnement multi-cloud.

- **Complexité d'un fragment** Nous nous sommes basés sur la métrique *the Interface Complexity metric* [CMNR06] introduite dans la section 2.2.3.4 pour le calcul de la complexité d'un fragment i (CoF_i), en comparant un fragment de processus à une activité. Étant donné qu'un fragment est considéré comme une activité boite noire, sa longueur est égale à 1 . Les facteurs *mess_in* et *mess_out* quant à eux correspondent respectivement aux *nombre de messages entrants* et *nombre de messages sortants* (voir [CMNR06]).

$$CoF_i = (mess\_in_i \times mess\_out_i)^2 \tag{3.1}$$

  Les termes $mess\_in_i \times mess\_out_i$ représentent le nombre de combinaisons possibles d'une source vers une destination. Leur pondération est basée sur la conviction que la complexité est plus que linéaire en termes de connections qu'un fragment peut avoir avec son environnement (voir [HK81] pour plus de détails).

- **Complexité d'une collaboration (COM)**

  Étant donné que l'attaquant n'a a priori aucune connaissance de la logique du processus, l'idée est de considérer la collaboration comme un ensemble de fragments, i.e. des activités combinées aléatoirement. Par conséquent, la complexité de la collaboration peut être définie dans un premier temps comme étant le produit des complexités de tous ses fragments. Ainsi, en combinant les facteurs introduits précédemment, nous avons élaboré la formule suivante, qui mesure le niveau d'obfuscation d'une collaboration

$$\begin{aligned} COM &= \prod_i CoF_i \\ &= \prod_i (mess\_in_i \times mess\_out_i)^2 \end{aligned}$$

où i est le i^{ème} fragment de la collaboration.

- **Complexité d'une collaboration multi-cloud (CCOM)**

  Pour introduire et donner plus de poids au contexte multi-cloud, la complexité de la collaboration est multipliée par le nombre de fournisseurs (approximativement le nombre de fragments). En effet, nous nous sommes basés sur les travaux de [BE81], qui considèrent que comprendre et manipuler des éléments interconnectés est plus difficile si leur nombre est grand. Par conséquent, nous considérons que plus il y a de fragments dans la collaboration, plus elle est complexe [NGY$^+$18].

$$\begin{aligned} CCOM &= NoF \times COM \\ &= NoF \times \prod_i CoF_i \\ &= NoF \times \prod_i (mess\_in_i \times mess\_out_i)^2 \end{aligned}$$

3.5.2.2 Complexité d'une coalition

Cette métrique permet de mesurer à quel point il est complexe pour les fournisseurs de former une coalition après l'introduction des faux fragments (comme introduit dans la section 3.4).

Évidemment, une question qui peut se poser à ce stade est pourquoi deux métriques et non pas seulement une ? Et quelle est la position de chaque métrique par rapport à l'autre ?

Il est clair qu'ajouter de faux fragments augmente la complexité d'un modèle de processus étant donné que cela augmente les facteurs NoC(nombre de fragments), $mess\_in$ (nombre de messages entrants) et $mess\_out$ (nombre de messages sortants).

Cependant, étant donné que la métrique précédente ne prend pas en compte la position des fragments sensibles (ne se base que sur leur séparation pour le calcul de la complexité en ignorant la distance qui les sépare), ni le fait d'augmenter la complexité sur un chemin bien précis, i.e. entre les fragments sensibles, nous avons introduit une métrique calculant la complexité d'une éventuelle collusion entre fournisseurs cloud, i.e. l'effort devant être entrepris par un cloud malicieux afin de reconstruire la logique du processus. Cette métrique nommée *Collusion complexity metric* se base principalement sur la longueur des chemins séparant les tâches sensibles, et considère cette longueur comme une quantité de travail à effectuer par les fournisseurs cloud afin de réussir à conspirer. Elle peut être considérée comme une collection de trois métriques : $AverageLength_{Coalition}$, $LongestLength_{Coalition}$ et $ShortestLength_{Coalition}$

- $\boldsymbol{AverageLength_{Coalition}}$ est une métrique qui mesure l'effort moyen qu'un cloud malicieux doive fournir afin d'établir une relation entre deux clouds qui déploient des fragments sensibles du processus.

$$AverageLength_{Coalition} = AVG(LPC_i)_{\forall i \in P} \tag{3.2}$$

  Avec :

  – P : l'ensemble de tous les chemins reliant les fragments sensibles (voir section 3.4.1).

- AVG : la fonction average qui retourne la taille moyenne d'une coalition.
- LPC_i : le nombre de cloud sur le chemin i de l'ensemble P.

- ***LongestLength$_{Coalition}$*** est une métrique pour le calcul de l'effort le plus important qu'un cloud malicieux doive fournir afin d'établir une relation entre deux clouds qui déploient des fragments sensibles du processus.

$$LongestLength_{Coalition} = Max(LPC_i)_{\forall i \in P} \tag{3.3}$$

où la fonction Max retourne le maximum de LPC_i, i.e. le chemin le plus long entre les fragments sensibles.

- ***ShortestLength$_{Coalition}$*** Cette métrique calcule l'effort minimum qu'un cloud malicieux doive fournir afin d'établir une relation entre deux clouds qui déploient des fragments sensibles du processus.

$$Shortest_{Coalition} = Min(LPC_i)_{\forall i \in P} \tag{3.4}$$

où la fonction Min retourne le minimum de LPC_i, i.e. le chemin le plus court entre les fragments sensibles.

Ces trois métriques sont considérées comme étant complémentaires car elles permettent de situer l'effort maximum, moyen et minimum qu'un cloud malicieux doive fournir afin de créer une coalition de cloud.

Par ailleurs, les métriques *Complexité d'une collaboration multi-cloud (CCOM)* et *Complexité d'une coalition* sont également considérées comme *complémentaires*, i.e. la première permet de calculer la complexité globale de la collaboration à partir des messages entrants et sortants sans tenir compte de la possibilité de coalition. La seconde quant à elle calcule la complexité en tenant compte de la taille des chemins, et donc prend en compte la possibilité de coalition lors du calcul du niveau de complexité.

3.6 Expérimentations

Nous présentons au cours de cette section les différentes expérimentations réalisées en vue de tester l'efficacité de notre approche. Pour ce faire, nous introduisons en premier lieu l'environnement de développement des différentes techniques proposées. Nous présentons et discutons par la suite les résultats de notre première expérimentation, qui vise à calculer le niveau d'obfuscation de collaborations générées de différentes manières (en se basant sur notre approche, génération aléatoire...etc). Cette évaluation s'appuie sur notre métrique définie dans la section 3.5.2.1. Enfin, nous introduisons la dernière expérimentation dont l'objectif est de vérifier l'efficacité de l'introduction des faux fragments sur la complexité d'une éventuelle coalition entre fournisseurs cloud, en appliquant notre métrique introduite dans la section 3.5.2.2.

3.6.1 Environnement de développement

Pour le développement des différentes techniques que nous avons proposées, nous avons opté pour le langage de programmation Java. Ceci est motivé par sa portabilité et par le fait que plusieurs outils que nous avons utilisés pour la mise en œuvre sont déjà implémentés

en Java. Pour tester nos approches, nous avons utilisé des processus métiers sous format BPMN.

Afin de visualiser l'ensemble des collaborations obtenues suite à nos expérimentations, nous avons utilisé l'outil *graphiz*[20] couplé à notre environnement java. Plus particulièrement, cet outil parse les fichiers contenant la description de nos collaborations (obtenues après exécution de nos algorithmes) afin de les afficher sous forme d'images.

3.6.2 Mesure du niveau d'obfuscation d'une collaboration

Le premier ensemble d'expériences étudie l'effet de notre algorithme sur le niveau d'obfuscation des processus ainsi que sur le nombre de fragments obtenus. Ce niveau d'obfuscation est calculé sur la base de la métrique décrite dans la section 3.5.2.1. Ainsi, pour analyser si notre algorithme augmente le niveau d'obfuscation, nous choisissons de comparer les collaborations générées de trois manières différentes :

- A travers une distribution aléatoire simple.
- En appliquant les règles définies dans notre algorithme avec un positionnement aléatoire des fragments, i.e. déployer les fragments aléatoirement sur des clouds en s'assurant de respecter les contraintes de sécurité générées par notre algorithme.
- En appliquant les règles définies dans notre algorithme avec un positionnement itératif des fragments, i.e. on déploie sur le même cloud tant que nos règles le permettent.

Le déploiement d'un processus sur différents clouds peut générer un coût important, notamment dû aux différents transferts de messages et de données entre les différents clouds de la collaboration. Ainsi, pour pallier ce problème, le troisième mode de distribution peut être considéré comme étant un compromis entre coût et sécurité.

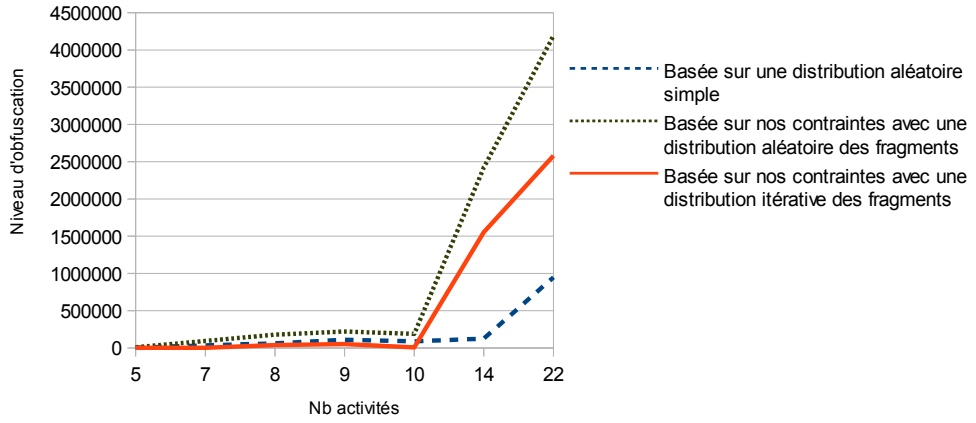
Il est par ailleurs à noter que les différentes expérimentations menées sont des simulations. En effet, effectuer des expérimentations avec des paramètres de clouds réels est difficile en raison du manque de standardisation du modèle BPMaaS [21]. Ainsi, afin de vérifier l'exactitude de nos hypothèses, nous avons généré jusqu'à 1000 instances pour chaque processus métier (nous avons utilisé différents processus de différentes tailles, i.e. nombre d'activité) pour les trois modes de distributions décrits précédemment.

Il est également important de retenir que seul peu de travaux abordent notre sujet, et généralement dans un contexte différent du nôtre. Par exemple, [XWG11] et [KNRS13] ont principalement pour objectif de protéger les requêtes émises, afin d'empêcher autant que possible un éventuel attaquant d'en retirer d'importantes informations. Par conséquent, en raison du manque de travaux déjà effectués dans ce domaine, nous comparons notre travail à une distribution aléatoire.

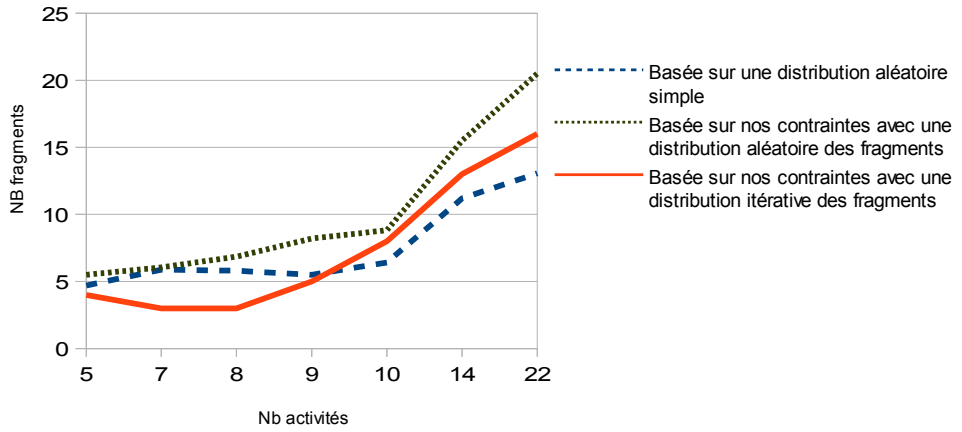
3.6.2.1 Résultats

L'expérience effectuée illustre le niveau d'obfuscation moyen, ainsi que le nombre de fragments obtenus suite aux trois modes de distribution introduits ci-dessus en faisant

20. http ://www.graphviz.org/
21. Business Process Management as a Service



(a) Niveaux d'obfuscation



(b) Nombre de fragments

FIGURE 3.19 – Résultats obtenus avec $NoF = n$

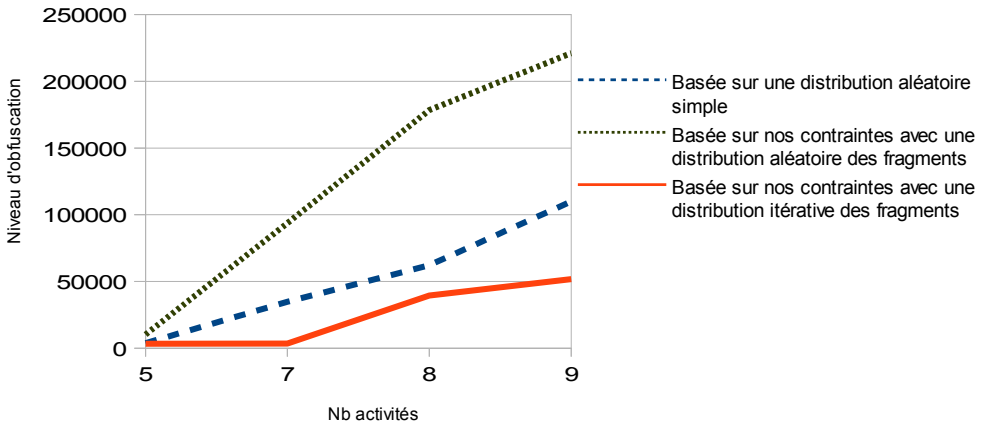
varier le nombre d'activités des processus. Les résultats obtenus sont illustrés dans la figure 3.19.

Comme illustré (figure 3.19a), le niveau d'obfuscation obtenu en appliquant les règles définies par notre algorithme avec un positionnement aléatoire des fragments non contraints par notre algorithme, est toujours meilleur que le niveau d'obfuscation obtenu par une distribution aléatoire simple. Il est aussi à noter que la différence entre eux augmente exponentiellement à partir de processus de 10 activités. Pour une meilleure représentation et pour plus de clarté, nous zoomons sur la figure 3.19a comme décrit sur la figure 3.20.

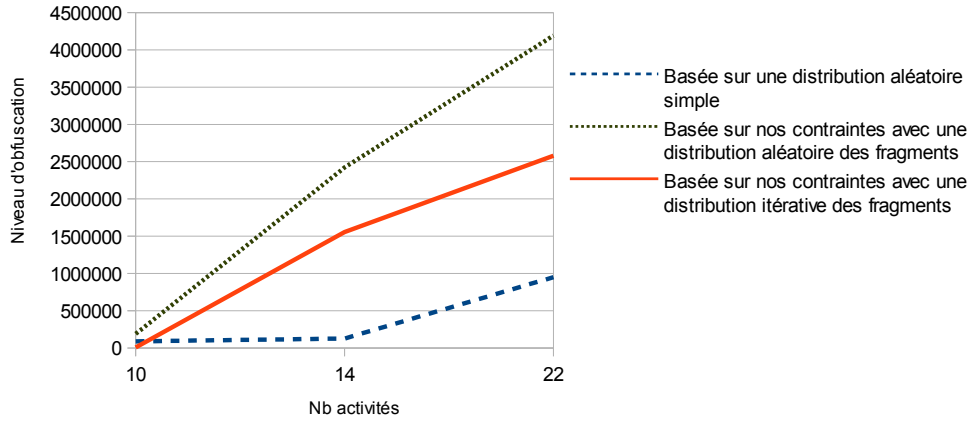
Ainsi, comme illustré dans la figure 3.20a, notre approche combinée à un positionnement itératif devient intéressante pour des processus de plus de 10 activités. En effet, pour de plus petits processus, il est plus intéressant d'utiliser le déploiement aléatoire simple.

Ceci est dû au fait que dans certains cas, les petits processus ne contiennent pas forcement un nombre élevé de tâches complexes. Par conséquent, appliquer nos règles combinées à un déploiement itératif engendrera peu de fragments (à cause du peu de règles de séparations appliquées), minimisant donc le nombre de messages échangés d'une part, et maximisant le nombre d'informations dans chaque fragment d'autre part.

De plus, et comme illustré dans la figure 3.19b, notre algorithme augmente le nombre de fragments associés à la collaboration (seulement à partir de processus de 10 activités

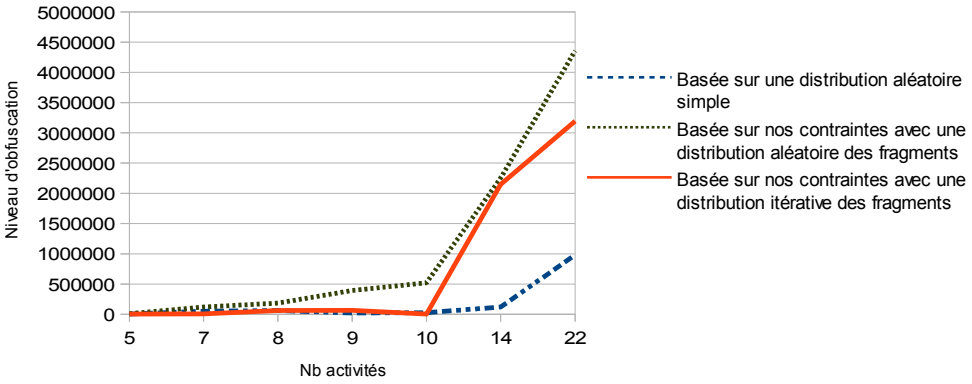


(a) Niveaux d'obfuscation avec nombre d'activité inférieur à 10

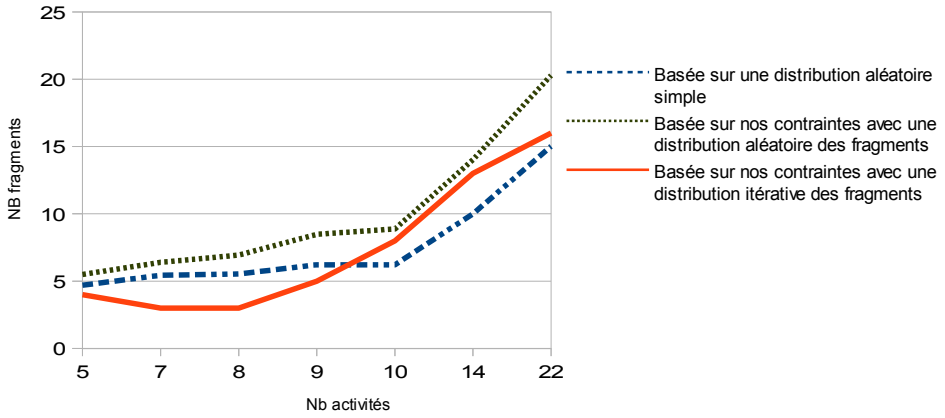


(b) Niveaux d'obfuscation avec nombre d'activité supérieur ou égal à 10

FIGURE 3.20 – Les niveaux d'obfuscation obtenus avec $NoF = n$

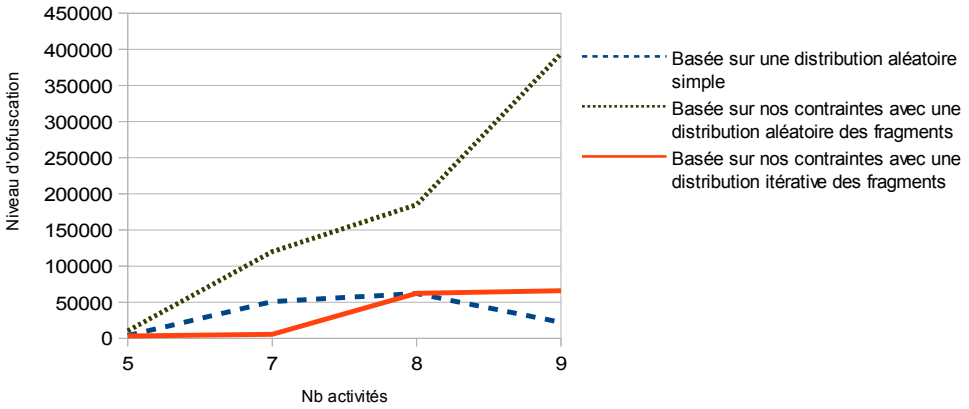


(a) Niveaux d'obfuscation

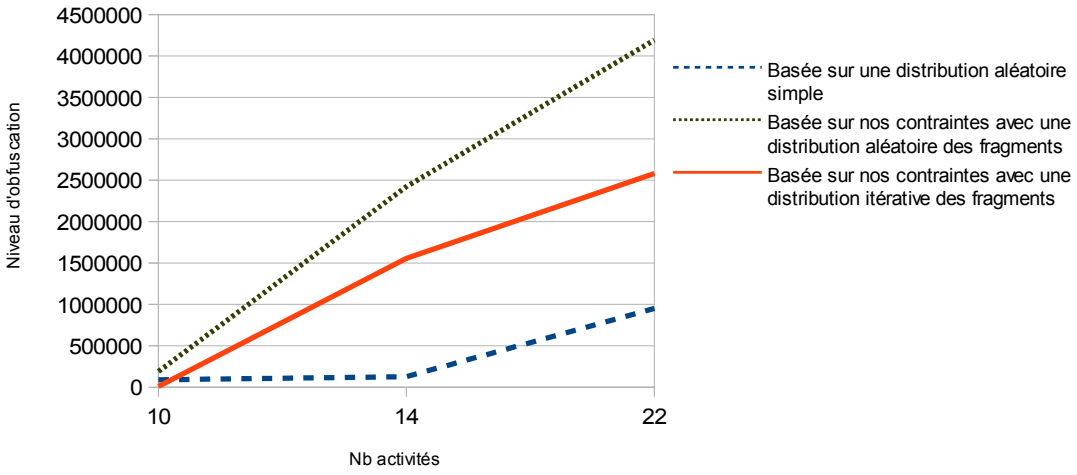


(b) Nombre de fragments

FIGURE 3.21 – Résultats obtenus avec NoF = n*(n-1)/2



(a) Niveaux d'obfuscation avec nombre d'activité inférieur à 10



(b) Niveaux d'obfuscation avec nombre d'activité supérieur ou égal à 10

FIGURE 3.22 – Les niveaux d'obfuscation obtenus avec NoF = n*(n-1)/2

lorsque nos règles sont associées à un déploiement itératif), augmentant ainsi le nombre de connexions entre eux, et par conséquent le nombre de chemins possibles. D'autre part, notre algorithme permet d'augmenter le nombre de clouds qui doivent conspirer afin de retrouver la logique de fonctionnement du processus (i.e. inverser l'ingénierie de la collaboration).

