Ce chapitre présente une étude bibliographique autour des concepts, des approches et des technologies liés aux objectifs abordés dans le chapitre précédent : la gestion des propriétés non-fonctionnelles des services, la prise en compte de ces propriétés lors de l'étape de modélisation des processus métier et lors de la réconciliation entre les activités métier et les services techniques, et enfin la rationalisation de la gestion des services et de leurs métadonnées en utilisant la gouvernance SOA. Cet état de l'art sert de base au travail de recherche visant à atteindre ces objectifs. En nous basant sur la littérature, nous commençons par détailler l'architecture orientée service, sa brique principale : les services, les services Web ainsi que leurs propriétés non-fonctionnelles. Par la suite, nous introduisons les processus métier d'une façon générale avant de nous intéresser aux processus métier appliqués dans un contexte SOA. Nous étudions les standards existants pour la modélisation de ces processus et pour l'annotation avec des exigences non-fonctionnelles. Enfin, nous étudions la gouvernance SOA et les approches de sélection de services selon les exigences non-fonctionnelles.

Architecture orientée services (SOA)

En quelques années, l'architecture orientée services est devenue un thème majeur pour le système d'information des entreprises. Elle représente une nouvelle approche de conception des applications qui fait gagner en flexibilité, en agilité et en réactivité au niveau du système d'information en reposant sur le principe de couplage lâche et tout en garantissant l'interopérabilité entre ces applications [Erl, 2004], [Krafzig et al., 2005]. Ceci a encouragé les entreprises à baser leur système d'information sur cette nouvelle philosophie. Bieberstein définit l'architecture SOA comme suit :

> « *Service-Oriented Architecture enables organizations to be agile and flexible enough to adopt new business strategies and produce new services to overcome the challenges created by business dynamism today.* » *[Bieberstein, 2006]*

La SOA représente une nouvelle forme architecturale pour les systèmes d'information. Elle fournit une vision où les applications (internes et externes) sont exposées sous forme de services et peuvent communiquer entre elles d'une manière simple [Raymond, 2007]. Dans cette vision, chaque application fournit une partie (ou l'ensemble) de ses fonctionnalités sous forme de services invocables par d'autres applications. Ce concept d'architecture utilise donc les services comme des composants principaux pour la construction des applications distribuées d'une manière facile et peu coûteuse. Ceci

permet au système d'information de prendre en compte les quatre enjeux suivants [Jardim-Goncalves et al., 2006], [Dan et al., 2008] et [Papazoglou et al., 2008] :

- *Rationalisation* : diminution de la duplication des modules et du code, en réduisant les dépendances entre les applications et en urbanisant les liens inter-applicatifs. D'où une réduction des coûts ;
- *Interopérabilité* : possibilité pour les blocs du système d'information (composants, applications, services, etc.) d'échanger des données et des informations tout en assurant la cohérence entre eux ;
- *Réutilisation* : réduction des risques et de la complexité du système d'information par la réutilisation des composants (ou une partie de leurs codes testés) avant de décider de créer des nouveaux ;
- *Agilité* : capacité du système d'information à appréhender et à prendre en compte les évolutions du métier de l'entreprise en y apportant des réponses simples, efficaces et rapides.

II.2.1. Les services

Les services représentent la brique de base d'une architecture SOA et peuvent être définis ainsi :

> *« Un service est une unité logicielle autonome et indépendante de toute plateforme, qui peut être décrite, publiée, découverte, orchestrée et programmée en utilisant des protocoles standards dans le but de créer des réseaux d'applications collaboratives distribuées à très grande échelle. »* [Baligand, 2008]

Un service est composé de trois parties principales, tel que illustré dans la Figure 8 : (i) l'interface qui constitue la partie « *publique* » et contient entre autres la liste des opérations, les entrées et les sorties, (ii) l'implémentation qui constitue la partie « *privée* » et correspond au développement des fonctionnalités du service. Même si l'implémentation dépend de la plateforme utilisée, l'approche SOA permet d'abstraire l'hétérogénéité technique grâce à la publication d'une interface commune appelable. Enfin (iii) le point de déploiement (*end-point*) qui représente le point par lequel le service peut être invoqué, généralement il correspond à une adresse URL (*Uniform Resource Locator*).