Cet ensemble de résultats fut obtenu en prenant le nombre de fournisseurs pour le facteur NoF. Comme expliqué précédemment, pour donner plus de poids à l'environnement multi-cloud, nous considérons cette valeur comme seuil minimum. Nous avons par ailleurs effectué un second ensemble expérimentations en tenant compte du nombre de connexions probables entre ces fournisseurs, en utilisant la valeur $n(n-1)/2$ pour le facteur NoF comme introduit dans la section i). En effet, nous considérons que plus il y a de connexions entre les fournisseurs et plus le processus est complexe. Les résultats obtenus sont illustrés dans la figure 3.21a.

Comme précisé et illustré dans les figures 3.22a et 3.22b, les résultats obtenus se rapprochent fortement de ceux obtenus suite à la première expérimentation, en offrant cependant un meilleur niveau d'obfuscation (principalement pour les processus avec un nombre important d'activités e.g. 14 ou 22 activités). En effet, donner plus de poids à l'environnement multi-cloud donne plus de poids au nombre de potentielles connexions de la collaboration. Ces connexions compliquent considérablement la structure du processus, permettant ainsi d'obtenir un meilleur niveau d'obfuscation. Il est à noter que certains changements sont également dûs au positionnement aléatoire de certaines tâches, qui génère un nombre différent de fragments dans certains cas.

3.6.2.2 Discussion

La conclusion des expériences est assez positive car la complexité des collaborations générées à travers notre approche (en appliquant nos règles combinées à un déploiement aléatoire ou itératif) est largement meilleure (supérieure) en moyenne, à la complexité moyenne des collaborations générées de façon aléatoire simple. Il est à noter que le résultat obtenu est une moyenne et certaines collaborations générées aléatoirement peuvent être meilleures dans certains cas. De plus, nos collaborations sont logiquement générées, i.e. basées sur la logique et la structure du processus. Par conséquent, leur complexité peut être plus facilement gérée par le client que celles générées de façon complètement aléatoire.

Cet argument ne s'applique pas aux collaborations générées par les concepteurs. Cependant, l'avantage de notre approche (section 3.3) réside dans le fait qu'elle puisse s'appliquer sur de grands processus et puisse intégrer d'autres dimensions (Qos, gestion du risque ...) pour un déploiement optimal, très difficile voire impossible à gérer manuellement pour de grands processus.

En se basant sur nos expérimentations, nous en concluons que le second mode de distribution (application de nos règles combinées à un déploiement aléatoire) est efficace avec tous les processus quelque soit leur taille (i.e. grands ou petits processus). Cependant, le troisième mode de distribution (application de nos règles combinées à un déploiement itératif) n'est intéressant que pour les processus ayant une taille supérieure ou égale à 10 activités. Peut-être qu'avec des processus avec moins d'environs 20 activités, les concepteurs peuvent mieux résoudre le problème. Cependant, les processus réels ont en général des dizaines d'activités, voire des centaines, et selon certains auteurs des milliers dans le contexte du big data.

Il est par ailleurs à noter qu'utiliser le troisième mode de distribution peut s'avérer être plus intéressant que le second dans certains cas. Par exemple, pour certaines organisations voulant certes protéger leur processus (qui peut impliquer beaucoup d'activités) mais à un coût raisonnable (à travers l'utilisation d'un nombre limité de fragments). En effet, nous soulignons qu'utiliser un nombre important de fragments peut être coûteux, et donc le troisième mode de distribution peut être considéré comme étant un bon compromis entre coût et sécurité.

3.6.3 Mesure du niveau d'obfuscation après introduction des faux fragments

Cet ensemble d'expériences a pour objectif d'étudier l'effet de l'introduction des faux fragments sur le niveau d'obfuscation des processus métiers. En effet, elles nous permettent de mesurer le niveau de difficulté moyen rencontré par les fournisseurs pour former une coalition. Pour ce faire, nous comparons des collaborations générées de trois manières différentes :

- Distribution aléatoire simple.
- Distribution basée uniquement sur nos règles de séparation (section 3.3.2.2).
- Distribution basée sur nos règles avec introduction de faux fragments (section 3.4).

Nous générons 1000 instances de collaborations pour chaque processus métier et pour chaque mode de distribution.

3.6.3.1 Résultats

La première expérience représente le niveau d'obfuscation obtenu en utilisant notre première métrique (i.e. complexité d'une collaboration 3.5.2.1). Les résultats obtenus sont illustrés dans la figure 3.23.

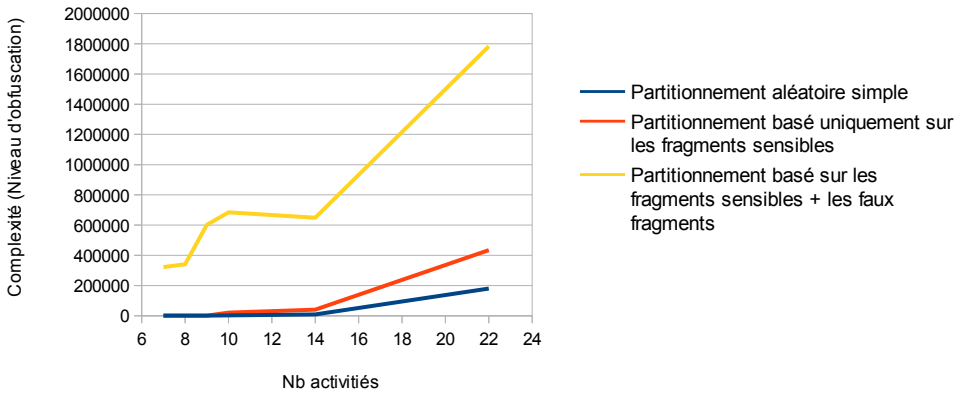
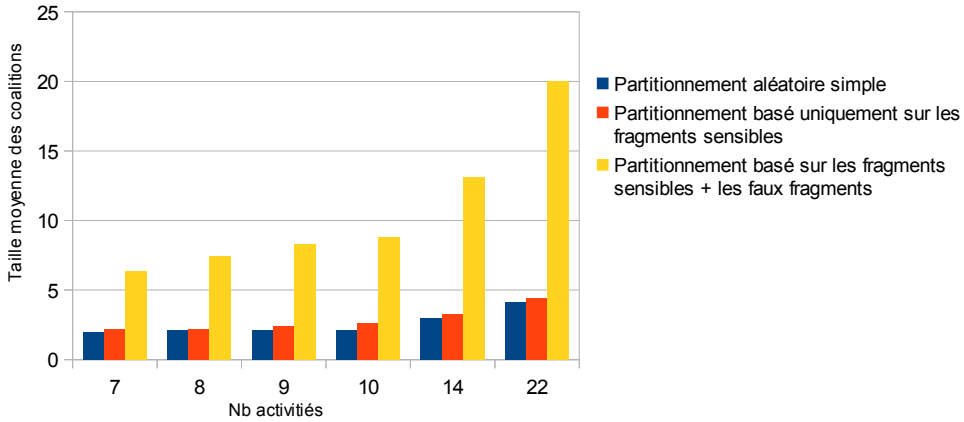
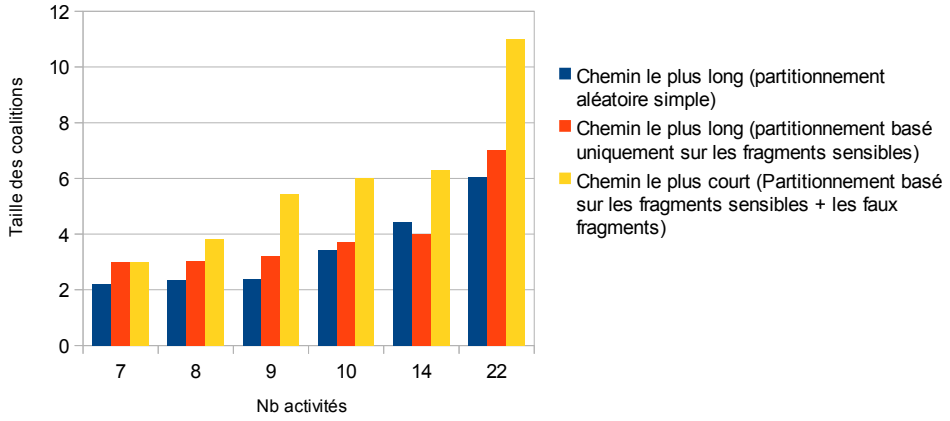


FIGURE 3.23 – Complexité des trois distributions



(a) Taille moyenne des coalitions



(b) Taille des coalitions

FIGURE 3.24 – Complexité de coalition conformément aux trois modes de distribution

Comme illustré dans la figure 3.23, le mécanisme de faux fragments permet d'augmenter le niveau d'obfuscation des collaborations linéairement (à partir de processus de plus de 14 tâches), en comparaison à un déploiement basé uniquement sur les règles de séparation (section 3.3.2.2) ou un déploiement aléatoire simple. En effet, le mécanisme de faux fragments permet d'augmenter le nombre de clouds ainsi que le nombre de messages échangés entre les fragments de la collaboration, i.e. le facteur NoF de la métrique.

Il est à noter que le niveau d'obfuscation d'un processus ne dépend pas seulement de sa taille, mais également de sa structure (e.g. nombre de passerelles, type des passerelles, etc.). En effet, puisque l'objectif primaire de notre approche est la protection contre les clouds qui déploient les fragments sensibles (généralement qui précèdent ou suivent une passerelle), nous ajoutons plus de faux fragments dans les processus qui comprennent plus de passerelles, augmentant par conséquent le nombre de clouds et messages échangés, permettant donc d'obtenir un plus grand niveau d'obfuscation (conformément à la métrique de la section 3.5.2.2).

Par ailleurs, comme introduit précédemment, cette métrique ne se base que sur la séparation des fragments sensibles pour le calcul de la complexité, en ignorant la distance qui les sépare. Ainsi, pour justifier et renforcer de l'efficacité de l'introduction des faux fragments sur la difficulté de reconstruire le processus de la part d'un éventuel cloud malicieux, nous effectuons une seconde expérimentation dont l'objectif est de calculer l'effort moyen apporté par un cloud pour former une coalition.

Les résultats obtenus suite à cette seconde expérimentation sont illustrés par la figure 3.24a. Ces résultats furent obtenus en utilisant la métrique $AverageLength_{Coal}$ introduite dans la section 3.5.2.2, qui calcule la taille moyenne des chemins séparant les tâches sensibles des collaborations générées à travers les trois modes de distribution. Comme indiqué, ce mécanisme de faux fragments permet d'offrir un plus long chemin séparant ces tâches sensibles (proportionnel au niveau d'obfuscation) qu'une distribution aléatoire ou une distribution basée uniquement sur nos règles de séparation.

De plus, et comme indiqué dans la figure 3.24b qui représente les résultats de notre expérimentation en utilisant les métriques $LongestLength_{Coal}$ et $ShortestLength_{Coal}$ (section 3.5.2.2) (qui calculent respectivement le chemin le plus court pour le mécanisme de faux fragments et le chemin le plus long pour les deux autres modes de distribution), le chemin le plus court du mécanisme faux fragment a dans le pire cas, la même taille que les plus long chemins des autres mode de distribution, résultant donc en un niveau d'obfuscation aussi bon dans le pire cas, et meilleur dans les autres.

3.6.3.2 Discussion

Les résultats d'expérimentations démontrent l'efficacité de l'ajout des faux fragments sur les niveaux d'obfuscation des processus. En effet, puisque ce mécanisme implique de nouvelles contraintes de séparation (comme introduit dans la section 3.4) en plus de celles définies dans la section 3.3.2.2, il permet de minimiser le nombre d'activités par fragments, et d'augmenter le nombre de fournisseurs, réduisant le taux d'informations quant au savoir-faire dans les fragments. Ceci rend plus difficile l'ingénierie inverse du rôle de chaque fragment dans la collaboration.

3.7 Conclusion

Dans ce chapitre nous avons présenté la première partie de notre contribution, qui consiste en un mécanisme d'obfuscation basé sur la nature des fragments. Après identification des fragments *critiques*, nous avons présenté l'ensemble des règles utilisées pour l'assignation des tâches du processus aux différents clouds. Ces règles d'assignation nous ont permis de générer un ensemble de contraintes de *séparation*, permettant d'éviter le déploiement de deux tâches complémentaires sur le même cloud. Nous avons par la suite appliqué cette méthode sur notre exemple de motivation de la section 1.1, et l'avons représenté sous forme d'une collaboration respectant les contraintes introduites précédemment.

La deuxième partie de ce chapitre fut dédiée à notre seconde contribution, i.e. la complexification d'un modèle de processus par ajout de faux fragments, dans le but de contrer les éventuelles coalitions des clouds appartenant à la collaboration. Nous avons tenté de répondre à trois questions primordiales, i.e. quand considérer ces faux fragments ? où les insérer ? et enfin comment les déployer dans un environnement mutli-cloud ? Les réponses à ces questions nous ont permis d'introduire ces fragments de manière à augmenter le nombre et la tailles des chemins séparant les tâches complémentaires, réduisant de ce fait la probabilité d'une coalition.

La troisième partie du chapitre a introduit un ensemble de métriques que nous avons utilisé pour vérifier l'exactitude et l'efficacité de nos méthode à travers un ensemble d'expérimentations. Les résultats obtenus ont confirmé nos hypothèse de départ, i.e une séparation de fragments de processus en se basant sur la nature des fragments (sensibles ou

pas) en premier lieu et sur les faux fragments en second lieu, permet de complexifier la logique du processus.

Cependant, malgré la définition de méthodes de protection du savoir-faire de processus, des risques subsistent toujours, i.e. il est possible qu'un cloud malicieux réussisse à convaincre tous les autres clouds de la collaboration à conspirer, même si la probabilité que cela se produise est infime. Il est donc nécessaire de pouvoir calculer ce niveau de risque subsistant. Dans cette optique, nous proposons dans le prochain chapitre une métrique pour le calcul du risque auquel est exposée une collaboration lors d'un déploiement multi-cloud.

Chapitre 4

Gestion du risque

Sommaire

4.1 Introduction

Le second point abordé au cours de cette thèse concerne la quantification des risques de sécurité auxquels sont exposés les processus métiers lors d'un déploiement cloud. Comme introduit précédemment, les entreprises ont besoin de solutions afin d'adapter leur conception et leur modélisation pour un déploiement fiable de leurs processus dans un contexte cloud. Ainsi, pour répondre à ce défi, de nouvelles méthodes et outils doivent être définis.

Dans cette optique, nous avons introduit dans le chapitre 3 deux approches complémentaires [GANYG15][NGY+18][NGY+16], qui permettent d'obfusquer un processus métier en premier lieu, et de complexifier sa structure en second lieu pour un déploiement plus sécurisé. Cependant, malgré l'efficacité de ces méthodes pour augmenter la complexité d'un

processus métier, des risques subsistent. Il est donc nécessaire de pouvoir les quantifier afin de choisir la configuration la moins risquée (tâches/clouds).

Nous présentons dans ce chapitre notre modèle de risque, permettant de calculer le risque auquel est exposé un processus métier dans un environnement cloud. Contrairement aux différentes méthodes proposées dans la littérature qui se basent sur les données du cloud pour le calcul le niveau de risque, notre approche se base sur les données des clients [NGYT17].

4.2 Coalition de clouds

Dans le contexte d'une collaboration de BP fragments, nous considérons que pour qu'un attaquant puisse découvrir une information sensible, il est nécessaire qu'il possède à son niveau à la fois un fragment sensible ainsi que son fragment complémentaire. Ces fragments complémentaires si réunis ensemble, permettent à un cloud malicieux de découvrir une partie de la logique du processus, à cause de l'importance des informations qu'ils manipulent. Ainsi, afin d'éviter ces fuites d'informations, des contraintes de séparation $separate(f_i, f_j)$ [22] sont généralement appliquées. Ces contraintes permettent de déployer les fragments complémentaires sur différents clouds. Dans notre contexte, il s'agit des fragments *décision* et son fragment complémentaire, *synthèse* et son fragment complémentaire et des fragments *alternatifs*. Pour plus de détails, se référer à la section 3.3.1 du chapitre 3.

Afin d'accéder simultanément à ces deux informations (généralement déployées sur différents clouds pour des raisons de sécurité), il est nécessaire d'une part de trouver un chemin entre les deux tâches complémentaires sensibles, et d'autre part d'arriver à *conspirer* avec tous les clouds se trouvant sur ce chemin les séparant.

Par exemple, la figure 4.1 représente une collaboration de trois clouds (C_1, C_2, C_3) dans laquelle C_1 et C_3 possèdent chacun une tâche sensible, qui est complémentaire à celle se trouvant sur l'autre cloud. Puisqu'il n'existe aucun message direct entre C_1 et C_3, aucun d'eux n'est directement au courant de cette collaboration. Cependant, en suivant les différents messages échangés (*send* et *receive*), ils peuvent reconstruire le chemin les reliant. Par exemple, si C_1 suit le message $send_{01}$ et arrive à convaincre C_2 à conspirer, il sera en position de voir tous les messages que C_2 envoie et reçoit. Par conséquent, à partir de C_2 et en suivant son message $send_{02}$, C_1 sera au courant de la participation de C_3 à la collaboration en suivant le chemin $C_1 - C_2 - C_3$ (construit à partir de l'échange des messages). En utilisant le $receive_{02}$ dans un premier temps, et en conspirant avec C_2, C_3 sera dans la possibilité de reconstruire le chemin le reliant à C_1 (typiquement $C_3 - C_2 - C_1$).

Ainsi, nous partons dans notre approche du principe que plus il y a de clouds qui doivent conspirer, plus difficile ce sera pour eux de former une coalition. Par conséquent, *la collaboration sera moins risquée* e.g. convaincre deux clouds à conspirer est plus simple que d'en convaincre cinq.

22. f_i et f_j sont des fragments complémentaires

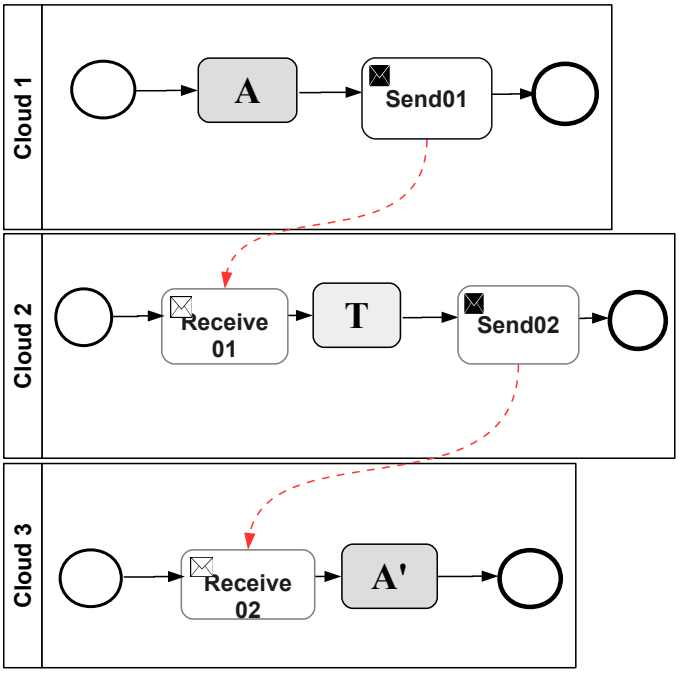


FIGURE 4.1 – collaborations de fragments

4.3 Approche globale pour le calcul du risque

4.3.1 Formule globale de calcul du niveau de risque

L'approche de calcul du risque côté client proposée dans cette thèse est principalement basée sur le concept introduit dans la section précédente, i.e. plus il y a de clouds qui séparent les tâches sensibles, et moins risquée sera la collaboration.

En général, le risque est définit par la probabilité d'un incident ayant un certain impact négatif [Aus04]. Dans le contexte de la sécurité des IT , où les composants informatiques (e.g. matériel, réseau, etc.) soutiennent les actifs de l'entreprise (e.g informations, processus, etc.), le risque peut être défini d'une manière plus détaillée comme étant la probabilité qu'une menace, accidentellement ou intentionnellement exploite une ou plusieurs vulnérabilités (failles ou faiblesses dans les procédures de sécurité du système, dans la conception, la mise en œuvre ou les contrôles internes) de l'environnement informatique avec un impact négatif (destruction, altération, vol, etc).

Conformément à ces définitions, le risque auquel est exposé un artéfact [23] peut être formellement défini comme suit [Aus04, Eur09, May09] :

23. peut être considéré comme un programme, un script, une page web

$$\mathbf{Risque}(\mathbf{a}) = \mathbf{V}(\mathbf{a}) \times \mathbf{T}(\mathbf{a}) \times \mathbf{I}(\mathbf{a}) \tag{4.1}$$

Où

- V : la vulnérabilité de l'artéfact.
- T : La menace sur l'artéfact.
- I : l'impact sur l'artéfact.

4.3.2 Approche générale pour la mesure du risque d'une collaboration

L'objectif de la métrique introduite dans ce chapitre est le calcul du niveau de risque auquel est exposée une configuration donnée (tâches affectées aux clouds). Cette métrique est basée sur la définition du risque introduite précédemment (équation 4.1).

Cette définition peut avoir plusieurs interprétations conduisant à différents modèles de risques. En effet, en se basant sur cette définition, Goettelmann et al. [GDG+14] ont proposé un modèle de risque qui repose sur les données des fournisseurs cloud. Ce type de modèle est appelée *approche de gestion du risque côté cloud*. Nous proposons dans cette thèse une nouvelle approche où la gestion des risques dépend des données des *clients cloud*. Cette méthode met en avant les tâches sensibles comme étant les éléments les plus importants à préserver.

Plus précisément, le principe de notre approche du calcul du risque côté client est pour une configuration cloud donnée, comme suit :

- Dans un premier temps, calculer le risque auquel est exposée chaque tâche sensible.
- Ensuite, élaborer une valeur de risque globale pour l'ensemble de la collaboration comme la somme de tous les risques des tâches sensibles.

4.3.3 Calcul du niveau de risque d'une tâche sensible

Pour qu'un attaquant puisse accéder à des informations sensibles, il doit posséder à son niveau à la fois un fragment sensible ainsi que son fragment complémentaire. Par conséquent, nous formalisons le risque auquel est exposée une tâche sensible comme la probabilité qu'un cloud exécutant une tâche sensible découvre sa tâche complémentaire correspondante, ou plus précisément de découvrir un chemin entre la tâche sensible et la tâche complémentaire en utilisant les opérations *send* de ses fragments, et/ou en faisant un retour en arrière en utilisant les opérations *receive* de son fragment, comme illustré dans la figure 4.1.

Par ailleurs, nous considérons que le risque auquel une tâche sensible est exposée dépend directement des paramètres suivants :

- l'intégrité du cloud exécutant cette tâche, i.e. plus il est fiable et moins il est probable qu'il accepte de faire partie d'une coalition.
- l'intégrité des clouds se trouvant sur les chemins (*send*/*receive*) séparant une tâche sensible et sa tâche complémentaire. Plus précisément, nous considérons que moins il y a de clouds malhonnêtes sur ce chemin, et moins la tâche sensible est vulnérable.

Il est à noter que puisqu'il peut exister plusieurs chemins reliant deux tâches sensibles complémentaires, nous calculons d'abord le niveau de risque de la tâche sensible sur chaque chemin la séparant de sa tâche complémentaire, et enfin calculons le niveau de risque global auquel elle est exposée à partir de ces différents niveaux.

4.4 Modèle de risque

Cette section introduit notre modèle pour le calcul du niveau risque auquel est exposé l'ensemble du processus lors d'un déploiement dans un environnement multi-cloud.

Pour ce faire, nous définissons en premier lieu la notion de chemin nécessaire à la bonne compréhension de notre méthode. Nous introduisons par la suite le niveau de risque auquel est exposée une tâche sensible sur chaque chemin p_j, et formalisons le niveau de risque global d'une tâche sensible sur l'ensemble des chemins la séparant de sa tâche complémentaire. Enfin, nous introduisons le risque global auquel est exposée l'ensemble de la collaboration.

4.4.1 Chemin séparant deux tâches complémentaires $p_j(t_i, t'_i)$

Un chemin $p_j(t_i, t'_i)$ reliant deux tâches sensibles complémentaires t_i et t'_i est défini comme l'ensemble des clouds exécutant des tâches du processus sur le chemin logique j séparant t_i de t'_i.

Par exemple, dans la figure 4.1, le chemin qui sépare la tâche sensible A de sa tâche complémentaire A' est $C_1 - C_2 - C_3$.

4.4.2 Niveau de risque d'une tâche sensible sur un chemin p_j

Conformément à la définition 4.1, le niveau de risque d'une tâche sensible t_i sur un chemin p_j est défini comme suit :

$$\mathbf{Risque}(\mathbf{t_i}, \mathbf{p_j}) = \mathbf{Vulnerabilité}(\mathbf{t_i}) \times \mathbf{Menaces}(\mathbf{t_i}, \mathbf{p_j}) \times \mathbf{Impact}(\mathbf{t_i}, \mathbf{p_j}) \tag{4.2}$$

- **Vulnerabilité**$(\mathbf{t_i})$ la vulnérabilité d'une tâche provient du cloud qui l'exécute, ou à quel point il est prêt à conspirer avec d'autres clouds.
- **Menaces**$(\mathbf{t_i}, \mathbf{p_j})$ la menace à laquelle est exposée une tâche t_i, ou plus précisément à quel point les clouds se trouvant sur le chemin la séparant de sa tâche complémentaire sont prêts à conspirer.
- **Impact**$(\mathbf{t_i}, \mathbf{p_j})$ représente l'impact de la coalition des clouds se trouvant sur le chemin $p_j(t_i, t'_i)$.

Il est à noter que si la menace et l'impact varient d'un chemin à un autre, la vulnérabilité est une valeur fixe (dépend uniquement du cloud dans lequel la tâche est déployée).

4.4.2.1 Vulnérabilité d'une tâche

La vulnérabilité d'une tâche provient du cloud qui l'exécute, i.e. à quel point est-il prêt à conspirer avec d'autres clouds ? Par conséquent, nous relions directement la vulnérabilité d'une tâche au niveau de réputation du cloud qui la déploie. En effet, nous considérons que plus un cloud a une bonne réputation, moins il sera enclin à conspirer et moins la tâche sera exposée aux menaces.

Formellement, la vulnérabilité d'une tâche t_i (t_i étant une valeur normalisée appartenant à l'intervalle $]0;1[$) est définie comme suit :

$$\textbf{Vulnerabilité}(\mathbf{t_i}) = \mathbf{1} - \mathbf{rep}(\mathbf{c}) \tag{4.3}$$

Où :

1. $\mathbf{t_i}$: représente la tâche sensible.
2. $\mathbf{c}$: le cloud qui déploie t_i.
3. $\mathbf{rep}(\mathbf{c})$: la réputation du cloud c.

Mesurer le niveau de réputation des clouds ne fait pas partie de nos travaux. Plusieurs travaux ont été proposés pour mesurer dynamiquement cette réputation [JIB07]. Notre seule hypothèse est qu'elle peut être normalisée dans un intervalle [0,1] : 0 pour le moins réputé et 1 pour le meilleur.

4.4.2.2 Menaces sur la tâche t_i relativement à un chemin p_j

Rappelons que plus il y a de clouds qui doivent conspirer (plus de clouds sur le chemin séparant les tâches complémentaires), plus difficile sera pour eux de former une coalition. Ainsi, la menace qui pèse sur une tâche sensible peut être atténuée en augmentant le nombre de clouds la séparant de sa tâche complémentaire. Par conséquent, nous considérons que la menace qui pèse sur une tâche sensible peut être reliée à la longueur du chemin la séparant de sa tâche complémentaire. Et plus précisément, à la réputation des clouds sur ce chemin : plus il y a de clouds avec une bonne réputation, i.e. considérés comme étant des clouds de confiance, plus il sera difficile pour un cloud malicieux de former une coalition.

De plus, nous considérons que cette difficulté n'est pas linéaire. Augmenter le nombre de clouds séparant les tâches complémentaires réduit de beaucoup les menaces auxquelles elles sont exposées, suite à la difficulté de convaincre un cloud à faire partie de la coalition. En effet, comme introduit plus haut, le nombre de clouds de la coalition affecte sensiblement les menaces auxquelles est exposée une tâche sensible, i.e plus ils sont nombreux, et plus il sera difficile pour un cloud malicieux de convaincre tous ces clouds de faire partie de la coalition, réduisant par conséquent exponentiellement les menaces.

Ainsi, la menace qui pèse sur une tâche t_i relativement à un chemin p_j peut être définie comme suit :

$$\mathbf{Menace(t_i, p_j)} = \mathbf{s_t} * \left(\frac{\mathbf{1}}{\mathbf{e^{card(p_j(t_i,t_{i'}))} \sum_{j=1}^{card(p_j(t_i,t_{i'}))} rep(C_j)}}\right) \tag{4.4}$$

où

- $\mathbf{s_t}$: sensitivité de la tâche t ($\in [0,1]$).
- $\mathbf{card(p_j(t_i, t_{i'}))}$: la cardinalité de l'ensemble $p_j(t_i, t_{i'})$, i.e. le nombre de clouds sur le chemin séparant les tâches complémentaires sur le chemin p_j.
- $\mathbf{rep(c_j)}$: la réputation du cloud j.

Il est à noter que la notion de sensitivité d'une tâche est introduite dans [NGY+15]. C'est une valeur définie par le concepteur dans l'intervalle [0,1] : 0 pour la moins sensitive et 1 pour la plus sensitive.

4.4.2.3 Impact des menaces sur la tâche t_i relativement à un chemin p_i

L'impact d'une menace peut être corrélé au nombre de clouds impliqués dans la coalition, i.e. plus il y a de clouds dans la coalition, plus l'impact sera important étant donné que tout cloud faisant partie de la coalition pourra profiter de la faille de sécurité. Par conséquent, nous avons décidé de mesurer l'impact d'une menace comme le taux (pourcentage) de clouds impliqués dans la coalition.

L'équation 4.5 calcule l'impact d'une menace pour une tâche (t_i sur un chemin p_j).

$$\mathbf{Impact(t_i, p_j)} = \frac{\mathbf{card(p_j(t_i, t_{i'}))}}{\mathbf{cardSys}} \tag{4.5}$$

où

1. $\mathbf{card(p_j(t_i, t_{i'}))}$: la cardinalité de l'ensemble $p_j(t_i, t_{i'})$, i.e. le nombre de clouds séparant les tâches complémentaires sur le chemin p_j.
2. **cardSys** : la cardinalité du système, i.e. le nombre de clouds impliqués dans la collaboration.
3. $\mathbf{rep(c_j)}$: la réputation du cloud j.

Il est possible par ailleurs de considérer que cette définition de l'impact est inconsistante avec l'idée du risque qui diminue lorsque la longueur du chemin augmente. Cependant, l'impact augmente linéairement alors que la menace diminue exponentiellement. Par conséquent, le risque global diminue rapidement lorsque la longueur du chemin augmente.

4.4.2.4 Risque global d'une tâche (considérant tous les chemins)

Le risque d'une tâche t_i est définie comme la somme de tous les risques sur tous les chemins possibles.

$$\mathbf{Risque(t_i)} = \sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{Risque(t_i, p_j)} \tag{4.6}$$

où

- **m** : nombre de chemins séparant t_i de sa tâche complémentaire $t_{i'}$.
- **Risque($\mathbf{t_i}$, $\mathbf{j}$)** : risque de la tâche t_i en sélectionnant le chemin j.

4.4.3 Niveau de risque d'une collaboration

Le niveau de risque auquel est exposé l'ensemble du processus représente la somme des risques auxquels sont exposées toutes ses tâches sensibles. Ce niveau de risque est calculé comme suit :

$$\mathbf{Risque(collaboration)} = \sum_{\mathbf{k=1}}^{\mathbf{p}} \mathbf{Risque(t_k)} \tag{4.7}$$

où

- **p** : nombre de tâches sensibles.
- **Risque($\mathbf{t_k}$)** : niveau de risque de la tâche t_k sur tous les chemins la reliant à sa tâche complémentaire.

4.5 Application à l'exemple de motivation : cas de la tâche sensible VDC et sa tâche complémentaire DTR

Pour illustrer le modèle de risque introduit dans la section 4.4, nous l'appliquons à notre exemple de motivation afin de mesurer le risque qu'un cloud malicieux puisse relier la tâche sensible VDC déployée dans le fragment 1, à sa tâche complémentaire DTR déployée dans le fragment 14. Pour ce faire, nous nous basons sur la figure 4.2 (déjà introduite dans le chapitre 3) qui représente le déploiement du processus de prêt bancaire.

La figure 4.3 représente l'ensemble des chemins qui relient les tâches VDC et DTR dans la figure 4.2. La partie supérieure représente les chemins qui peuvent être découverts à travers les messages *receive*, tandis que la partie inférieure représente ceux découverts à travers les messages *send*. Nous associons à chaque chemin sa taille, i.e. le nombre de clouds qu'il inclut.

4.5.1 Étape 1

Dans la première étape, nous déterminons l'ensemble des chemins qui séparent les tâches VDC et DTR. Par exemple, en commençant du fragment $SP1$ et en suivant l'opération $send_{287}$, le chemin de taille 4 et composé des sous-processus ($SP1 - SP5 - SP10 -$

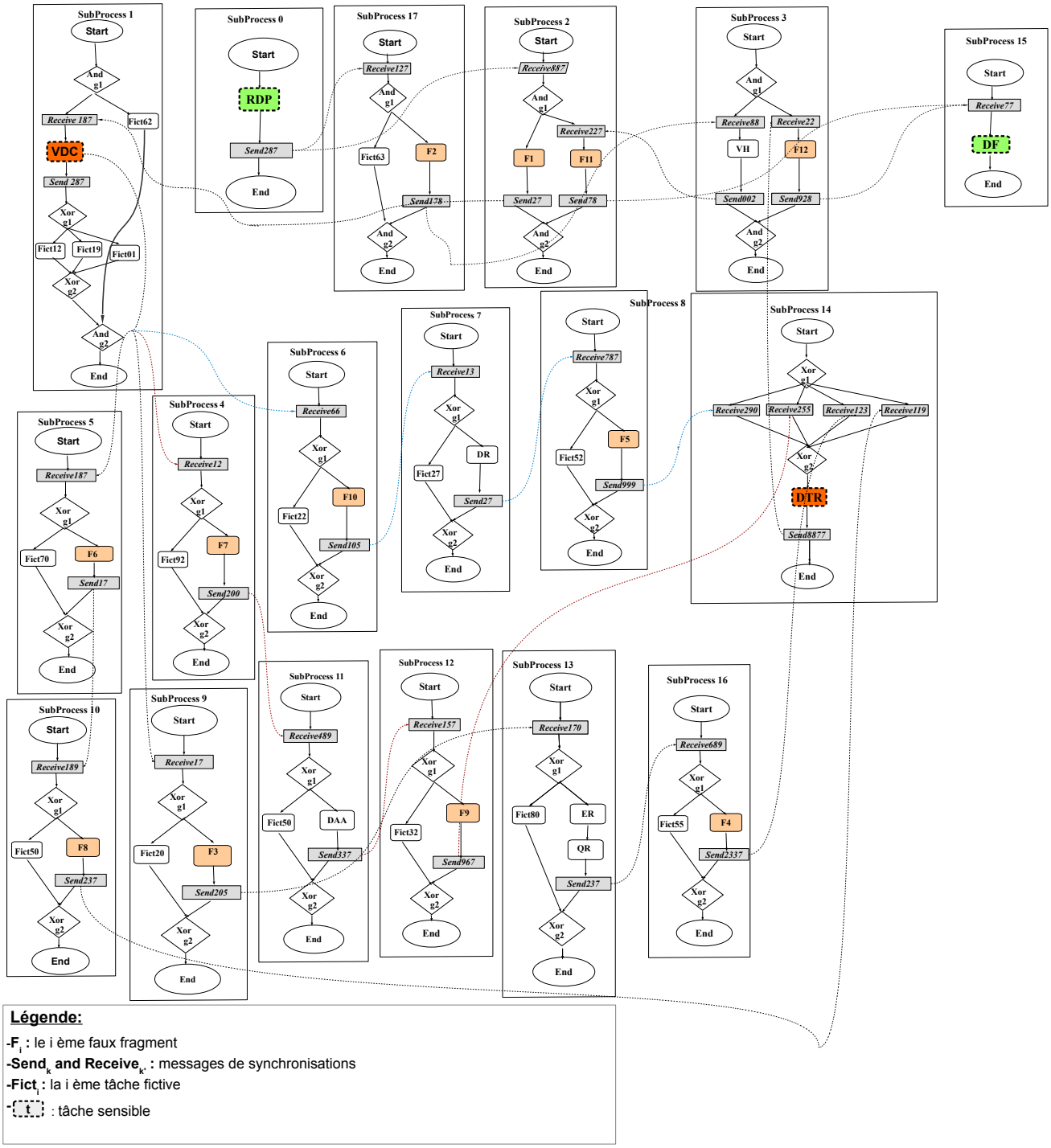


FIGURE 4.2 – Déploiement du processus de prêt bancaire avec faux fragments intégrés

$SP14$) (mis en évidence dans la figure 4.2 et illustré dans la partie basse de la figure 4.3) conduit de VDC à DTR.

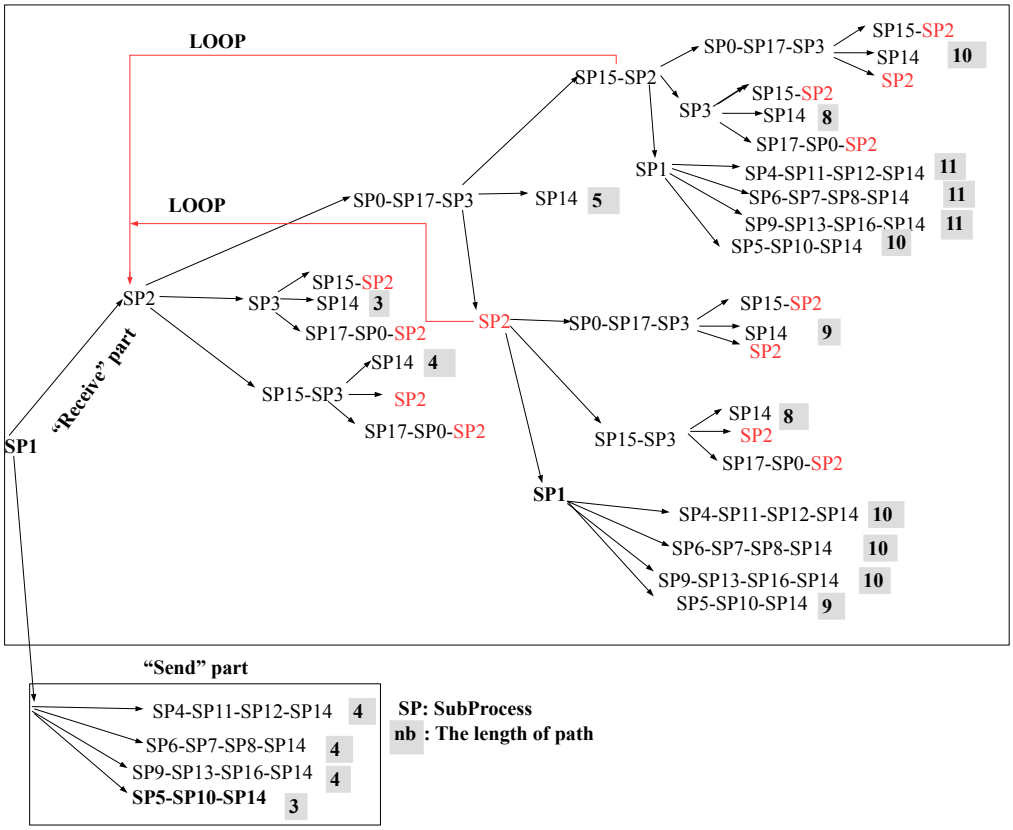


FIGURE 4.3 – Chemins possibles entre la tâche VDC du fragment 1 et sa tâche complémentaire DTR du fragment 14