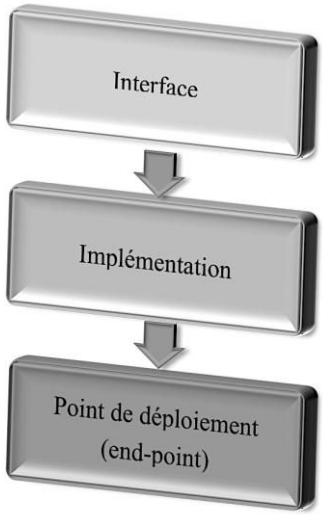

Figure 8 : *Structuration d'un service.*

Ce qui distingue l'architecture SOA des autres architectures, c'est le couplage lâche et la granularité des services. Ces propriétés permettent de faciliter la reconfiguration des processus quand les fonctions métier évoluent ou changent. Le concept de service, qui est véhiculé par le modèle de l'architecture orientée services, se veut indépendant de la mise en œuvre des applications constituantes. D'autre part, la granularité des services rend possible et plus flexible l'adaptation, l'évolution des applications et leurs collaborations avec d'autres.

Le fournisseur du service et les utilisateurs (appelés aussi consommateurs) interagissent entre eux sans qu'ils n'aient à faire de concession sur la structure propre, le comportement interne et l'implantation des services. Concrètement, cela se déroule en quatre phases réunificatrices comme l'illustre la Figure 9 ci-après.

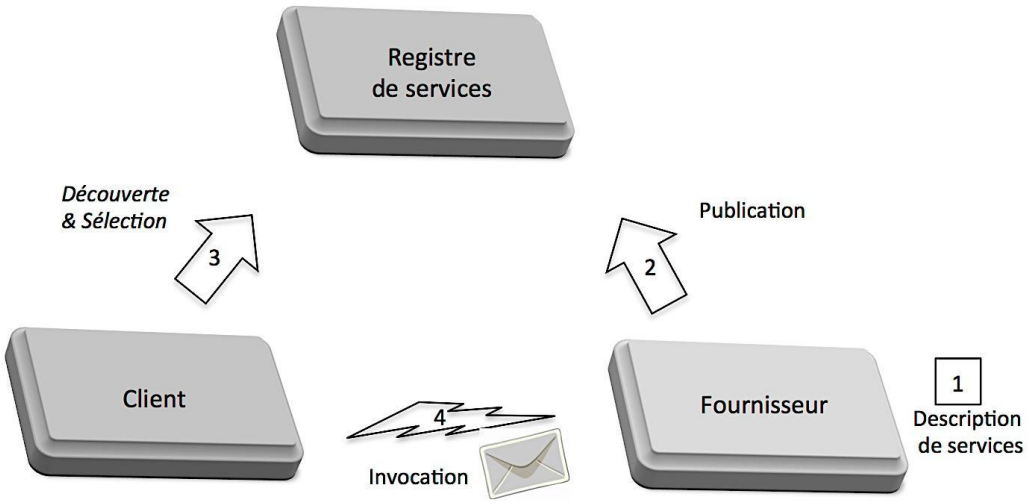

Figure 9 : *Principe de fonctionnement des services.*

1) La *description de services* : les fournisseurs décrivent tous les détails concernant les fonctionnalités auxquelles leurs services répondent, à savoir : les paramètres d'entrée, le format et le type de données retournées ainsi que le protocole à utiliser lors des requêtes dans le but de faciliter la tâche aux fournisseurs ;

2) La *publication de services* : les fournisseurs, après avoir décrit leurs services, vont les publier à l'extérieur pour qu'ils soient disponibles pour les utilisateurs. La publication consiste donc à enregistrer les descriptions de services dans le registre ;

3) La *découverte et la sélection de services* : cette phase recouvre la possibilité donnée aux utilisateurs de rechercher un service parmi tous ceux qui ont été déjà publiés dans le registre de services. Par la suite, l'utilisateur peut choisir un ou plusieurs services disponibles ;

4) *L'invocation de services* : cette dernière phase consiste à invoquer le service choisi pour l'utiliser. L'invocation se fait via la transmission de messages.