Chemins	**Taille du chemin**	**Vulnéraniblité**	**Menace**	**Impact**	**Niveau de risque**	**Niveau de risque normalisé**
SP2-SP0-SP17-SP13-SP14	5	0,5	$21.10^{-}8$	0,27	0.000000028	0.0000030
SP2-SP3-SP14	3	0,5	$31.10^{-}4$	0,16	$92.10^{-}4$	1
SP2-SP15-SP3-SP14	4	0,5	$10.10^{-}5$	0,22	$11.10^{-}6$	0,00119
SP4-SP11-SP12-SP14	4	0,5	$23.10^{-}5$	0,22	$25.10^{-}6$	0,0027
SP6-SP7-SP8-SP14	4	0,5	$17.10^{-}4$	0,22	$19.10^{-}5$	0,0206
SP9-SP13-SP16-SP14	4	0,5	$95.10^{-}7$	0,22	$10.10^{-}7$	0,00010
SP5-SP10-SP14	3	0,5	0,014	0,16	0,0011	0,119
SP2-SP0-SP17-SP3-SP15-SP2-SP0-SP17-SP3-SP14	10	0,5	$67.10^{-}25$	0,55	$18.10^{-}25$	$19.10^{-}23$
SP2-SP0-SP17-SP3-SP15-SP2-SP3-S14	8	0,5	$17.10^{-}16$	0,44	$39.10^{-}17$	$42.10^{-}15$
SP2-SP0-SP17-SP3-SP15-SP2-SP1-SP4-SP11-SP12-SP14	11	0,5	$33.10^{-}27$	0,61	$102.10^{-}27$	$11.10^{-}24$
SP2-SP0-SP17-SP3-SP15-SP2-SP1-SP6-SP7-SP8-SP14	11	0,5	$82.10^{-}25$	0,61	$25.10^{-}25$	$27.10^{-}23$
SP2-SP0-SP17-SP3-SP15-SP2-SP1-SP9-SP13-SP16-SP14	11	0,5	$20.10^{-}31$	0,61	$154.10^{-}31$	0
SP2-SP0-SP17-SP3-SP15-SP2-SP1-SP5-SP10-SP14	11	0,5	$73.10^{-}22$	0,55	$202.10^{-}22$	$210.10^{-}20$
SP2-SP0-SP17-SP3-SP2-SP0-SP17-SP13-SP14	9	0,5	$81.10^{-}22$	0,5	$20,73.10^{-}22$	$22.10^{-}20$
SP2-SP0-SP17-SP3-SP2-SP15-SP3-SP14	8	0,5	$17,89.10^{-}16$	0,44	$39,37.10^{-}17$	$42.10^{-}15$
SP2-SP0-SP17-SP3-SP2-SP1-SP4-SP11-SP12-SP14	10	0,5	$13,50.10^{-}23$	0,55	$37,13.10^{-}24$	$4.10^{-}22$
SP2-SP0-SP17-SP3-SP2-SP1-SP6-SP7-SP8-SP14	10	0,5	$20,02.10^{-}21$	0,55	$55,11.10^{-}22$	$59.10^{-}20$
SP2-SP0-SP17-SP3-SP2-SP1-SP9-SP13-SP16-SP14	10	0,5	$45,29.10^{-}27$	0,55	$12,45.10^{-}27$	$13.10^{-}15$
SP2-SP0-SP17-SP3-SP2-SP1-SP5-SP10-SP14	9	0,5	$10,91.10^{-}18$	0,5	$27,28.10^{-}19$	$29.10^{-}17$
$\boldsymbol{Risk(VDC) = \sum_{i=1}^{n} Risk(t_i), n \in SetOfPaths}$			***0.00055***			***0.06***

Table 4.1 – Niveaux de risque de VDC

Cloud	Réputation	Cloud	Réputation	Cloud	Réputation
$SP0$	0,8	$SP6$	0,1	$SP12$	0,2
$SP1$	0,5	$SP7$	0,4	$SP13$	0,7
$SP2$	0,4	$SP8$	0,2	$SP14$	0,8
$SP3$	0,6	$SP9$	0,6	$SP15$	0,3
$SP4$	0,3	$SP10$	0,3	$SP16$	0,7
$SP5$	0,2	$SP11$	0,7	$SP17$	0,3

TABLE 4.2 – Réputations des clouds

4.5.2 Étape 2

Après sélection de l'ensemble des chemins, nous calculons le risque auquel est exposée la tâche VDC sur chaque chemin à partir de la vulnérabilité, les menaces et l'impact.

Par exemple, en appliquant les équations 4.3, 4.4 et 4.5 au chemin ($SP1-SP5-SP10-SP14$) on retourne respectivement pour la tâche VDC sa vulnérabilité, ses menaces et l'impact (en se basant sur les réputations des clouds illustrés dans le tableau 4.2) comme suit :

$Vulnerabilité(VDC) = (1 - 0,5) = 0,5$

$Menaces(VDC) = 0.7 \times (\frac{1}{e^{4\sum_{j=1}^{4} Rep(C_j)}})$

Où $C = \{SP2 - SP15 - SP13 - SP14\}$.

Nous obtenons donc

$Menaces(VDC) = 0,7 \times (\frac{1}{e^{4\times 1,8}}) = 0.00052$

L'impact de cette menace est calculé quant à lui comme suit :

$Impact(VDC) = (\frac{4}{18}) = 0,22$

Nous calculons donc le risque auquel est exposée VDC sur ce chemin en utilisant l'équation 4.1 comme suit
$Risque(VDC) = 0,5 \times 0.00052 \times 0,22 = 57.10^{-}6$.

Nous appliquons les mêmes étapes pour tous les chemins de la figure 4.3. Les résultats obtenus sont illustrés dans la table 4.1. Comme illustré, certains chemins permettent d'assurer que la tâche ne soit exposée qu'à un petit niveau de risque. En effet, plus long est le chemin, moins la tâche sera exposée à des risques de divulgation (e.g. un chemin composé de 12 clouds permet un niveau de risque de $92.10^{-}38$ tandis qu'un chemin de quatre clouds permet un niveau de risque $75.10^{-}6$). Ceci est dû au fait que convaincre douze clouds à conspirer est bien plus difficile que de n'en convaincre que quatre, résultant donc en de petits niveaux de risques pour les longs chemins.

4.5.3 Etape 3

Après application de la même méthode que celle appliquée à VDC à l'ensemble des tâches sensibles du processus, nous calculons le risque auquel est exposée l'ensemble de collaboration en utilisant l'équation 5.13.

$Risque(collaboration) = \sum_{i=1}^{n} Risk(t_i)$ où n est l'ensemble des tâches sensibles.

4.6 Validation du modèle de risque

Afin de vérifier l'efficacité et l'applicabilité de notre métrique, nous avons conduit un ensemble d'expérimentations dont l'objectif est de mesurer le niveau de risque auquel est exposé un processus lors d'un déploiement cloud. Ce niveau de risque représente la possibilité d'une coalition entre les clouds de le collaboration : plus ce risque est élevé plus il est probable que la coalition ait lieu.

Pour ce faire, nous comparons des collaborations générées à travers les trois modes de distribution suivants :

- Distribution aléatoire.
- Distribution basée sur nos règles de séparation (section 3.3.2.2).
- Distribution basée sur nos règles avec introduction des faux fragments (section 3.4).

Étant donné que le calcul du risque requiert la connaissance du niveau de réputation des clouds qui déploient les tâches (voir section 4.3) ; et puisque nous n'étions pas en mesure de trouver de réelles données (i.e. les fournisseurs ne révèlent pas le niveau de réputation de leur serveurs clouds), nous avons choisi de générer aléatoirement un ensemble de réputations de clouds.

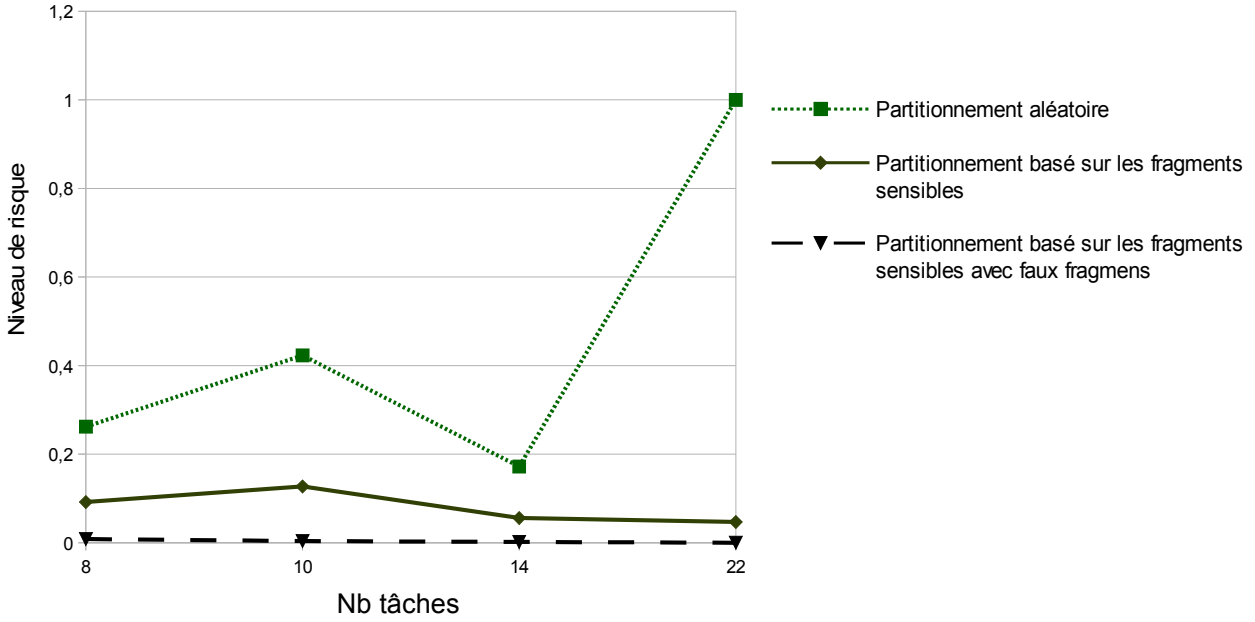
Les résultats d'expérimentation illustrés dans la figure 4.4 représentent le niveau de risque moyen auquel sont exposés les processus métiers lors du déploiement dans le cloud par rapport au nombre de fragments inclus dans les collaborations. Ainsi, et comme illustré dans la figure 4.4a, nous remarquons que les configurations basées sur la séparation des fragments sensibles (section 3.3.2.2) sont moins risquées que celles définies aléatoirement. Cependant, comme nous pouvons le voir, l'introduction des faux fragments couplés à ce mécanisme permet d'obtenir un niveau de risque encore moins élevé. Ceci s'explique par le fait que les faux fragments augmentent la taille des éventuelles coalitions, et donc les rendent moins probables (i.e. convaincre un important nombre de clouds est généralement compliqué).

La figure 4.4b représente le nombre de clouds inclus dans les collaborations. Nous pouvons déduire de la figure que le nombre de clouds n'est pas forcement corrélé au nombre d'activités mais plutôt au nombre de tâches sensibles que le processus inclut, i.e. plus il y a de tâches sensibles plus il y aura de contraintes de séparation, augmentant donc le nombre de clouds de la collaboration (ce qui explique pourquoi le processus ayant 10 activités ait besoin de plus de clouds que le processus de 14 activités).

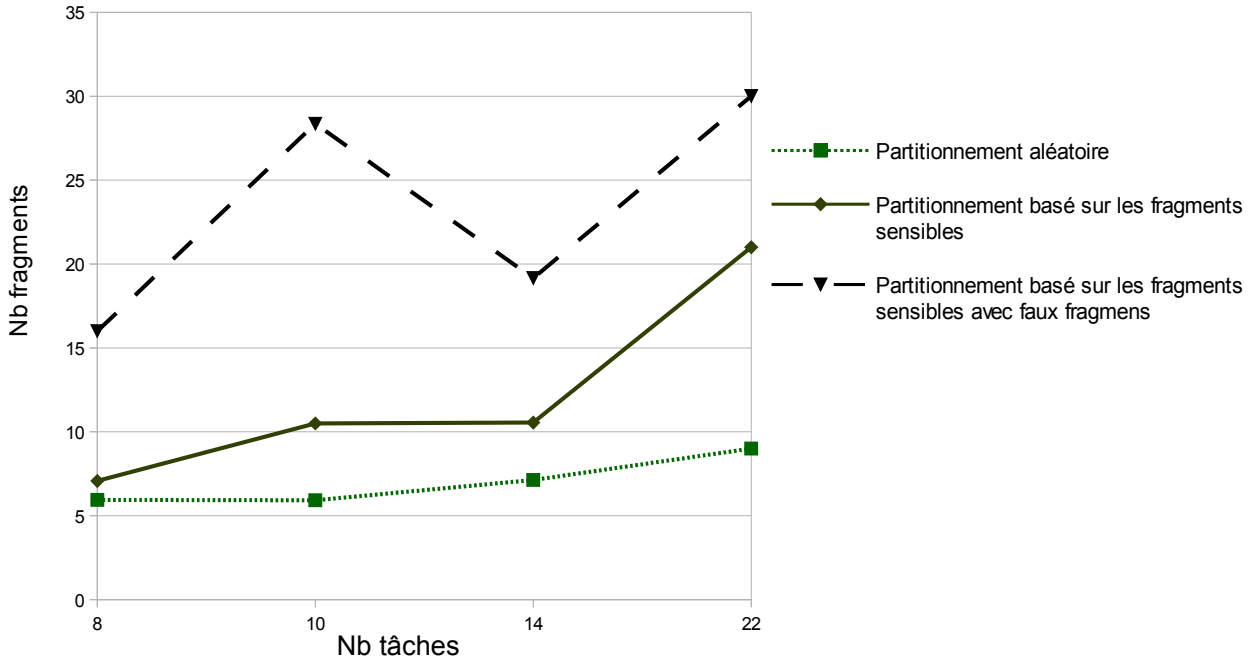
4.6.1 Discussion

Les résultats d'expérimentations démontrent l'efficacité de nos méthodes introduites dans le chapitre 3 sur la réduction du niveau de risque auquel est exposé un processus lors d'un déploiement cloud.

En effet, appliquer les contraintes de séparation introduites dans la section 3.3.2.2 étendues par le mécanisme de faux fragments augmente le nombre de clouds devant conspirer



(a) Niveaux de risques des collaborations



(b) Nombre de fragments

FIGURE 4.4 – Évaluation du risque conformément aux trois distributions

afin de former une coalition, réduisant sa probabilité et donc le risque auquel sont exposées les collaborations. Par ailleurs, en plus d'augmenter le nombre de clouds, nos mécanismes génèrent un nombre plus important de messages échangés entre les fragments de la collaboration, diminuant par conséquent pour un cloud malicieux la probabilité de sélectionner rapidement le chemin le plus court séparant les tâches sensibles.

4.7 Conclusion

Ce chapitre a introduit le second point traité au cours de cette thèse, qui consiste à évaluer le niveau de risque auquel est exposée une collaboration lors d'un déploiement cloud. Après présentation de l'approche globale adoptée pour le calcul du risque, nous avons introduit notre modèle de risque qui calcule en premier lieu le niveau de risque des tâches *sensibles*, et en second lieu le niveau global de risque auquel est exposée l'ensemble de la collaboration. Cette méthodologie fut par la suite illustrée à travers un exemple de calcul du risque en se basant sur l'exemple de motivation de la section 1.

Le prochain chapitre est dédié à la sélection des configurations les plus optimales pour l'exécution des processus métiers, assurant un minimum de sécurité requise et un coût acceptable. Nous proposons à travers notre approche deux algorithmes, dont l'objectif est de respectivement minimiser le risque auquel est exposée la collaboration en limitant cependant le coût accepté, et vice versa.

Chapitre 5

Sécurité versus Coût

Sommaire

5.1 Introduction

Le dernier point abordé durant cette thèse concerne la sélection d'une configuration de clouds pour l'exécution d'un processus métier, assurant un minimum requis de sécurité un coût acceptable.

Comme introduit précédemment, la plupart des travaux dans la littérature y compris les nôtres s'appuient sur la décomposition du processus en sous fragments, qui seront exécutés sur plusieurs clouds. Cependant, cette fragmentation a pour résultat l'augmentation du nombre de clouds faisant partie de la collaboration, i.e. plus le processus est fragmenté, plus il requiert un nombre important de clouds, générant ainsi un coût d'exécution important. Par conséquent, il est nécessaire de trouver des solutions permettant

aux entreprises de profiter des avantages qu'offre la technologie cloud de façon sûre, mais à un coût respectant leur budget.

Par ailleurs, plusieurs configurations sont généralement possibles. Par exemple, en appliquant la méthodologie de fragmentation sur notre exemple de motivation du chapitre 1, le processus d'assurance peut être divisé en plusieurs fragments qui peuvent être exécutés sur différents clouds. De plus, mais de plus chaque fragment peut être affecté à des clouds différents, créant différentes configurations possibles. Les figures 5.1 et 5.2 représentent deux configurations parmi l'ensemble des configurations possibles.

Ces problèmes d'affectation peuvent être simplement résolus en utilisant des *solver* permettant d'obtenir la solution optimale. Cependant, pour des processus de centaines voir de milliers d'activités, cette solution peut rapidement devenir lente et coûteuse. Par conséquent, il est préférable d'utiliser des heuristiques que des solutions exactes qui ne passent pas à l'échelle [MR11].

Nous proposons dans ce chapitre une approche permettant d'aider les concepteurs à optimiser l'utilisation des ressources du cloud en cherchant un équilibre entre le côut de l'exécution du processus et le niveau de sécurité. Pour ce faire, nous proposons deux heuristiques permettant respectivement de minimiser le risque de déploiement du processus en contraignant le coût à un seuil maximum pour la première approche, et de minimiser le coût de déploiement à un seuil de risque maximum pour la seconde.

Ainsi, nous introduisons le principe général de notre approche et quelques notions nécessaires à sa bonne compréhension. Nous présentons ensuite nos heuristiques et introduisons une solution d'optimisation pour chacune d'entre elles, à travers des modèles de réduction du risque et du coût. Enfin, nous introduisons les différents paramètres utilisés en vue de tester nos approches et présentons les différentes expérimentations menées en vue d'évaluer leur efficacité.

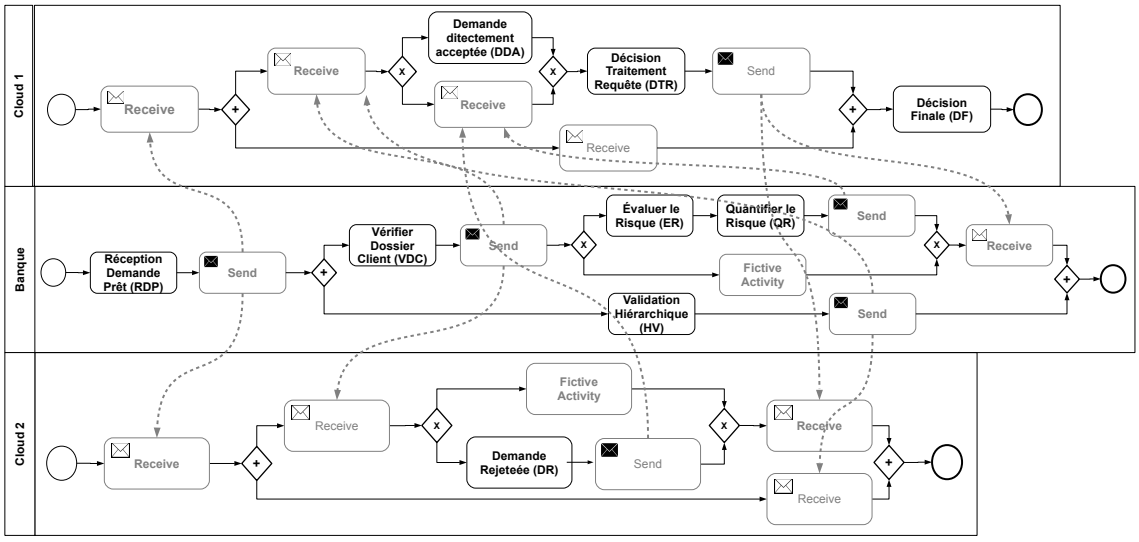


FIGURE 5.1 – Représentation du processus prêt bancaire sous la forme d'une collaboration de fragments de processus

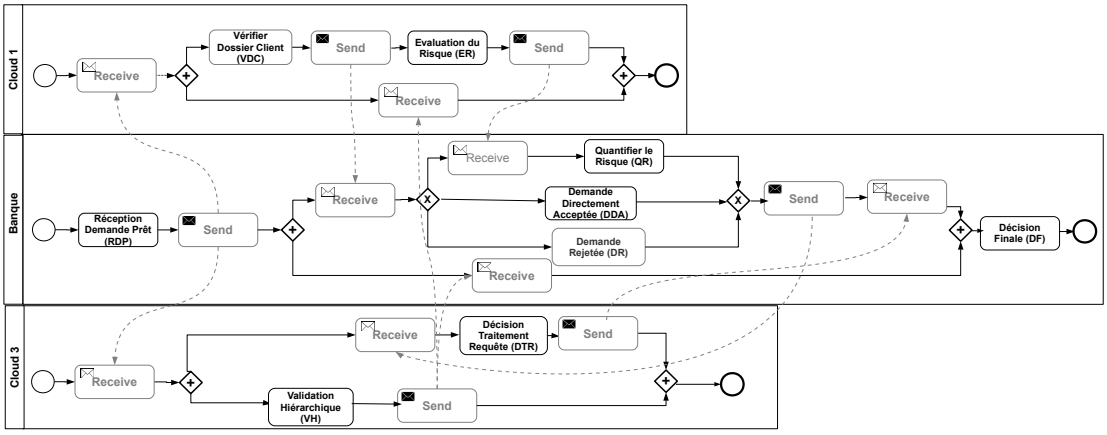


FIGURE 5.2 – Autre exemple de collaboration représentant le processus de prêt bancaire

5.2 Sécurité versus coût

L'entreprise qui déploie le processus peut avoir des besoins bien spécifiques, i.e. augmenter la sécurité de sa stratégie de prise de décisions à un coût qui reste cependant dans son budget, ou bien minimiser le coût d'utilisation du cloud même si la sécurité procurée n'est pas forcement la meilleure qui soit. Pour ce faire, les entreprises requièrent certains supports leur permettant de sélectionner la configuration qui réponde au mieux à leurs besoins.