II.2.2. Les services Web

Les services Web sont des services qui peuvent être publiés, découverts et invoqués par le biais de l'internet. Cette technologie de services constitue une solution pour implémenter une architecture SOA [Newcomer et al., 2004]. Cependant, un service Web peut être considéré comme étant une application autonome, mise à disposition d'autres applications et qui peut être publiée, découverte et invoquée via une infrastructure du Web [Tidwell, 2000] et [Alonso et al., 2004]. Le consortium UDDI (*Universal Description, Discovery and Integration*) fournit une définition plus spécifique d'un service Web et le caractérise comme suit :

> *« Un service Web est une application métier autonome et modulaire qui possède des interfaces basées sur des standards et orientées Internet.* » [OASIS, 2000]

Selon cette définition, les services Web fonctionnent d'une manière qui n'est pas intégrée. C'est-à-dire qu'au lieu d'intégrer toutes les fonctionnalités dans une seule application globale, chacune des fonctionnalités est développée en tant qu'un service Web, ce qui facilitera sa réutilisation par d'autres applications. Le consortium W3C propose une définition encore plus détaillée autour des services Web :

> *« A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages.* » [W3C, 2004]

Le service Web est une application logicielle, autonome, modulaire et appelable via les protocoles du Web par d'autres applications existantes sur le Web permettant l'échange de données. Les requêtes de demandes et de réponses doivent être conformes aux protocoles et technologies standards. Les trois standards les plus connus et piliers des services Web sont les suivants :

- ***WSDL*** : (*Web Service Description Language*) [Christensen et al., 2001] est un format XML qui décrit les informations nécessaires des services (comment se connecter et utiliser un service Web, etc.). Il décrit : (i) l'interface publique : les différentes opérations qu'elle propose et la structure des messages d'entrée/sortie pour chacune et (ii) le protocole de

communication : le format des messages que le service utilise, les méthodes que le client peut invoquer et la localisation du service ;

- ***UDDI*** : (*Universal Description, Discovery and Integration*) [OASIS, 2000] normalise une solution d'annuaire distribué de services Web, permettant à la fois la publication et l'exploration (recherche) de services ;
- ***SOAP*** : (*Simple Object Access Protocol*) [W3C, 2000] est un protocole d'échange inter-applications indépendant de toute plateforme et basé sur le langage XML.

Ces standards permettent, dès lors qu'ils sont implémentés sur les plateformes et les produits, aux clients et aux services de communiquer sur un très large spectre de plateformes et d'environnements. Dans la Figure 10, nous décrivons le principe de la découverte et de l'utilisation d'un service Web. Le client souhaitant rechercher un service Web effectue une requête de découverte sur le registre de services UDDI. Le registre, à son tour, trie parmi les services dont il dispose en se basant sur les critères demandés par le client avant de lui retourner une liste d'adresses des fichiers WSDL qui répondent à ses besoins. Le client pourra par la suite accéder au service via le document WSDL et l'invoquer pour l'utiliser grâce au protocole SOAP.

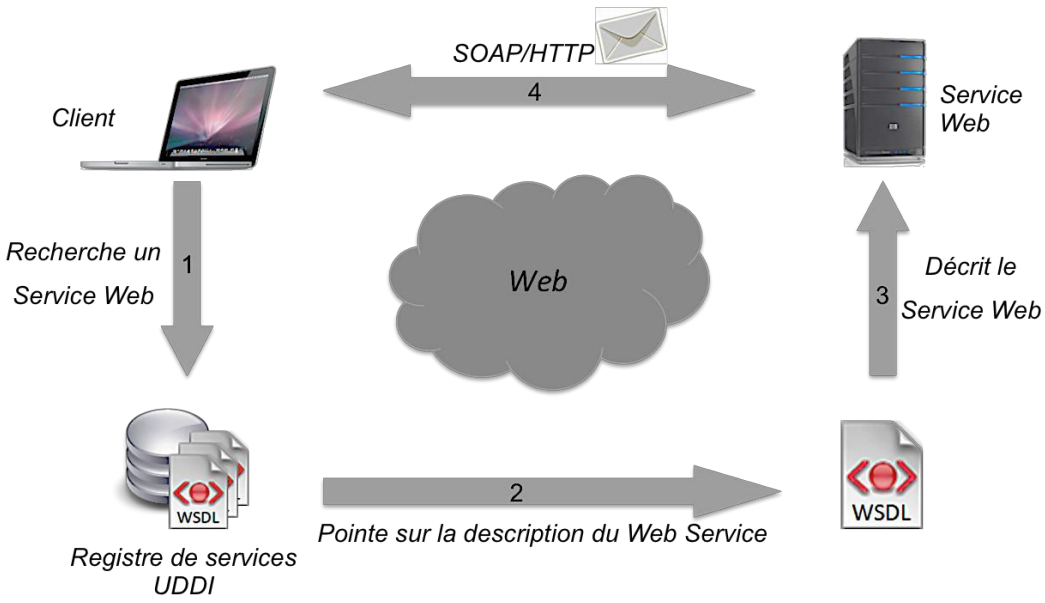

Figure 10 : *Principe de découverte et d'utilisation d'un service Web.*

Les approches basées sur ces trois technologies, à savoir WSDL, UDDI et SOAP, permettent une découverte fonctionnelle des services Web. C'est-à-dire une découverte selon les propriétés fonctionnelles qui décrivent les opérations accomplies par le service. D'autres approches ont été développées afin d'enrichir cette découverte. Parmi ces approches nous citons : les *approches non-fonctionnelles* qui proposent des propriétés non-fonctionnelles pour définir les services Web d'un

point de vue qui va au delà de ses fonctionnalités intrinsèques telles que : la performance, la qualité, la sécurité, l'interopérabilité, la fiabilité, etc. Au cours de ces travaux, notre intérêt se porte sur les approches non-fonctionnelles afin d'enrichir davantage la description des services.

II.2.3. Les propriétés non-fonctionnelles des services Web

Les propriétés non-fonctionnelles jouent un rôle très important et représentent un critère essentiel pour la pertinence d'un service [Papazoglou et al., 2007] lors de la sélection [Liu et al., 2004] ou de la composition [Zeng et al., 2004]. Cette section permet de décrire les propriétés non-fonctionnelles des services et de présenter quelques approches courantes de leur modélisation.

Les propriétés non-fonctionnelles, aussi appelées les propriétés extra-fonctionnelles, des services concernent tout ce qui ne découle pas directement des aspects fonctionnels comme la performance, le prix, la sécurité [Chung et al., 1995]. À titre d'exemple, dans le projet CHOReOS, pour pouvoir faire un déploiement auto-adaptatif du bus de services, un service Web Amazon offrant cette fonctionnalité est utilisé. Toutefois, l'utilisation est payante. Le prix du service constitue une propriété non-fonctionnelle.

Dans la littérature et dans une architecture SOA, les propriétés non-fonctionnelles d'un service sont souvent liées au concept de la qualité de service (*Quality of Service* - QoS) et sont même parfois utilisées sans aucune distinction. Toutefois, les auteurs dans [Cardoso et al., 2004], ne limitent pas les propriétés non-fonctionnelles qu'à ce seul concept de QoS et proposent une classification en deux catégories : les propriétés quantitatives et les propriétés qualitatives. Les propriétés quantitatives sont exprimées par des valeurs et dépendent fortement de l'infrastructure réseau tels que le temps de réponse, la disponibilité, la fiabilité, etc. Les propriétés qualitatives ne dépendent pas de l'infrastructure réseau et sont parfois plus simples à énoncer, par exemple la sécurité et l'interopérabilité qui sont exprimées en valeur booléenne, etc. Dans la suite de ce manuscrit, nous considérons cette définition contextuelle des propriétés non-fonctionnelles :