Nous présentons dans ce qui suit le principe général sur lequel se base notre approche de sélection de configurations. Cependant, avant d'entrer dans plus de détails, nous introduisons quelques notions nécessaires à la bonne compréhension des approches adoptées.

- **Contraintes :** Nous utilisons dans notre approche un ensemble de contraintes ayant pour objectif de séparer/co-localiser certaines tâches selon les besoins de l'utilisateur (voir les sections 3.2.2 et 3.3.2.2 du chapitre 3 pour plus de détails). D'autres besoins pourraient nécessiter l'utilisation de clouds ayant une bonne réputation. Par conséquent, le nombre de contraintes à appliquer varie selon les besoins de l'utilisateur. Dans nos approches, nous nous basons principalement sur des contraintes de séparation. Ces contraintes sont classées par priorité selon les tâches qu'elles affectent, i.e. plus elles sont sensibles, plus ces contraintes sont prioritaires et importantes.

- **Seuils acceptables :** Dans l'optique d'éviter des configurations trop coûteuses ou trop risquées, les concepteurs définissent des seuils de risque et de coût qui permettent d'obtenir des configurations dans des limites acceptables. Ces seuils sont généralement définis sur la base de leur expérience et de leur stratégie de gestion des ressources, et peuvent être ajustés dynamiquement.

  Afin de définir les seuils de risque et de coût, nous nous sommes basés dans nos approches sur une sorte d'itération test qui permet de calculer le risque et le coût minimum pouvant être obtenus. Si d'ores et déjà aucune configuration ne respecte ces seuils à cette itération, ils doivent être augmentés.

  – *Risque minimum* : Peut être obtenu en appliquant toutes les contraintes lors de la décomposition, augmentant ainsi le nombre de clouds de la collaboration, réduisant par conséquent le niveau de risque auquel le processus est exposé.

– *Coût minimum* : Est obtenu en n'appliquant aucune contrainte fonctionnelle (contrainte de séparation), permettant ainsi de déployer le processus sur peu de clouds, minimisant ainsi le coût de déploiement au maximum.

5.2.1 Principe général de notre approche

Nous partons dans notre approche du principe que plus un processus est fragmenté et exécuté sur différents clouds, plus son coût de déploiement sera important, minimisant cependant le risque de coalition de clouds malicieux, et donc de violation des données exposées (voir les chapitres 3 et 4 pour plus de détail sur le principe de coalition). Réciproquement, moins le processus est fragmenté, moins coûteux sera son déploiement, mais plus important sera le niveau de risque auquel il est exposé.

Comme introduit dans le chapitre 2, plusieurs travaux ont été proposés dans la littérature pour la décomposition des processus à différentes fins, dont l'obfuscation [DPdSS12]. Mais en général, cette décomposition se fait à travers un ensemble de contraintes de *séparation* et de *co-localisation* qui préconisent de déployer certaines tâches sur différents clouds (séparation), ou au contraire de les localiser sur le même cloud (co-localisation) selon la nature de ces tâches et de l'importance des informations qu'elles comprennent. Certaines de ces contraintes peuvent également préconiser de déployer la tâche en local si son déploiement sur un environnement cloud est trop risqué. Par ailleurs, d'autres contraintes sont considérées comme étant plus fortes que d'autres, et peuvent être classées par priorité selon la sensibilité des tâches concernées et les besoins de l'organisation.

Ainsi, le principe de notre approche est d'ajouter ou supprimer itérativement des contraintes à l'ensemble des contraintes appliquées lors du déploiement du processus, en veillant à ne pas dépasser les seuils de risque ou de coût (selon l'approche appliquée). En effet, plus on applique de contraintes de séparation, plus le processus est fragmenté sur plusieurs clouds résultant en un déploiement plus sûr, mais aussi plus coûteux. Respectivement, plus on supprime de contraintes de séparation, moins le processus sera fragmenté, résultant en un déploiement moins coûteux, mais aussi moins sûr.

L'ajout des contraintes se fait en commençant par les contraintes les plus contraignantes (plus prioritaires), contrairement à la suppression qui commence par les moins contraignantes (moins influentes). A chaque itération, nous calculons le risque et le coût induit par notre déploiement en fonction des contraintes appliquées. L'itération suivante ne sera déclenchée que si l'itération en cours génère une configuration respectant le seuil défini.

Nous présentons dans ce qui suit nos heuristiques permettant de minimiser le risque au détriment du coût et vice versa.

5.3 Heuristiques de réduction du risque et du coût

Cette section présente deux heuristiques ; une première qui minimise le risque de déploiement en respectant un seuil maximum de coût et une seconde qui minimise le coût de déploiement en assurant un seuil de risque maximum à ne pas dépasser. Nous introduisons ces heuristiques à travers un ensemble d'algorithmes qui expliquent notre méthodologie de déploiement.

Avant d'entrer plus en détail dans le fonctionnement de ces algorithmes, nous présen-

Paramètres	
P	Le processus utilisé
C	L'ensemble des clouds disponibles
T_c	le seuil de coût maximum
T_r	Le seuil de risque maximum
S	L'ensemble de contraintes appliquées
C_{dep}	L'ensemble des clouds qui déploient des tâches du processus
Config	Configuration optimale de l'itération en cours
FinalConfig	Configuration optimale finale
SensitiveTasks	L'ensemble des tâches sensibles du processus P

TABLE 5.1 – Paramètres utilisés

tons dans le tableau 5.1 l'ensemble des paramètres utilisés dans les différents algorithmes proposés au cours de ce chapitre.

5.3.1 Algorithmes pour la minimisation du risque avec seuil de coût

Nous introduisons dans ce qui suit deux algorithmes complémentaires. L'algorithme 1 permet de sélectionner la configuration finale optimale. Il s'appuie sur l'algorithme 2 qui permet de récupérer la configuration optimale courante en fonction de l'itération en cours.

Étant donné que notre objectif est la réduction du risque, l'ensemble S sera itérativement enrichis par des contraintes S_i ordonnées par priorité ($i \in [1, N]$) où S_1 représente la contrainte la plus contraignante (qui affecte le plus le niveau de risque). L'algorithme s'arrête dans deux cas : lorsque toutes les contraintes ont été ajoutées ou lorsqu'une configuration ne respectant pas le seuil de coût a été retournée.

L'algorithme commence par sélectionner la configuration la moins coûteuse en n'appliquant pas de contraintes (ligne 3). Si cette dernière ne respecte tout de même pas le seuil de coût, cela veut dire qu'il doit impérativement être augmenté (auquel cas aucune configuration ne pourra être sélectionnée comme étant la plus optimale).

Si au contraire la configuration obtenue respecte le seuil de coût (ligne 8), l'algorithme ajoute itérativement une contrainte S_i à l'ensemble des contraintes S, permettant ainsi d'obtenir une configuration qui réduit encore plus le risque avec cependant un coût de déploiement plus élevé. L'algorithme continue d'ajouter des contraintes par ordre de priorité (de la plus à la moins importante) jusqu'à l'obtention d'une configuration ne respectant pas notre seuil de coût (ligne 18). Dans ce cas, l'algorithme s'arrête et retourne la dernière configuration respectant le seuil fixé.

Par ailleurs, comme introduit précédemment, l'algorithme 2 permet de sélectionner la configuration optimale de l'itération en cours. Pour ce faire, il se base sur la logique suivante :

- Déployer en premier lieu les tâches sensibles sur les clouds ayant les meilleurs réputations (ligne 3).
- Minimiser le nombre de clouds de la collaboration en favorisant ceux qui déploient déjà des tâches du processus en utilisant l'algorithme 3 (ligne 7).

Algorithm 1 Deployment with Risk minimizing while varying constraints

Require: P, C, T_c, S
Ensure: Optimal $FinalConfig$

```
1:  stop ← false
2:  S ← ∅
3:  Config = ConfigurationWithRiskMin(P, C, S)
4:  if (config ≠ ∅) then
5:      Cost ← config.CostComputation()
6:      if (Cost > T_c) then
7:          Threshold too low it must be increased
8:      else
9:          i ← 1
10:         while (i⩽ N && !stop ) do
11:             S ← S + S_i                          ▷ Adding a constraint
12:             Config =ConfigurationWithRiskMin(P, C, S)
13:             if (config ≠ ∅) then
14:                 Cost ← config.CostComputation()
15:                 if (Cost < T_c) then
16:                     FinalConfig ← config
17:                     i ← i + 1
18:                 else                  ▷ The cost threshold is overpassed
19:
20:                     stop ← true
21:         EndWhile
22: return FinalConfig
```

- Si le nombre de clouds est insuffisant pour déployer toutes les tâches du processus en respectant l'ensemble des contraintes S, alors l'algorithme s'arrête ; aucune configuration n'est possible avec les paramètres entrés (ligne 10).

- Si au contraire toutes les tâches sensibles sont déployées, l'algorithme essaye de déployer le reste des tâches (non contraintes), en tentant d'abord un déploiement sur des clouds qui déploient déjà des tâches du processus, tout en respectant les contraintes de séparation S. Ceci a pour objectif de réduire le coût (ligne 15). Dans le cas contraire, l'algorithme sélectionne le cloud le moins cher (si des clouds restent disponibles). Auquel cas l'algorithme s'arrête, i.e. aucune configuration n'est possible (ligne 20).

- Dans le cas où toutes les tâches sont déployées (ligne 23), la configuration finale est définie (ligne 24).

L'objectif de l'algorithme 3 est de vérifier si une tâche t_i peut être déployée sur un cloud c_i en respectant un ensemble de contraintes S. En effet, si le cloud c_i déploie une tâche t_j et qu'une contrainte de séparation $separate(t_i, t_j)$ appartient à l'ensemble S, la tâche t_i ne peut être déployée sur c_i.

Algorithm 2 ConfigurationWithRiskMin

Require: P, C, S, $SensitiveTasks$
Ensure: $Config$
1: **for** (each t ∈ SensitiveTasks) **do**
2: **if** ($C_{dep} = \emptyset$) **then** ▷ No task has been yet deployed
3: *Select cloud c_i with best reputation ()*
4: $c_i.add(t)$
5: $C_{dep}.add(c_i)$
6: **else** ▷ Trying on clouds c_i already deploying a task
7: **if** ($CanBeDeployedOn(t, c_i, S)$) **then**
8: $c_i.add(t)$
9: **else** ▷ deploying on available clouds if available
10: **if** ($C = \emptyset$) **then** ▷ No more cloud
11: $config \leftarrow \emptyset$
12: **if** (sensitive Tasks Deployed) **then**
▷ Deploying the no sensitive tasks
13: **for** (each t ∈ no sensitive tasks) **do**
14: *Trying on clouds c_i already deploying tasks of p*
15: **if** ($CanBeDeployedOn(t, c_i, S)$ **then**
16: $c_i.add(t)$
17: **else** ▷ Deploying on a new cloud if available
18: **if** ($C \neq \emptyset$) **then**
19: Selecting the cheapest available cloud c_i in C
20: **else**
21: $config \leftarrow \emptyset$
22: **if** (all Tasks Deployed **then**)
23: $config \leftarrow C_{dep}$
24: **return** $config$

Algorithm 3 CanBeDeployedOn

Require: t, c_i, S
Ensure: Dep
1: $Dep \leftarrow true$
2: **for** (each $t_j \in c_i$) **do**
3: **while** (Dep) **do**
4: **if** ($Separate(t, t_j) \in S$) **then**
5: $Dep \leftarrow false$
6: **return** Dep

5.3.2 Algorithmes pour la minimisation du coût avec seuil de risque

Nous introduisons dans cette section notre seconde heuristique dont l'objectif est de réduire le coût de déploiement d'un processus métier dans le cloud, en assurant toutefois que le risque auquel il est exposé ne dépasse pas un certain seuil T_r. Pour ce faire, nous proposons un algorithme ayant pour objectif de sélectionner la configuration la plus optimale qui réponde aux besoins définis.

Vu que l'objectif est la réduction du coût, les différentes contraintes seront itérative-

ment supprimées par priorité descendante (les moins contraignantes d'abord) : S_N est la contraintes la moins prioritaire tandis que S_1 est la plus prioritaire.

Dans ce contexte, l'algorithme 4 permet de sélectionner la configuration finale optimale parmi les différentes itérations possibles (qui diffèrent selon l'ensemble des contraintes appliquées) ainsi que du seuil de risque. Il s'appuie également sur un autre algorithme (algorithme 5) afin de sélectionner la configuration optimale courante en fonction de l'itération en cours.

Algorithm 4 Deployment with Cost minimizing while varying constraints

Require: P, C, T_r, S
Ensure: Optimal $FinalConfig$
1: $S \leftarrow$ All constraints
2: $stop \leftarrow false$
3: $Config =$ **ConfigurationWithCostMin**(P, C, S)
4: **if** $(config \neq \emptyset)$ **then**
5: $Risk \leftarrow config.RiskComputation()$
6: **if** $(Risk > T_r)$ **then**
7: Threshold too low it must be increased
8: **else**
9: $i \leftarrow N$
10: **while** (i⩾1 && !$stop$) **do**
11: $S \leftarrow S - S_i$
12: $Config =$**ConfigurationWithCostMin**(P, C, S)
13: **if** $(config \neq \emptyset)$ **then**
14: $Risk \leftarrow config.RiskComputation()$
15: **if** $(Risk < T_r)$ **then**
16: $FinalConfig \leftarrow config$
17: $i \leftarrow i - 1$
18: **else** ▷ The risk threshold is overpassed
19: $stop \leftarrow true$
20: **EndWhile**
21: **return** $FinalConfig$

Contrairement à l'algorithme 1 qui n'applique aucune contrainte à la première itération, l'algorithme 4 les applique toutes dés le départ afin d'obtenir la configuration la moins risquée. Si cette configuration ne respecte cependant pas le seuil de risque, cela veut dire qu'il est impossible de trouver une configuration avec les contraintes en cours. Ainsi, le concepteur sera dans l'obligation d'augmenter le seuil de risque toléré.

Dans le cas contraire, i.e. la configuration obtenue respecte le risque, l'algorithme supprime itérativement les contraintes par ordre de priorité descendant de l'ensemble S (ligne 11). L'algorithme continue à supprimer des contraintes tant qu'on obtient des configurations respectant le seuil. Il se termine si toutes les contraintes ont été supprimées ou si une configuration qui ne respecte pas le seuil a été retournée (ligne 18). Dans le premier cas, l'algorithme sélectionne la dernière configuration obtenue tandis que dans le second, il retourne la dernière qui respecte le seuil de risque toléré. Comme l'algorithme 2, l'algorithme 5 sélectionne la meilleure configuration de l'itération en cours en minimisant le coût. Pour ce faire :

- L'algorithme sélectionne le cloud le moins cher pour le déploiement du processus.

Algorithm 5 ConfigurationWithCostMin

Require: P, C, S
Ensure: $Config$
1: **for** (each t ∈ P) **do**
2: **if** ($C_{dep} \neq \emptyset$) **then**
3: *Select the cheapest cloud* c_i *()*
4: $c_i.add(t)$
5: $C_{dep}.add(c_i)$
6: **else**
▷ Trying on a clouds already deploying tasks of p
7: **for** (each $c_i \in C_{dep}$) **do**
8: **if** ($CanBeDeployedOn(t, c_i, S)$) **then**
9: $c_i.add(t)$
10: **else**
▷ Deploying on a new cloud c_j if available
11: **if** ($C \neq \emptyset$) **then**
12: *Select the cheapest cloud* c_j
13: $c_j.add(t)$
14: $C_{dep}.add(c_j)$
15: **else** ▷ no more available cloud
16: $config \leftarrow \emptyset$
17: **if** (All tasks are deployed) **then**
18: $config \leftarrow C_{dep}$
19: **return** $config$

- Si au moins un cloud déploie déjà des tâches du processus, l'algorithme tente de déployer les nouvelles tâches dessus tant que les contraintes S le permettent (ligne 8). Ceci est fait pour d'une part privilégier l'utilisation des clouds les moins chers, ainsi que pour diminuer le taux et par conséquent le coût de communication entre les clouds de la collaboration.

- Si les contraintes de séparation empêchent le déploiement sur ces clouds, l'algorithme sélectionne le prochain cloud considéré comme étant le « moins cher » parmi l'ensemble des clouds disponibles (ligne 10). Si cet ensemble ne comprend plus de clouds, l'algorithme s'arrête ; aucune configuration n'est possible (ligne 16).

- Si au contraire l'algorithme arrive à déployer toutes les tâches du processus, la configuration finale est retournée pour l'itération en cours (ligne 18).

L'avantage de ces heuristiques est qu'elles peuvent être utilisées avec différents modèles de risques [DAGM16] [GDG+14] et de coût [MWT12] [ARAEBA13] (selon les besoins du client). Par ailleurs, pour s'assurer de l'exactitude de leur résultats et valider leur efficacité, nous proposons dans la section suivante deux modèles exacts (PLNE [24]) assurant un déploiement optimal en tenant compte des contraintes induites par le niveau de risque et le coût correspondant. Nous nous baserons par la suite sur ces modèles pour vérifier l'exactitude des résultats apportés par nos heuristiques.

24. programmation linéaire en nombres entiers : Modèles qui considèrent des problèmes d'optimisation décrits par une fonction, des contraintes linéaires et des variables entières

5.4 Modèles de réduction du risque et du coût

Cette section présente et formalise une solution d'optimisation exacte pour chacune de nos heuristiques à travers l'introduction de modèles pour le déploiement des processus métiers dans le cloud. Ces modèles seront par la suite utilisés comme un outil afin de vérifier la précision et l'exactitude des résultats induis par nos heuristiques.

Avant d'entrer dans plus de détails, il est important de rappeler que l'utilisation d'une solution exacte pour obtenir la meilleure configuration pourrait devenir lent et coûteux pour les processus de centaines ou même de milliers d'activités. C'est pourquoi nous avons défini des heuristiques dans notre approche.

L'ensemble des notations et variables nécessaires à la bonne compréhension de nos modèles sont représentés dans le tableau 5.2.

Notations	
$\mathcal{I}$	L'ensemble des tâches du processus
$\mathcal{J}$	L'ensemble des clouds
Rep_j	Réputation du cloud j
$Cost_j$	Coût du cloud j
T_c	Le seuil de coût
T_r	Le seuil de risque
$\mathcal{S}$	L'ensemble des tâches devant être séparées
${x_j}^i$	Variable binaire de décision : égale à 1 si la tâche i est assignée au cloud j, 0 sinon
R	Niveau de réputation assurant un niveau de risque accepté par l'organisation

Table 5.2 – Notations utilisées

5.4.1 Modèle de déploiement avec réduction du risque

Nous visons à travers ce modèle à minimiser le risque auquel est exposée la collaboration en réduisant la probabilité de coalition de clouds. Par ailleurs, comme ce risque est directement lié au *comportement des fournisseurs de cloud* impliqués dans la collaboration, nous sélectionnons ces clouds en maximisant leur niveaux de réputation, réduisant ainsi la probabilité d'une éventuelle coalition.

Notre modèle de risque est défini comme suit :

$$\max \sum_{i}^{i\in\mathcal{I}} \sum_{j}^{j\in\mathcal{J}} Rep_j {x_j}^i \tag{5.1}$$

tels que :

$$\forall i \in \mathcal{I}, \sum_{j}^{j\in\mathcal{J}} {x_j}^i = 1 \tag{5.2}$$

$$\sum_{i}^{i\in\mathcal{I}} \sum_{j}^{j\in\mathcal{J}} Cost_j {x_j}^i < T_c \tag{5.3}$$

$$\forall j \in \mathcal{J}, \forall < i_1, i_2 > \in \mathcal{S}, {x_j}^{i_1} {x_j}^{i_2} = 0 \tag{5.4}$$

$$\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, {x_i}^j \in \{0, 1\} \tag{5.5}$$

- La fonction objective est représentée par l'équation 5.1 qui représente la sélection des clouds ayant les meilleures réputation.
- Les équations 5.2 et 5.3 stipulent respectivement l'assignation d'une tâche à un seul cloud, et le respect d'un seuil de coût de déploiement bien défini.
- L'équation 5.4 représente les tâches devant être séparées (selon l'ensemble des contraintes de séparation S).