> *« Les propriétés non-fonctionnelles d'un service Web regroupent l'ensemble des paramètres qui correspondent à la réalisation de ses fonctionnalités. Ces paramètres peuvent être quantitatifs ou qualitatifs. C'est-à-dire l'ensemble de ses caractéristiques qui relèvent de l'infrastructure réseau (Qualité de Service) et celles qui ne le sont pas (sécurité, interopérabilité, etc.). »* [Cardoso et al., 2004]

Plusieurs approches et modèles se sont intéressés à enrichir la description des services Web en rajoutant des propriétés non-fonctionnelles [O'Brien et al., 2007]. Dans ce qui suit, nous présentons quelques modèles et langages pour l'annotation non-fonctionnelle des services.

QoS-based WS Description : *Quality of Service Web Service Description* est une ontologie développée sous le nom OWL-Q (*Web Ontology Language for Quality*). Elle représente une extension du standard OWL-S (*Web Ontology Language for Services*) [Kritikos et al., 2006]. Elle est basée sur un ensemble de facettes (onze facettes) qui sont développées séparément et qui peuvent être étendues. La facette principale concerne la définition d'un système complet pour associer des métriques aux propriétés non-fonctionnelles. Si ce langage permet la distinction entre les propriétés statiques et celles qui ne le sont pas (c'est-à-dire les propriétés qui ont des valeurs qui peuvent changer lors de l'exécution du service telles que le temps de réponse, la disponibilité, etc.), il ne permet pas de définir une hiérarchie entre ces attributs et ne fournit pas de liaisons avec les standards de qualité existants.

WS-QoSOnt : *Web Service-Quality of Service Ontology* est une ontologie proposée par Tran dans [Tran, 2008] et fournit une description de la QoS pour les services Web. Cette ontologie fournit une taxonomie de propriétés qui couvre la performance, la disponibilité, la fiabilité et les aspects de sécurité. Néanmoins, cette ontologie ne prend pas en considération les différents niveaux des propriétés non-fonctionnelles. C'est-à-dire les propriétés non-fonctionnelles qui sont appliquées au niveau métier et celles qui sont plutôt techniques.

WS-Policy : *Web Service-Policy*, est un standard développé par le consortium W3C en 2006 [W3C, 2006]. Il est composé de deux spécifications XML distinctes : *WS-Policy et WS-Policy Attachment* (ce dernier est défini pour permettre la compatibilité entre les différentes versions des standards WSDL et WS-Policy). Ce standard a pour objet d'améliorer l'interaction entre le fournisseur du service et les utilisateurs en rajoutant des informations supplémentaires (ou métadonnées) concernant les services (au-delà de celles qui sont déjà prévues dans le document WSDL). Le concept de base de WS Policy est l'*assertion*. Une assertion est une métadonnée qui exprime une exigence spécifique pour un service donné. Par exemple, une assertion pourrait indiquer des propriétés non-fonctionnelles telles que : un niveau requis de sécurité, l'utilisation d'un certain protocole, ou un temps de réponse requis lors de l'invocation du service, etc. Une assertion dans WS-Policy peut prendre la forme suivante :

<ServiceQuality ResponseTime_seconde = "1">

WSQF : la spécification de *Web Service Quality Factors* a été publiée en 2010 par le comité technique du *Web Service Quality Model* en tant que spécification d'OASIS [OASIS, 2010]. Elle regroupe les caractéristiques de base d'un service Web en proposant un ensemble de propriétés (ou facteurs) non-fonctionnelles. Comme l'illustre la Figure 11, ces propriétés sont classées en six groupes, à savoir *Business Value Quality*, *Service Level Measurement*, *Interoperability*, *Business Processing, Manageability* et enfin *Security*.

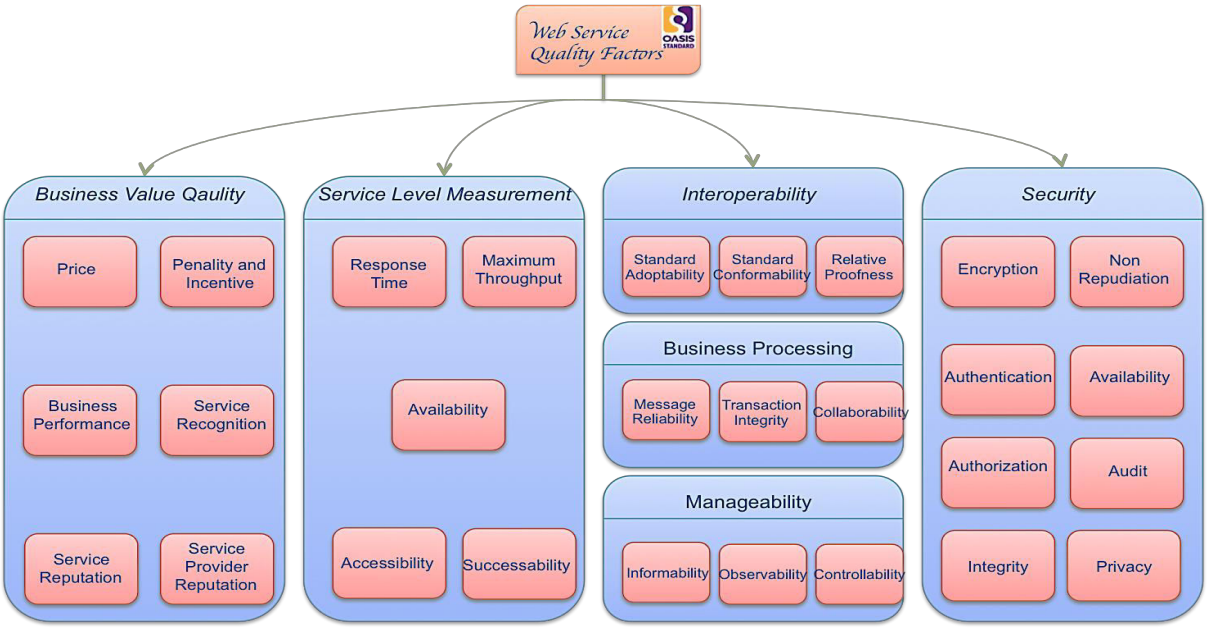

Figure 11 : *Propriétés non-fonctionnelles proposées par le standard WSQF.* [OASIS, 2010].

- Valeur de la qualité métier (*Business Value Quality*) : cette catégorie de propriétés se réfère à l'évaluation des services Web d'un point de vue métier. Parmi les propriétés non-fonctionnelles qui le composent, nous citons : le prix (*Price*), la popularité du service (*Service Recognition*), la réputation du service (*Service Reputation*), la réputation du fournisseur du service (*Service Provider Reputation*), etc.
- Mesure du niveau de la qualité de service (*Service Level Measurement Quality*) : cette catégorie caractérise la qualité du service qui doit accompagner l'utilisation du service. Cette catégorie regroupe le temps de réponse (*Response Time*), le débit maximum (*Maximum Throughput*), la disponibilité (*Availability*, représente le pourcentage du temps au cours duquel le service Web, en état de fonctionnement, est disponible), l'accessibilité (*Accessibility*, représente le pourcentage du temps au cours duquel le service Web, en état de fonctionnement et disponible sur le réseau, est accessible) et enfin la successibilité (*Successability*, représente le pourcentage de succès lors d'une exécution complète du service Web) ;
- Interopérabilité (*Interoperability*) : afin de garantir l'interopérabilité entre les services Web, l'utilisation conforme des standards lors de l'implémentation est fortement conseillée. Parmi les propriétés non-fonctionnelles d'interopérabilité, nous citons l'adoptabilité des normes (*Standard Adoptability*), la conformité aux standards (*Standard Conformability*) et la pérennité relative (*Relative Proofness*) ;
- Traitement métier (*Business Processing*) : les services Web peuvent êtres menés à être utilisés dans des échanges critiques entre les partenaires métier, et dans ce cas, la fiabilité et la stabilité des services Web sont très importantes. *Business Processing* évalue ces