Cependant, puisque cette séparation de tâches est représentée par une multiplication (équation 5.4), nous nous sommes basés sur les principe introduits dans [RYC17] pour ce problème d'optimisation quadratique. Par ailleurs, puisque le temps de résolution peut devenir très lent, nous avons linéarisé ce problème comme suit :

$$\begin{gathered}
\forall < i_1, i_2 > \in \mathcal{S} \sum_{d=1}^{|\mathcal{S}|} \sum_{j}^{j \in \mathcal{J}} {x_j}^{i_1} + {x_j}^{i_2} - 2{w_j}^d \leq 1 \\
\forall i_1 \in \mathcal{S} \sum_{d=1}^{|\mathcal{S}|} \sum_{j}^{j \in \mathcal{J}} {w_j}^d \leq {x_j}^{i_1} \\
\forall i_2 \in \mathcal{S} \sum_{d=1}^{|\mathcal{S}|} \sum_{j}^{j \in \mathcal{J}} {w_j}^d \leq {x_j}^{i_2} \\
\sum_{d=1}^{|\mathcal{S}|} \sum_{j}^{j \in \mathcal{J}} {w_j}^d = 0 \\
\forall j \in \mathcal{J}, \forall d \in \mathcal{S}, {w_j}^d \in \{0, 1\}
\end{gathered} \tag{5.6}$$

5.4.2 Modèle de déploiement avec réduction du coût

Nous présentons dans ce qui suit notre modèle de déploiement pour la réduction des coûts engendrés. Ce modèle sélectionne la configuration qui minimise le coût en respectant un certain seuil de risque toléré par l'organisation. Ce niveau de risque est principalement basé sur un seuil de réputations de clouds. En effet, nous considérons que plus ces clouds ont une bonne réputation, moins le processus est exposé au risque de coalitions de clouds (voir chapitre 4).

Notre modèle de déploiement est définie comme suit :

$$\min \sum_{i}^{i \in \mathcal{I}} \sum_{j}^{j \in \mathcal{J}} Cost_j {x_j}^i \tag{5.7}$$

tels que :

$$\forall i \in \mathcal{I}, \sum_{j}^{j \in \mathcal{J}} {x_j}^i = 1 \tag{5.8}$$

$$\sum_{i}^{i\in\mathcal{I}}\sum_{j}^{j\in\mathcal{J}} Rep_j {x_j}^i > R \tag{5.9}$$

$$\forall j \in \mathcal{J}, \forall < i_1, i_2 > \in \mathcal{S}, {x_j}^{i_1} {x_j}^{i_2} = 0 \tag{5.10}$$

$$\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, {x_i}^j \in \{0, 1\} \tag{5.11}$$

- La fonction objective (équation 5.7) représente le fait de sélectionner les clouds en minimisant le coût de déploiement.
- La contrainte de l'équation 5.9 assure d'avoir un minimum de réputation respectanr le niveau R. L'objectif étant de limiter le niveau de risque auquel est exposé le processus.

Les équations restantes sont les mêmes que celles définies pour le modèle de risque. Il est cependant à noter que l'équation 5.10 a aussi été linéarisée. Les résultats sont les mêmes que ceux précédemment définis (voir equation 5.6).

5.5 Application

Nous présentons dans cette section les différents paramètres utilisés en vue d'appliquer et de tester l'efficacité des heuristiques introduites dans la section 5.3. Pour ce faire, nous présentons les modèles de risque et de coût utilisés dans nos différentes expérimentations, ainsi que les différentes contraintes appliquées au fil des itérations des algorithmes 1 et 4.

5.5.1 Modèles de risque et de coût

Comme introduit précédemment, nos heuristiques peuvent s'appliquer avec différents modèles de risque et de coût. Durant nos expérimentations, nous nous sommes basés sur le modèle de risque introduit dans le chapitre 4, qui tient compte de la réputation des clouds qui déploient les tâches sensibles, ainsi que du nombre de clouds qui séparent les tâches complémentaires (voir la section 4.4 du chapitre 4 pour plus de détail).

Pour rappel, le niveau de risque d'une tâche sensible sur un chemin donné ainsi que le niveau de risque d'une collaboration son calculés comme suit :

$$\mathbf{Risque}(\mathbf{t_i}) = \sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{Risque}(\mathbf{t_i}, \mathbf{p_j}) \tag{5.12}$$

où

- $\mathbf{m}$: nombre de chemins séparant t_i de sa tâche complémentaire $t_{i'}$.
- $\mathbf{Risque}(\mathbf{t_i}, \mathbf{j})$: risque de la tâche t_i en sélectionnant le chemin j.

$$\mathbf{Risque(collaboration)} = \sum_{\mathbf{k=1}}^{\mathbf{p}} \mathbf{Risque(t_k)} \tag{5.13}$$

où

- **p** : nombre de tâches sensibles.
- **Risque($\mathbf{t_k}$)** : niveau de risque de la tâche t_k.

Concernant le modèle de coût, nous nous sommes basés sur celui introduit dans [GDG+14] qui tient compte des paramètres suivants pour le calcul du coût d'une configuration donnée :

i) Coût d'exécution : Représente la puissance du processeur et le nombre de ressources nécessaires pour l'exécution des tâches du processus. Ce coût est calculé en déterminant le coût par unité de temps généré par les ressources pour l'exécution du processus métier.

ii) Coût de transfert : Représente le coût de transfert des données de et vers les clouds appartenant à la collaboration. Ces coûts de transfert varient selon la distance qui sépare les clouds, i.e. transférer des données vers un cloud se situant dans la même région est moins cher qu'un transfert en dehors de cette région. Ce coût est exprimé en Dollars par GigaByte.

iii) Coût de stockage : Représente le coût de la mémoire nécessaire aux données du processus pour leur maintenance, leur gestion, et leur sauvegarde (back up). Ce coût est exprimé en Dollars par Giga Byte.

Le coût global d'une collaboration est défini par la somme des coûts de déploiement de toutes les tâches du processus comme suit :

$$Cost = \sum_{i=1}^{n} \{EC(c_{ti}) + SC(c_{ti}) + TC_{ti}\} \tag{5.14}$$

$$TC_{ti} = \sum_{j=1}^{m} TC(c_{tj}, c_{ti}) + \sum_{k=1}^{v} TC(c_{ti}, c_{tk}) \tag{5.15}$$

où

- n : nombre de tâches du processus.
- $EC(c_{t_i})$: coût d'exécution de la tâche t_i sur le cloud c.
- $SC(c_{t_i})$: coût de stockage des données de la tâche t_i sur le cloud c.
- TC_{t_i} : coût de transfert de et vers c.
- m l'ensemble des prédécesseurs de i.
- v l'ensemble des successeurs de i.

La table 5.3 représente un exemple d'offres de tarification sur lesquelles nous nous sommes basés au cours de nos expérimentations.

Offres Cloud	Coût d'exécution ($/GHz/mo)	Coût de stockage ($/GB/mo)	Coût de transfert ($/GB)
Softlayer [a]	$20	$0.10	$0.10
CloudSigma AG [b]	$13.86	$0.18	$0.06
FireHost [c]	$25.70	$2.78	$0.50
SHI International, Corp. [d]	$11.56	$0.29	$0.01
Terremark [e]	$3.60	$0.25	$0.17

a. http ://www.softlayer.com/fr
b. https ://www.cloudsigma.com
c. https ://www.firehost.com
d. https ://www.shi.com
e. http ://www.terremark.com

TABLE 5.3 – Offres de Cloud

5.5.2 Contraintes appliquées

Nous utilisons au cours de notre approche un ensemble de contraintes ayant pour objectif de séparer/co-localiser certaines tâches, selon les besoins de l'utilisateur. Par ailleurs, pour tester l'efficacité de nos approches, nous avons utilisé les contraintes suivantes :

- **Contraintes type 1** : Séparer les *décisions* et les *synthèses* de leur tâches complémentaires (voir section 3.3.2.2 pour plus de détails).
- **Contraintes type 2** : Séparer les fragments alternatifs (3.3.2.2).
- **Contraintes type 3** : Si le processus comprend des faux fragments, les séparer sur différents clouds (section 3.4.1.2).

Étant donné que les fragments *décisions* et *synthèses* manipulent d'importantes informations, les contraintes de type 1 évitent de condenser trop d'informations sur le même cloud en séparant ces fragments. De plus, vu que les décisions peuvent être extraites des fragments alternatifs, les contraintes de type 2 permettent de les séparer.

Le dernier type de contrainte permet d'augmenter la longueur et le nombre de chemins qui séparent les tâches sensibles (voir la section 3.4 du chapitre 3 pour plus de détails).

Nous considérons que le premier type est plus prioritaire et plus efficace pour la réduction du risque que le type 2, qui lui même est plus efficace que le type 3.

Par ailleurs, comme expliqué dans la section 5.3, nos algorithmes de réduction de risque et de coût (algorithmes 1 et 4) se basent sur l'ajout/ suppression de contraintes au fil des différentes itérations (selon l'algorithme appliqué). Il est à noter que nous nous sommes basés sur le tableau 5.4 lors de l'application de nos algorithmes.

5.6 Expérimentations

Nous présentons dans cette section les différentes expérimentations réalisées en vue de tester l'efficacité des heuristiques introduites dans la section 5.3.

Nous commençons par introduire ci-dessous l'environnement de développement. Ensuite, nous présentons notre première expérimentation ayant pour objectif d'analyser le comportement et les résultats retournés par nos heuristiques.

Enfin, puisque les heuristiques retournent des résultats approximatifs, nous avons mené une seconde expérimentation qui se base sur les modèles introduit dans la section [**?**], afin de valider leurs résultats. En effet, cette expérimentation permet d'une part de comparer le comportement des heuristiques avec celui des méthodes exactes, i.e. où à quel point nos solutions se rapprochent de la solution optimale. D'autre part, elle permet de vérifier également leur précision à travers le calcul de leur taux d'erreur.

5.6.1 Environnement de développement

Les différentes approches proposées furent développées en Java en raison de sa portabilité et de sa richesse. Par ailleurs, ce choix est également motivé par le fait que nos modèles PLNE furent implémentés en utilisant le solver *IBM ILOG CPLEX Optimization 12.7.1* qui s'adapte bien avec le langage java.

Nous avons également utilisé dans nos expérimentations un processus métier modélisé en BPMN composé de 200 activités, qui furent générées en se basant sur le principe développé dans [NGY+16].

De plus, nous avons considéré les hypothèses suivantes lors de nos expérimentations :

- *Réputation des clouds* : puisque nous n'étions pas en mesure de nous baser sur de réels données, nous avons généré aléatoirement un ensemble de réputations de clouds.
- *Coût des clouds* : Nous nous sommes basés sur le tableau 5.3 au cours de nos expérimentations. Cependant, puisque nous n'étions pas en mesure de définir au préalable le nombre de clouds nécessaires pour le déploiement du processus, nous avons généré un ensemble d'offres aléatoirement à partir d'une plage de valeurs similaires à celles du tableau 5.3.
- *Contraintes de séparation* : Nous ajoutons/supprimons les contraintes comme introduit dans le tableau 5.4.
- Nous considérons qu'il n'existe aucune corrélation entre les coûts et la réputation des clouds, i.e. on peut trouver un cloud non coûteux avec une bonne réputation.

5.6.2 Évaluation des heuristiques

La première expérimentation permet d'évaluer nos approches d'optimisation (algorithmes 1 et 4) à travers un ensemble d'itérations (tableau 5.4) en appliquant les modèles de risque et de coût introduit dans la section 5.5. Les résultats obtenus sont illustrés dans la figure 5.3.

Plus précisément, la figure 5.3a représente les résultats obtenus pour la réduction du risque (algorithme 1). Comme illustré, notre méthode permet de minimiser le risque auquel est exposée la collaboration lors d'un déploiement cloud, en augmentant cependant son coût. Ainsi, en fonction des ressources financières sur lesquelles le seuil est fixé, notre approche retourne la dernière configuration qui le respecte ($(risque, coût) = (0.00069, 6990)$ dans ce cas).

Numéro d'itération	***Minimisation du risque (avec seuil de coût)***	***Minimisation du coût (avec seuil de risque)***
Itération 1	***Paramètres :*** -Aucune contrainte. -Considérée comme borne inférieure pour le seuil de coût. ***Résultats :*** - Augmenter le seuil si aucune configuration n'est sélectionnée.	***Paramètres :*** -Appliquer toutes les contraintes. -Considérée comme borne inférieure pour le seuil de risque. ***Résultats :*** -Augmenter le seuil si aucune configuration n'est sélectionnée.
Itération 2	***Paramètres :*** -Appliquer les contraintes de type 1. ***Résultats :*** - Si la configuration obtenue respecte le seuil de coût, nous lançons la prochaine itération pour obtenir un meilleur compromis (configurations moins risquée et plus chère, mais qui respecte le seuil de coût accepté).	***Paramètres :*** - Supprimer les contraintes de type 3. Nous commençons par ce type car il est considéré comme étant moins contraignant que les types 1 et 2. ***Résultats :*** -lancer la prochaine itération si la configuration obtenue respecte toujours le seuil.
Itération 3	***Paramètres :*** -Ajouter la contrainte de type 2. ***Résultats :*** - Si la configuration obtenue continue à respecter le seuil t_c, nous lançons la dernière itération.	***Paramètres :*** - Supprimer la contrainte de type 2. ***Résultats :*** -La dernière itération n'est lancée que si la configuration obtenue respecte le seuil t_r.
Itération 4	***Paramètres :*** -Ajout de la contrainte restante (de type 3). ***Résultats :*** - Génération de la configuration la moins risquée mais également la plus coûteuse.	***Paramètres :*** - Supprimer la contrainte restante. ***Résultats :*** -Obtention de la configuration la moins coûteuse, ayant cependant un niveau de risque qui ne respecte peut être pas le seuil t_r.

TABLE 5.4 – Contraintes appliquées

La figure 5.3b représente les résultats obtenus pour la réduction du coût (algorithme 4). Comme illustré, la première configuration obtenue offre un niveau de risque réduit, à un coût cependant important. Ce coût est par conséquent réduit à travers les différentes itérations (grâce à la suppression des contraintes du tableau 5.4) jusqu'à l'obtention d'une configuration dont le niveau de risque ne respecte pas le seuil autorisé. Ce niveau de risque est considéré comme étant trop élevé pour un déploiement cloud. Par conséquent, notre approche sélectionne la configuration $(risque, coût) = (0.0091, 1924)$.

Même si les résultats obtenus sont satisfaisants, une marge d'erreur subsiste. Ainsi, il est nécessaire de s'assurer que nos algorithmes retournent des solutions aussi proches que possible de l'optimale. Pour ce faire, nous avons mené un second ensemble d'expériences ayant pour objectif de comparer d'une part le comportement des heuristiques vis à vis des méthodes exactes, et d'autre part de calculer leur taux d'erreur afin de vérifier la précision des résultats obtenus.

5.6.3 Comparaison des résultats avec les solutions exactes

Nous avons mené un second ensemble d'expérimentations ayant pour objectif de s'assurer de la précision des résultats obtenus. Pour ce faire, nous nous sommes basés sur les modèles PLNE (qui furent implémentés en utilisant le solver *IBM ILOG CPLEX Optimization 12.7.1*). Les résultats obtenus sont illustrés dans les figures 5.4, 5.5 et 5.6.

La figure 5.4a représente les résultats obtenus pour la réduction du risque en appliquant les deux méthodes (heuristique et exacte). Comme illustré, notre méthode retourne des résultats qui se rapproche fortement de ceux de la solution exacte. De plus, nous remarquons que la solution exacte ne retourne aucune solution dans la dernière itération, tandis que notre approche retourne une solution qui dépasse de très peu le seuil de coût autorisé, mais qui réduit considérablement le risque. Par conséquent, la dernière itération de l'heuristique peut éventuellement aider le concepteur à mieux fixer le seuil de coût, contrairement à la méthode exacte qui ne retourne pas de résultats durant cette itération.

La figure 5.4b représente les résultats obtenus pour la réduction du coût. Comme illustré, l'heuristique et la méthode exacte en plus de retourner des résultats qui se rapprochent fortement, ont un comportement similaire, i.e. les deux méthodes ne respectent pas le seuil de risque dans la dernière itération.

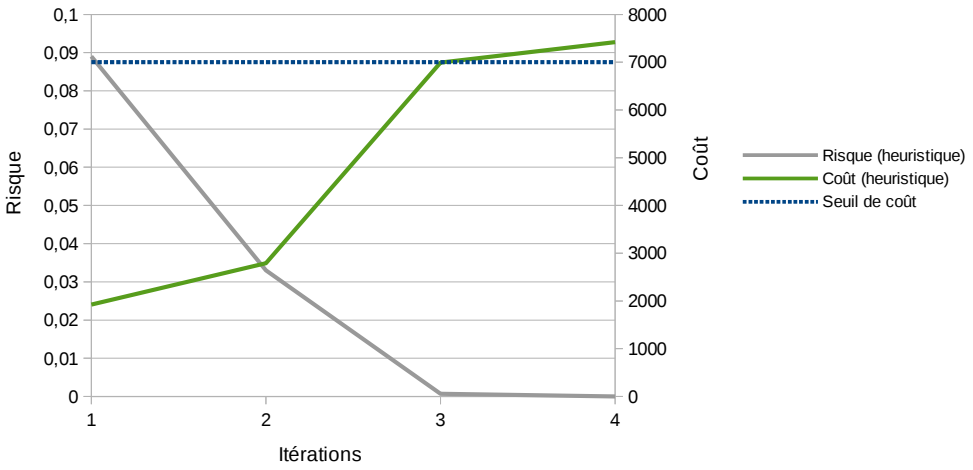
Comme prévu, nous soulignons une corrélation entre *le niveau de risque, le nombre de clouds et le coût*, i.e. moins une collaboration est risquée, plus il y a de clouds impliqués et donc plus elle est coûteuse (voir les figures 5.5a et 5.5).

De plus, nous remarquons que les heuristiques ont un temps de calcul plus rapide que celui des méthodes exactes (voir figure.5.6a et 5.6b).

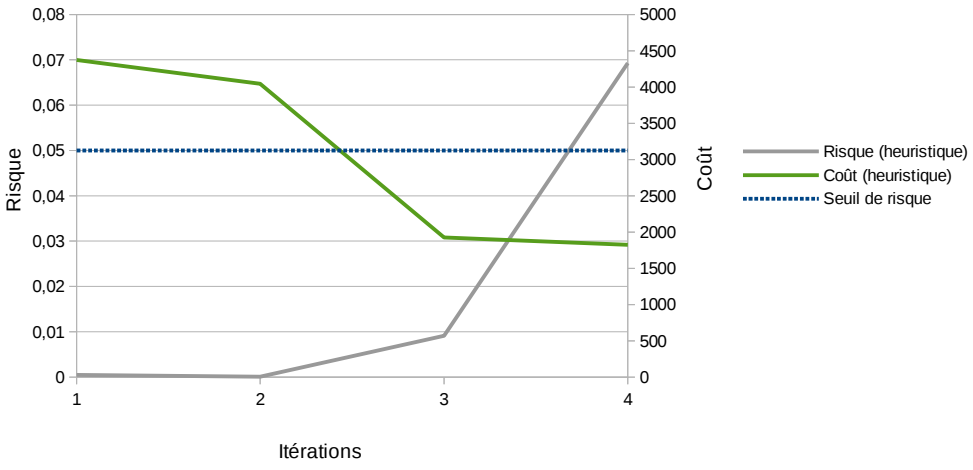
Par ailleurs, pour s'assurer de la précision de nos résultats, nous avons mené un dernier ensemble d'expérimentations dont l'objectif est de calculer le taux d'erreur des heuristiques en se basant sur l'erreur relative

$$x' = \frac{X_{heuristique} - X_{exacte}}{X_{exacte}} \tag{5.16}$$

- x' : le taux d'erreur.
- X_{exact} : la valeur retournée par la solution exacte.

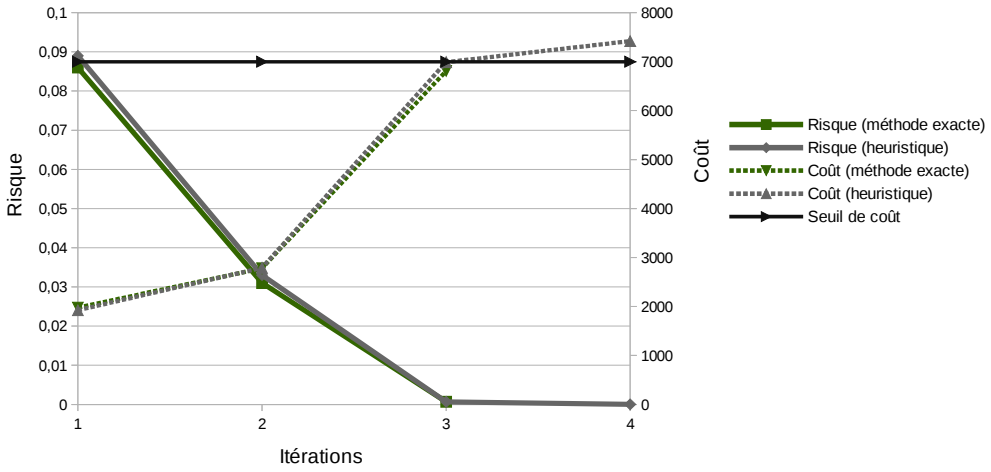


(a) Réduction du risque

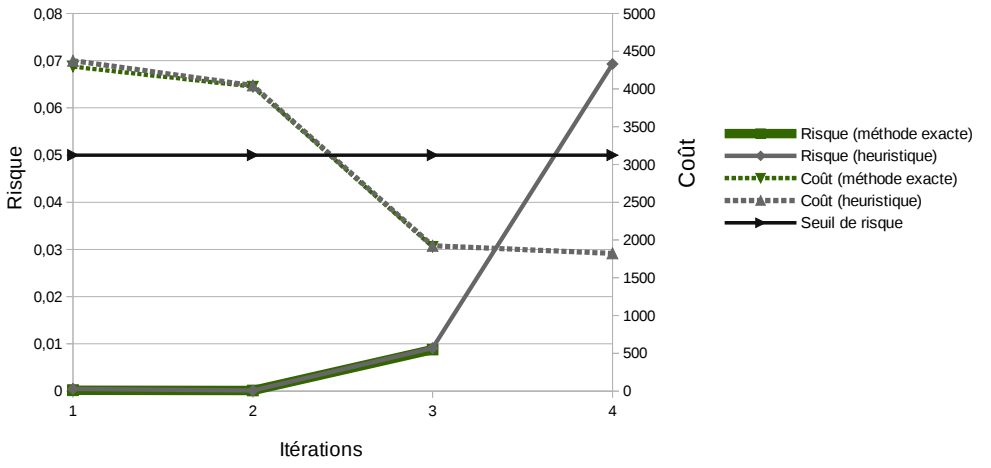


(b) Réduction du coût

FIGURE 5.3 – Résultats des heuristiques

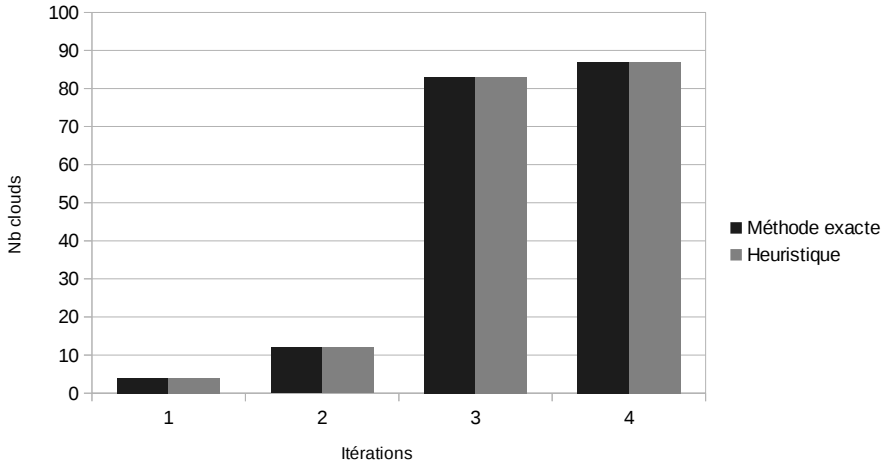


(a) Réduction du risque par les méthodes exacte et heuristique

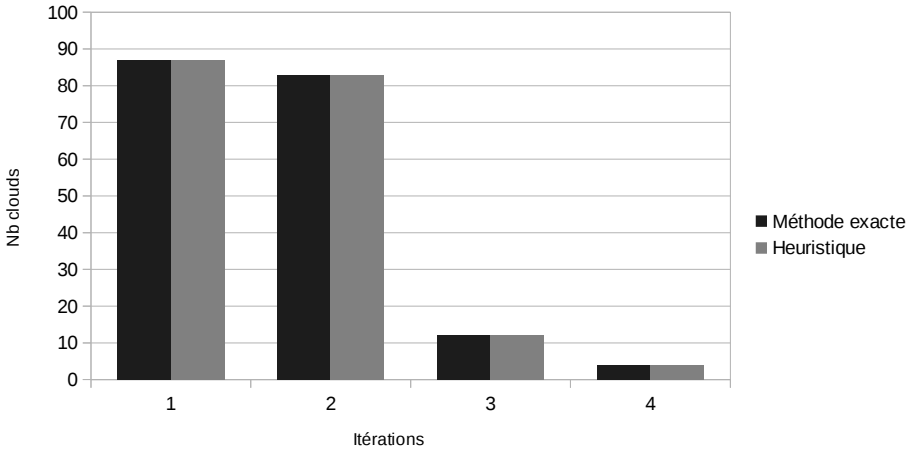


(b) Réduction du coût par les méthodes exacte et heuristique

FIGURE 5.4 – Réduction du risque et du coût par les méthodes heuristiques et exactes

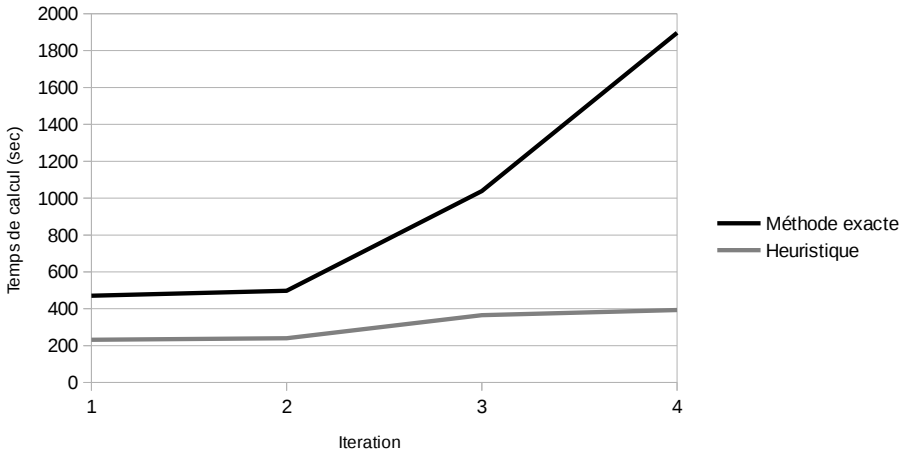


(a) Nombre de clouds impliqués pour la réduction du risque (exacte et heuristique)

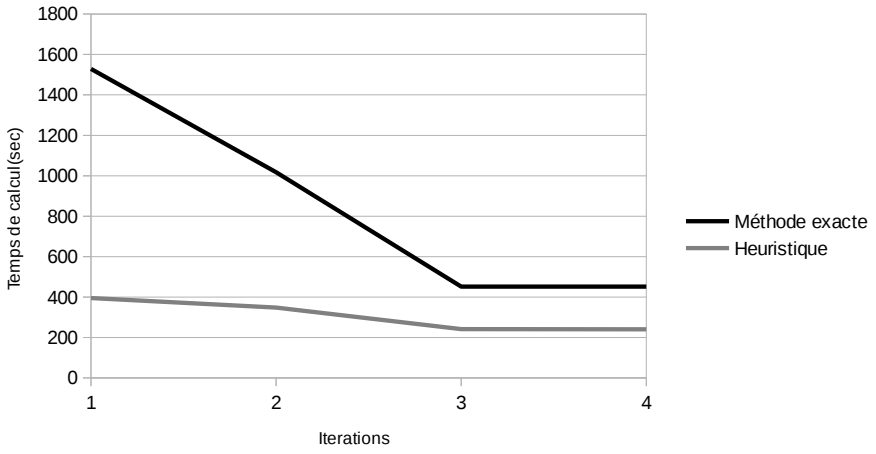


(b) Nombre de clouds impliqués pour la réduction du coût (exacte et heuristique)

FIGURE 5.5 – Nombre de clouds impliqués

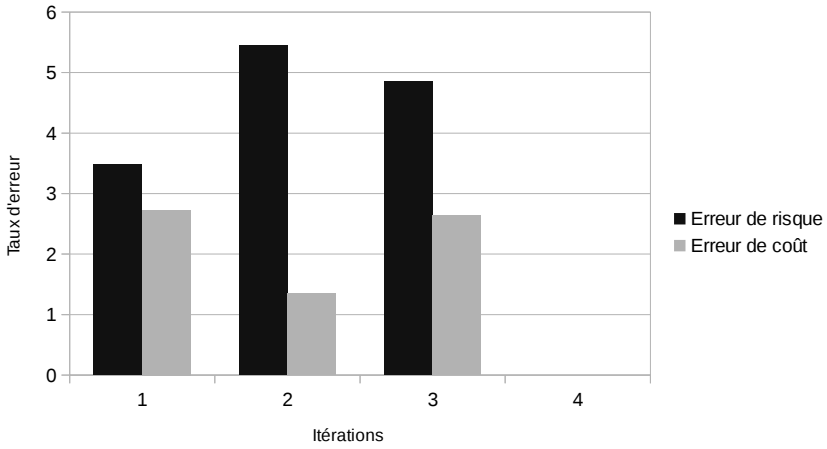


(a) Temps de calcul pour la réduction du risque

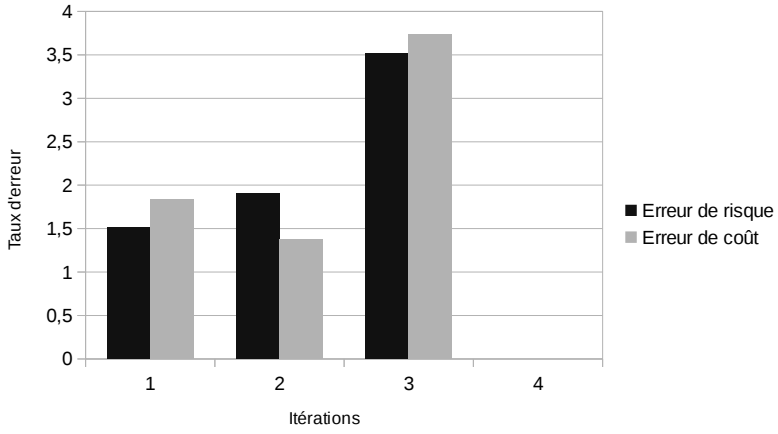


(b) Temps de calcul pour la réduction du coût

FIGURE 5.6 – Temps de calcul



(a) Taux d'erreur pour la réduction du risque



(b) Taux d'erreur pour la réduction du coût

FIGURE 5.7 – Taux d'erreur des heuristiques

- $X_{heuristics}$: la valeur retournée par l'heuristique.

Les résultats obtenus sont illustrés dans la figure 5.7. Comme illustré, l'erreur relative de la première approche n'excède pas 5,45% et 2,72% pour le risque et le coût contre 3,52% and 3,73% pour la seconde, confirmant ainsi la crédibilité des résultats obtenus en appliquant nos approches.

5.6.4 Discussion

Les résultats des expérimentations démontrent l'efficacité des heuristiques introduites dans la section 5.3 pour la réduction du risque et du coût.

Puisque ces méthodes se basent sur un ensemble d'itérations permettant d'ajouter/supprimer les contraintes (selon les besoins de l'utilisateur, i.e. réduction du risque au détriment du coût ou l'inverse), elles permettent de retourner des solutions qui se rapprochent fortement de celles des solutions exactes. En effet, la similarité de leur comportements à travers les différentes itérations nous permet de nous assurer de l'efficacité de nos approches. Par ailleurs, leur faible taux d'erreur nous fournit une garantie quant à leur précision.

5.7 Conclusion

Dans ce chapitre, nous avons présenté le dernier point abordé au cours de cette thèse, qui concerne la sélection des clouds pour l'exécution des processus métiers avec un maximum de sécurité et un coût acceptable.

Dans cette optique, nous avons proposé deux méthodes qui dépendent directement des besoins des utilisateurs. Une première dont l'objectif est de réduire les risques de sécurité auxquels le processus est exposé lors d'un déploiement dans le cloud, en fixant cependant le coût engendré à un certain seuil. La seconde permet au contraire de réduire les coûts de déploiement en fisant le risque à un certain seuil acceptable.

Ces deux méthodes se basent sur un ensemble de contraintes qui sont ajoutées/supprimées selon les besoins exprimés. Plus on veut qu'un processus soit protegé, plus on applique de contraintes, engendrant par conséquent un coût important. Si les besoins exprimés se focalisent sur les coûts, les contraintes appliquées seront réduites, exposant par contre le processus à des risques de sécurité.

Afin de s'assurer de l'applicabilité de nos approches , nous avons formalisé pour chacune d'entre elles une solution d'optimisation exacte à travers l'introduction de modèles pour le déploiement des processus métiers. Ces modèles furent par la suite utilisés comme outils pour valider nos approches. La similarité entre les résultats obtenus par les heuristiques et par les solutions exactes nous ont permis de nous assurer de l'efficacité de nos méthodes.

Troisième partie

Bilan et perspectives

Chapitre 6

Bilan et perspectives

6.1 Rappel des objectifs de la thèse

Les nouvelles technologies de l'information et de la communication ont fait l'objet d'investissements considérables ces dernières années suite à l'accroissement des besoins en informatique. En effet, les entreprises étant sujettes à des changements fréquents, requièrent un certain niveau de flexibilité et d'agilité afin de rester à la pointe du marché actuel, en offrant un haut degré d'efficacité tout en respectant les délais de livraison. Alors que la gestion des processus métiers (BPM «Business Process Management») leur permet de mieux appréhender et gérer leurs processus métiers, le cloud computing permet un sérieux gain en terme de coût et de temps quant à l'exécution de ces processus.

Cependant, étant un environnement partagé, le cloud computing a induit de sérieux risques de sécurité qui dissuadent les entreprises quant à son adoption. Plus particulièrement, les processus métiers qui reflètent le savoir-faire des organisations sont exposés à des risques de divulgation dans un environnement cloud. Afin de pallier à ce problème, il est nécessaire d'offrir aux entreprises des solutions garantissant la préservation de la logique de et la de prise de décisions de leurs processus. Le but étant de pouvoir d'une part déployer/réutiliser des fragments de processus sur/à partir du cloud afin de profiter de ses avantages. D'autre part protéger les fragments de processus comprenant des informations considérés comme étant sensibles, pouvant porter préjudice à l'organisation en cas de fuite.

6.2 Bilan des contributions

Dans ce manuscrit, nous avons souligné le besoin de protéger les processus métiers dans un environnement cloud. En effet, d'importantes informations sont contenues dans certains fragments de processus au sujet du système de prise de décisions des organisations. Ainsi, nous avons tout d'abord définis un fragment de manière générale et un fragment de processus sensible de manière particulière.

Nous avons ensuite présenté notre première contribution qui consiste en l'obfuscation d'un processus métier avant déploiement dans un environnement cloud. Cette contribution est basée sur un algorithme d'obfuscation, qui permet de décomposer d'une manière automatique un processus métier sans intervention humaine, en un ensemble de fragments déployés sur différents clouds. Cette décomposition est principalement basée sur un ensemble de contraintes de séparation. Ces contraintes ont pour objectif de minimiser le taux

d'informations pertinentes sur chaque cloud, en séparant les fragments sensibles complémentaires qui si réunis ensemble, permettent de deviner partiellement ou complètement la logique du processus.

Bien qu'efficace pour protéger un processus métier contre un éventuel cloud malicieux en raison du peu d'information se situant sur chaque cloud, cette solution ne permet pas de prévenir contre une coalition de clouds. En effet, en collaborant et en partageant leurs informations, les différents clouds de la collaboration seront à même de reconstruire la logique du processus.

Pour pallier à ce problème, nous avons introduit une approche complémentaire à la première, permettant d'introduire de faux fragments à certains endroits stratégique du processus. L'objectif étant de complexifier sa structure, dans le but de compliquer la reconstruction du processus en cas de coalition. Ceci est fait en augmentant la distance qui sépare les clouds qui déploient des tâches complémentaires. En effet, plus ils seront séparés, plus il sera difficile pour eux de collaborer i.e à cause de la nécessité de convaincre tous les clouds sur les chemins qui les séparent de faire partie de la coalition.

L'efficacité de ces méthodes fut testée à travers un ensemble de métriques que nous avons proposés, dont l'objectif est de mesurer le niveau d'obfuscation d'un processus métier lors d'un déploiement cloud. La première métrique a pour objectif de calculer la complexité d'un processus en se basant sur la séparation des fragments sensibles, ainsi que sur le nombre de messages (entrants et sortants) échangés. Contrairement à la première métrique, la seconde tient compte de la distance qui sépare les fragments sensibles complémentaires pour le calcul du niveau de complexité des processus. Cette seconde métrique a pour objectif de quantifier l'effet de l'introduction des faux fragments sur la difficulté de reconstruction du processus.

Notre troisième contribution se focalise sur le calcul du niveau de risque auquel reste exposé un processus métier après obfuscation. En effet, même si l'opération d'obfuscation permet de minimiser considérablement la possibilité de découvrir la logique du processus, un risque subsiste toujours. Ce niveau de risque basé sur des données côté client, est traduit par les niveaux de risques auxquels sont exposées les tâches sensibles. Ces niveaux de risques diffèrent selon la réputation des clouds qui déploient les tâches sensibles, la taille des chemins qui les séparent de leur tâche complémentaire, ainsi que de la réputations des clouds se situant sur ces chemins là. En effet, plus les tâches complémentaires sont séparées par des clouds de bonnes réputations, et moins les tâches sensibles seront exposées à des risques. L'objectif de cette métrique est de situer la fiabilité de la configuration choisie en fonction des besoins de l'organisation.

L'introduction de faux fragments et l'utilisation de plusieurs clouds pour le déploiement des processus est au prix d'un certain coût, qui peut se révéler être trop élevé pour la sécurité apportée, et porter atteinte à certaines dimensions de la qualité de service (QoS). Dans ce contexte, notre quatrième contribution est axée sur le côté optimisation. Plus précisément, notre objectif est de déployer les processus métiers dans un environnement cloud en répondant aux besoins des organisations en terme de risque de et coût. En effet, certaines organisations déploient des processus particulièrement sensibles et requièrent que le risque auquel est exposé leur processus soit le minimum possible. Respectivement, d'autres organisations se voit utiliser les ressources du cloud en se focalisant sur le coût engendré par ce dernier. Par ailleurs, le risque et le côut représentent deux critères étant généralement conflictuels i.e minimiser l'un augmente l'autre. Notre approche permet dans ce sens de répondre à ces besoins en déployant les processus d'une manière à répondre aux besoins des organisations, en limitant cependant le risque et le côut engendrés. Pour ce

faire, nous avons sélectionné les clouds ayant la meilleure réputation ou étant les moins coûteux (en fonction des besoins) et les avons réutilisé tant que les contraintes de séparation appliquées le permettent. Cette méthodologie nous a permis d'une part de minimiser le nombre de clouds de la collaboration en cas de nécessité de réduction du coût. D'autre part, elle nous a permis de favoriser l'utilisation des clouds ayant les meilleurs réputations en cas de nécessité de réduction du risque.

6.3 Perspectives

Dans l'objectif d'adresser certaines limites de nos travaux, nous proposons différentes perspectives qui pourraient améliorer ou approfondir les contributions de cette thèse.

Prise en compte de la nature des données Nos méthodes d'obfuscation sont prioritairement basées sur la nature des fragments i.e. séparer autant que possible les fragments sensibles complémentaires sur différents clouds. Cependant, les données échangées entre les différents clouds de la collaboration peuvent être une source d'informations pertinentes au sujet de l'activité de l'organisation et de sa stratégie de prise de décision. Notamment certaines données ayant une sensibilité élevée peuvent révéler certaines informations compromettantes, qui si entre les mains de clouds malicieux, peuvent porter préjudice à l'organisation.

Une solution que nous envisageons pour nos travaux futurs est d'une part d'obfusquer ces données avant déploiement dans un environnement cloud, et d'autre part d'utiliser une méthode d'obfuscation différente à chaque exécution. En effet, comme en général un attaquant peut inverser l'ingénierie des données utilisées en analysant un minimum d'entrées de jeu de données, utiliser une méthode différente à chaque exécution complique largement cette analyse.

Utiliser de réels clouds pour les évaluations Nous avons utilisé au cours de nos évaluations un ensemble de clouds que nous avons générés de façon aléatoire. Ainsi, nos différentes expérimentations sont plutôt considérées comme étant des simulations. Malgré le fait que ces simulations permettent effectivement de s'assurer de la validité de nos résultats, nous projetons de tester nos différentes expérimentations sur de réels clouds afin de vérifier l'applicabilité de nos méthodes d'une manière plus effective.

Par ailleurs, nous nous baserons également sur l'historique des comportements de ces clouds afin d'améliorer nos approches, dans l'optique d'éviter et de prévenir au maximum la possibilité de coalitions.

Intégrer d'autres dimensions de la qualité de service QoS Nous nous sommes concentrés au cours de cette thèse sur la sécurité et puis sur sa relation avec le coût pour évaluer la qualité d'un déploiement cloud. Mais, d'autres critères de qualité de service peuvent être considérés tel que les performances.

En effet, l'opération d'obfuscation a un impact direct sur les performances du système. Cette dégradation de performances (e.g. le temps global d'exécution) est accentuée par la complexification du processus, à travers l'introduction de faux fragments. Ainsi, notre objectif est d'étendre notre approche de déploiement de façon à tenir compte de cette dimension lors du déploiement cloud.

Bibliographie

[Acc11] Rafael Accorsi. Business process as a service : Chances for remote auditing. In *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*, pages 398–403. IEEE, 2011.

[ACR05] Ravi Aron, Eric K Clemons, and Sashi Reddi. Just right outsourcing : understanding and managing risk. *Journal of Management Information Systems*, 22(2) :37–55, 2005.

[AKC01] Edward B Allen, Taghi M Khoshgoftaar, and Ye Chen. Measuring coupling and cohesion of software modules : an information-theory approach. In *Software Metrics Symposium, 2001. METRICS 2001. Seventh International*, pages 124–134. IEEE, 2001.

[ALIJ13] Yanuarizki Amanatullah, Charles Lim, Heru Purnomo Ipung, and Arkav Juliandri. Toward cloud computing reference architecture : Cloud service management perspective. In *ICT for Smart Society (ICISS), 2013 International Conference on*, pages 1–4. IEEE, 2013.

[Ama] Amazon EC2 and S3, Online at. `http://aws.amazon.com/`.

[ARAEBA13] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. Cloud computing pricing models : a survey. *International Journal of Grid and Distributed Computing*, 6(5) :93–106, 2013.

[Arn00] Ulli Arnold. New dimensions of outsourcing : a combination of transaction cost economics and the core competencies concept. *European Journal of Purchasing & Supply Management*, 6(1) :23–29, 2000.

[Aus04] Standard Australia. Handbook : Risk management guidelines, companion to as/nzs 4360 : 2004. *Standards Australia Internal Ltd, Sydney*, 2004.

[Bao13] Aymen Baouab. Gouvernance et supervision décentralisée des choregraphies inter-organisationnelles, 2013.

[BBA12] Mehdi Bentounsi, Salima Benbernou, and Mikhail J Atallah. Privacy-preserving business process outsourcing. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 662–663. IEEE, 2012.

[BBDA12] Mehdi Bentounsi, Salima Benbernou, Cheikh S Deme, and Mikhail J Atallah. Anonyfrag : an anonymization-based approach for privacy-preserving bpaas. In *1st International Workshop on Cloud Intelligence*, page 9. ACM, 2012.

[BE81] Laszlo A Belady and Carlo J Evangelisti. System partitioning and its measure. *Journal of Systems and Software*, 2(1) :23–29, 1981.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. *Journal of the ACM (JACM)*, 59(2) :6, 2012.

[BL73] D Elliott Bell and Leonard J LaPadula. Secure computer systems : Mathematical foundations. Technical report, DTIC Document, 1973.

[BR] Sandrine Blazy and Stephanie Riaud. Measuring the robustness of source program obfuscation. *Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel*, page 203.

[Bri13] *BPMN 2.0 Distilled : The Business Process Modeling Notation*. Lulu Press, 2013.

[BS05] Arini Balakrishnan and Chloe Schulze. Code obfuscation literature survey. *CS701 Construction of compilers*, 19, 2005.

[BYV+09] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6) :599–616, 2009.

[Cap12] Jan Cappaert. Code obfuscation techniques for software protection. *Katholieke Universiteit Leuven*, pages 1–112, 2012.

[Car05] Jorge Cardoso. How to measure the control-flow complexity of web processes and workflows. *Workflow handbook*, 2005 :199–212, 2005.

[CDLLRVDA13] Raffaele Conforti, Massimiliano De Leoni, Marcello La Rosa, and Wil MP Van Der Aalst. Supporting risk-informed decisions during business process execution. In *International Conference on Advanced Information Systems Engineering*, pages 116–132. Springer, 2013.

[CDPF+14] Mariano Ceccato, Massimiliano Di Penta, Paolo Falcarin, Filippo Ricca, Marco Torchiano, and Paolo Tonella. A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques. *Empirical Software Engineering*, 19(4) :1040–1074, 2014.

[CDPN+09] Mariano Ceccato, Massimiliano Di Penta, Jasvir Nagra, Paolo Falcarin, Filippo Ricca, Marco Torchiano, and Paolo Tonella. The effectiveness of source code obfuscation : an experimental assessment. In *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*, pages 178–187. IEEE, 2009.

[Cha13] Victor Chang. A case study for business integration as a service. *Trends in E-Business, E-Services, and E-Commerce : Impact of Technology on Goods, Services, and Business Transactions : Impact of Technology on Goods, Services, and Business Transactions*, page 229, 2013.

[Clo13] Cloud Security Alliance. The Notorious Nine - Cloud Computing Top Threats in 2013. Technical report, 2013.

[CMNR06] Jorge Cardoso, Jan Mendling, Gustaf Neumann, and Hajo A Reijers. A discourse on complexity of process models. In *International Conference on Business Process Management*, pages 117–128. Springer, 2006.

[CST10] Artur Caetano, A Silva, and José Tribolet. Business process decomposition. *Enterprise Modeling and Informations Systems Architectures*, 5(1), 2010.

[CTL97] Christian Collberg, Clark Thomborson, and Douglas Low. A taxonomy of obfuscating transformations. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 1997.

[DAGM16] Karim Djemame, Django Armstrong, Jordi Guitart, and Mario Macias. A risk assessment framework for cloud computing. *IEEE Transactions on Cloud Computing*, 4(3) :265–278, 2016.

[DCD+09] Florian Daniel, Fabio Casati, Vincenzo D'Andrea, Emmanuel Mulo, Uwe Zdun, Schahram Dustdar, Steve Strauch, David Schumm, Frank Leymann, Samir Sebahi, et al. Business compliance governance in service-oriented architectures. In *2009 International Conference on Advanced Information Networking and Applications*, pages 113–120. IEEE, 2009.

[Dey08] Jeremy Deyo. Software as a service (saas), 2008.

[DLRM+13] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A Reijers, et al. *Fundamentals of business process management*, volume 1. Springer, 2013.

[DPdSS12] Evert F Duipmans, Luis Ferreira Pires, and Luiz O Bonino da Silva Santos. Towards a bpm cloud architecture with data and activity distribution. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2012 IEEE 16th International*, pages 165–171. IEEE, 2012.

[DPdSS14] Evert Ferdinand Duipmans, Luís Ferreira Pires, and Luiz Olavo Bonino da Silva Santos. A transformation-based approach to business process management in the cloud. *Journal of grid computing*, 12(2) :191–219, 2014.

[EUL09] Hanna Eberle, Tobias Unger, and Frank Leymann. Process fragments. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 398–405. Springer, 2009.

[Eur09] European Network and Information Security Agency. Benefits, risks and recommendations for information security. Technical report, 2009.

[FDG10] Walid Fdhila, Marlon Dumas, and Claude Godart. Optimized decentralization of composite web services. In *Collaborative Computing : Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, pages 1–10. IEEE, 2010.

[Fdh11] Walid Fdhila. *Décentralisation optimisée et synchronisation des procédés métiers inter-organisationnels.* PhD thesis, Université Henri Poincaré-Nancy I, 2011.

[Fil12] Hans-Georg Fill. Using obfuscating transformations for supporting the sharing and analysis of conceptual models. 2012.

[FYG09] Walid Fdhila, Ustun Yildiz, and Claude Godart. A flexible approach for automatic process decentralization using dependency tables. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 847–855. IEEE, 2009.

[FZRL08] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. Ieee, 2008.

[GANYG15] Elio Goettelmann, Amina Ahmed-Nacer, Samir Youcef, and Claude Godart. Paving the way towards semi-automatic design-time business process model obfuscation. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 559–566, 2015.

[GDG+14] Elio Goettelmann, Karim Dahman, Benjamin Gateau, Eric Dubois, and Claude Godart. A security risk assessment model for business process deployment in the cloud. In *Services Computing (SCC), 2014 IEEE International Conference on*, pages 307–314. IEEE, 2014.

[GDGG14] Elio Goettelmann, Karim Dahman, Benjamin Gateau, and Claude Godart. A broker framework for secure and cost-effective business process deployment on multiple clouds. In *26. CAiSE 2014 Forum/Doctoral Consortium*, 2014.

[Gen08] Frank Gens. It cloud services user survey, pt. 2 : Top benefits & challenges. *IDC eXchange*, 2008.

[GFG13] Elio Goettelmann, Walid Fdhila, and Claude Godart. Partitioning and cloud deployment of composite web services under security constraints. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 193–200. IEEE, 2013.

[GGS13] J Octavio Gutierrez-Garcia and Kwang Mong Sim. Agent-based cloud service composition. *Applied intelligence*, 38(3) :436–464, 2013.

[GLGB13] Ahmed Gater, Fernando Lemos, Daniela Grigori, and Mokrane Bouzeghoub. S-discovery : A behavioral and quality-based service discovery on the cloud. In *CLOSER*, pages 104–109, 2013.

[GMG13] Elio Goettelmann, Nicolas Mayer, and Claude Godart. A general approach for a trusted deployment of a business process in clouds. In *The Fifth International Conference on Management of Emergent Digital EcoSystems*, pages 92–99. ACM, 2013.

[GMLH15] Martina Gerbl, Ronan McIvor, Sharon Loane, and Paul Humphreys. A multi-theory approach to understanding the business process outsourcing decision. *Journal of World Business*, 50(3) :505–518, 2015.

[Goe15] Elio Goettelmann. *Risk-aware Business Process Modelling and Trusted Deployment in the Cloud*. PhD thesis, Université de Lorraine, 2015.

[gooa] Google App Engine, Online at. `http://code.google.com/appengine/`.

[Goob] Google. Google Docs. `https://docs.google.com`.

[Hal77] Maurice Howard Halstead. *Elements of software science*, volume 7. Elsevier New York, 1977.

[HK81] Sallie Henry and Dennis Kafura. Software structure metrics based on information flow. *IEEE transactions on Software Engineering*, (5) :510–518, 1981.

[HT12] Mohammad Hamdaqa and Ladan Tahvildari. Cloud computing uncovered : a research landscape. *Advances in Computers*, 86 :41–85, 2012.

[IBM11] IBM. IBM Cloud Computing Reference Architecture 2.0. `http://www.ibm.com/developerworks/websphere/library/techarticles/1203_lau/1203_lau.html`, 2011.

[JIB07] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2) :618–644, 2007.

[JSGI09] Meiko Jensen, Jörg Schwenk, Nils Gruschka, and Luigi Lo Iacono. On technical security issues in cloud computing. In *2009 IEEE International Conference on Cloud Computing*, pages 109–116. Ieee, 2009.

[JZ11] Danish Jamil and Hassan Zaki. Cloud computing security. *International Journal of Engineering Science and Technology*, 3(4), 2011.

[Kal90] George E Kalb. Counting lines of code, confusions, conclusions, and recommendations. In *Briefing to the 3rd Annual REVIC User's Group Conference*, 1990.

[KLWL09] Ryan KL Ko, Stephen SG Lee, and Eng Wah Lee. Business process management (bpm) standards : a survey. *Business Process Management Journal*, 15(5) :744–791, 2009.

[KMZ15] Kenji E Kushida, Jonathan Murray, and John Zysman. Cloud computing : from scarcity to abundance. *Journal of Industry, Competition and Trade*, 15(1) :5–19, 2015.

[KNRS13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.

[Knu90] Donald E Knuth. The genesis of attribute grammars. In *Attribute Grammars and Their Applications*, pages 1–12. Springer, 1990.

[Ko09] Ryan KL Ko. A computer scientist's introductory guide to business process management (bpm). *Crossroads*, 15(4) :4, 2009.

[KS07] Robert J Kauffman and Ryan Sougstad. Value-at-risk in it services contracts. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 63–63. IEEE, 2007.

[KSP12] CD Karthic, S Sujatha, and V Praveenkumar. A dynamic cloud discovery framework for deploying of scientific computing services over a multi-cloud infrastructure. *Journal of Artificial Intelligence*, 5(4) :161, 2012.

[LD03] Cullen Linn and Saumya Debray. Obfuscation of executable code to improve resistance to static disassembly. In *The 10th ACM conference on Computer and communications security*, pages 290–299. ACM, 2003.

[LGSS12] Rafel Los, Don Gray, Dave Shackleford, and Bryan Sullivan. „top threats to cloud computing, 2012.

[LMS14] Wadha Labda, Nikolay Mehandjiev, and Pedro Sampaio. Modeling of privacy-aware business processes in bpmn to protect personal data. In *The 29th Annual ACM Symposium on Applied Computing*, pages 1399–1405. ACM, 2014.

[LOM+14] T Lynn, N O'Carroll, J Mooney, M Helfert, D Corcoran, G Hunt, L Van Der Werff, J Morrison, and P Healy. Towards a framework for defining and categorising business process-as-a-service (bpaas). In *21st International Product Development Management Conference. Citations : Not Avail*, 2014.

[LR00] Frank Leymann and Dieter Roller. Production workflow : concepts and techniques. 2000.

[LS07] Ruopeng Lu and Shazia Sadiq. A survey of comparative business process modeling approaches. In *International Conference on Business Information Systems*, pages 82–94. Springer, 2007.

[LV92] Lawrence Loh and N Venkatraman. Determinants of information technology outsourcing : a cross-sectional analysis. *Journal of management information systems*, 9(1) :7–24, 1992.

[LWF96] Mary C Lacity, Leslie P Willcocks, and David F Feeny. The value of selective it sourcing. *Sloan management review*, 37(3) :13, 1996.

[MA04] Ian McCarthy and Angela Anagnostou. The impact of outsourcing on the transaction costs and boundaries of manufacturing. *International journal of production economics*, 88(1) :61–71, 2004.

[MADB+06] Matias Madou, Bertrand Anckaert, Bruno De Bus, Koen De Bosschere, Jan Cappaert, and Bart Preneel. On the effectiveness of source code transformations for binary obfuscation. In *The International Conference on Software Engineering Research and Practice (SERP06)*, pages 527–533. CSREA Press, 2006.

[May09] Nicolas Mayer. *Model-based management of information system security risk*. PhD thesis, University of Namur, 2009.

[McC76] Thomas J McCabe. A complexity measure. *IEEE Transactions on software Engineering*, (4) :308–320, 1976.

[MG11] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[Mic] Microsoft. Windows Azure. `http://microsoft.com/windowsazure`.

[MLB+11] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—the business perspective. *Decision support systems*, 51(1) :176–189, 2011.

[MR11] Rafael Martí and Gerhard Reinelt. *The linear ordering problem : exact and heuristic methods in combinatorial optimization*, volume 175. Springer Science & Business Media, 2011.

[MTG11] Benedikt Martens, Frank Teuteberg, and Matthias Gräuler. Design and implementation of a community platform for the evaluation and selection of cloud computing services : a market analysis. In *ECIS*, volume 4, page 50, 2011.

[MWT12] Benedikt Martens, Marc Walterbusch, and Frank Teuteberg. Costing of cloud computing services : A total cost of ownership approach. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1563–1572. IEEE, 2012.

[NGY+15] Amina Ahmed Nacer, Elio Goettelmann, Samir Youcef, Abdelkamel Tari, and Claude Godart. Business process design by reusing business process fragments from the cloud. In *Service-Oriented Computing and Applications (SOCA), 2015 IEEE 8th International Conference on*, pages 193–200. IEEE, 2015.

[NGY+16] Amina Ahmed Nacer, Elio Goettelmann, Samir Youcef, Abdelkamel Tari, and Claude Godart. Obfuscating a business process by splitting its logic with fake fragments for securing a multi-cloud deployment. In *Services (SERVICES), 2016 IEEE World Congress on*, pages 18–25. IEEE, 2016.

[NGY+18] Amina Ahmed Nacer, Elio Goettelmann, Samir Youcef, Abdelkamel Tari, and Claude Godart. A design-time semi-automatic approach for obfuscating a business process model in a trusted multi-cloud deployment : A design-time approach for bp obfuscation. *International Journal of Web Services Research (IJWSR)*, 15(4) :61–81, 2018.

[NGYT17] Amina Ahmed Nacer, Claude Godart, Samir Youcef, and Abdelkamel Tari. A metric for evaluating the privacy level of a business process logic in a multi-cloud deployment. In *IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, 2017.

[OMG13] Object Management Group OMG. Unified modelling language (uml) 2.5. Technical report, OMG, 2013.

[Ovi93] Enrique I Oviedo. Control flow, data flow and program complexity. In *Software engineering metrics I*, pages 52–65. McGraw-Hill, Inc., 1993.

[Par66] Douglas F Parkhill. Challenge of the computer utility. 1966.

[PdSPdP14] Lucas Venezian Povoa, Wanderley Lopes de Souza, Luís Ferreira Pires, and Antonio Francisco do Prado. An approach to the decomposition of business processes for execution in the cloud. In *Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on*, pages 470–477. IEEE, 2014.

[PGBD10] Artem Polyvyanyy, Luciano García-Bañuelos, and Marlon Dumas. Structuring acyclic process models. In *International Conference on Business Process Management*, pages 276–293. Springer, 2010.

[PVV10] Artem Polyvyanyy, Jussi Vanhatalo, and Hagen Völzer. Simplified computation and generalization of the refined process structure tree. In *International Workshop on Web Services and Formal Methods*, pages 25–41. Springer, 2010.

[QRD14] Clément Quinton, Daniel Romero, and Laurence Duchien. Automated selection and configuration of cloud environments using software product lines principles. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pages 144–151. IEEE, 2014.

[Rac] RackSpace. RackSpace Cloud. `http://rackspacecloud/index.php`.

[RBBA15] Mouna Rekik, Khouloud Boukadi, and Hanene Ben-Abdallah. Business process outsourcing to the cloud : What activity to outsource ? In *Computer Systems and Applications (AICCSA), 2015 IEEE/ACS 12th International Conference of*, pages 1–7. IEEE, 2015.

[RBBA16] Mouna Rekik, Khouloud Boukadi, and Hanêne Ben-Abdallah. A comprehensive framework for business process outsourcing to the cloud. In *Services Computing (SCC), 2016 IEEE International Conference on*, pages 179–186. IEEE, 2016.

[RYC17] Guillaume Rosinosky, Samir Youcef, and François Charoy. Efficient migration-aware algorithms for elastic bpmaas. In *International Conference on Business Process Management*, pages 147–163. Springer, 2017.

[Sal] SalesForce. SalesForce. `http://salesforce.com/`.

[Sav] Savvis. Savvis Symphony. `http://wwweb.jextensions.com/savvisknowscloud.com/`.

[SDH+14] Le Sun, Hai Dong, Farookh Khadeer Hussain, Omar Khadeer Hussain, and Elizabeth Chang. Cloud service selection : State-of-the-art and future research directions. *Journal of Network and Computer Applications*, 45 :134–150, 2014.

[SG12] Vassil Stoitsev and Paul Grefen. Business process technology and the cloud : Defining a business process cloud platform. Technical report, Beta Working Paper Series 393, School of Industrial Engineering, Eindhoven University of Technology, 2012.

[SK11] Subashini Subashini and Veeraruna Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1) :1–11, 2011.

[SLGL09] Monirul Sharif, Andrea Lanzi, Jonathon Giffin, and Wenke Lee. Automatic reverse engineering of malware emulators. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 94–109. IEEE, 2009.

[SLM+10] David Schumm, Frank Leymann, Zhilei Ma, Thorsten Scheibler, and Steve Strauch. Integrating compliance into business processes. *Multikonferenz Wirtschaftsinformatik 2010*, page 421, 2010.

[SSZK12] Maria Salama, Ahmed Shawish, Amir Zeid, and Mohamed Kouta. Integrated qos utility-based model for cloud computing service provider selection. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pages 45–50. IEEE, 2012.

[Str10] Anja Strunk. Qos-aware service composition : A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74. IEEE, 2010.

[SW03] Jingqiu Shao and Yingxu Wang. A new measure of software complexity based on cognitive weights. *Canadian Journal of Electrical and Computer Engineering*, 28(2) :69–74, 2003.

[Sys04] Sparkx Systemps. The Business Process Model. `http://www.sparxsystems.com.au`, 2004.

[VDATHW03] Wil MP Van Der Aalst, Arthur HM Ter Hofstede, and Mathias Weske. Business process management : A survey. In *International conference on business process management*, pages 1–12. Springer, 2003.

[VdAvDH+03] Wil MP Van der Aalst, Boudewijn F van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and Anton JMM Weijters. Workflow mining : a survey of issues and approaches. *Data & knowledge engineering*, 47(2) :237–267, 2003.

[Vie09] John Viega. Cloud computing and the common man. *Computer*, 42(8) :106–108, 2009.

[VVK09] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. *Data & Knowledge Engineering*, 68(9) :793–818, 2009.

[VVL07] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and more focused control-flow analysis for business process models through sese decomposition. In *International Conference on Service-Oriented Computing*, pages 43–55. Springer, 2007.

[Wat12] Paul Watson. A multi-level security model for partitioning workflows over federated clouds. *Journal of Cloud Computing : Advances, Systems and Applications*, 1(1) :15, 2012.

[WBWK08] Kim Wüllenweber, Daniel Beimborn, Tim Weitzel, and Wolfgang König. The impact of process standardization on business process outsourcing success. *Information Systems Frontiers*, 10(2) :211–224, 2008.

[Wes07] Mathias Weske. *Business process management : concepts, languages, architectures*. Springer-Verlag Berlin Heidelberg, Nomber 2007.

[Wes12] Mathias Weske. Business process management architectures. In *Business Process Management*, pages 333–371. Springer, 2012.

[WHFL04] Leslie Willcocks, John Hindle, David Feeny, and Mary Lacity. It and business process outsourcing : The knowledge potential. *Information systems management*, 21(3) :7–15, 2004.

[WHH79] Martin R. Woodward, Michael A. Hennell, and David Hedley. A measure of control flow complexity in program text. *IEEE Transactions on Software Engineering*, (1) :45–50, 1979.

[Wro02] Gregory Wroblewski. *General method of program code obfuscation (draft)*. PhD thesis, Citeseer, 2002.

[XWG11] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23(8) :1200–1214, 2011.

[ZCB10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1) :7–18, 2010.

[Zoh] Zoho. Zoho Productivity Applications. `http://zoho.com/`.

[ZZZ09] Wenying Zeng, Yuelong Zhao, and Junwei Zeng. Cloud service and service selection algorithm research. In *The first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 1045–1048. ACM, 2009.