éléments intégrant aussi la fiabilité des messages (*Message Reliability*), l'intégrité des transactions (*Transaction Integrity*) et la capacité de collaboration (*Collaborability,* représente la capacité d'une plateforme de services à définir, contrôler et gérer les flux de services entre les participants afin de permettre aux services Web de participer dans une collaboration telle qu'une orchestration ou une chorégraphie) ;

- Gérabilité (*Manageability*) : les fonctions de gérabilité permettent d'évaluer le fonctionnement du service par rapport à ce qui a été prévu. Par ailleurs, en cas de défaillance, ces fonctions permettent d'évaluer l'existence des informations primitives et nécessaires au maintien du fonctionnement du service ;
- Sécurité (*Security*) : les services Web sont fréquemment exposés à des réseaux publics ce qui les rend vulnérables aux attaques et aux fraudes. La sécurité identifie le degré de protection du service par rapport aux diverses menaces. Nous retrouvons dans cette catégorie : le cryptage (*Encryption*), la non-répudiation (*Non-Repudiation*), l'authentification (*Authentication*), la disponibilité (*Availability*), l'autorisation (*Authorization*), le contrôle (*Audit*), l'intégrité (*Integrity*), et la confidentialité (*Privacy*).

Nous invitions le lecteur à se référer à l'Annexe 1 pour retrouver la description de toutes les propriétés non-fonctionnelles proposées par la spécification WSQF.

II.2.4. Synthèse sur les propriétés non-fonctionnelles

De nombreuses approches basées sur des ontologies ont été proposées pour l'amélioration de la description des services Web en rajoutant des propriétés non-fonctionnelles comme QoS-based WS Description et WS-QoSOnt. Tous ces travaux ne permettent pas d'identifier des niveaux d'abstractions des propriétés non-fonctionnelles, c'est-à-dire les propriétés qui concernent le niveau métier et celles qui sont plutôt d'un niveau technique. De même, ces approches basées sur les ontologies ne permettent pas de distinguer les propriétés non-fonctionnelles qui sont mesurables (qui doivent être contrôlées et suivies lors de l'exécution du service Web) de celles qui ne le sont pas.

Ainsi, les propriétés non-fonctionnelles telles que le temps de réponse ou la disponibilité (qui sont des propriétés mesurables) se retrouvent au même niveau que la sécurité ou l'interopérabilité par exemple. De plus, certaines de ces approches ne se sont pas intéressées à établir des liaisons avec les différents standards qui existent pour la qualité de services (par exemple WS-Security pour les propriétés de sécurité). Cependant, il n'existe pas de lien concret entre les propriétés non-fonctionnelles définies par ces approches et les propriétés des différents standards de qualité. Pourtant, ce type de lien s'avère crucial notamment pour l'interopérabilité et pour éviter les problèmes de terminologie (en utilisant des standards communément admis).

Ces problèmes ont été résolus par l'apparition de la spécification WSQF de OASIS [OASIS, 2010]. Elle présente une solution plutôt complète pour la description des services Web en prenant en compte l'aspect non-fonctionnel. WSQF définit un ensemble assez large de propriétés non-fonctionnelles et offre la possibilité d'être lié à des standards tels que le WS-Interoperability (WS-I) pour la gestion de l'interopérabilité, le standard WS-Security pour la gestion de la sécurité, ou le standard WS-Policy pour la gestion des règles. Pour ses multiples avantages, nous avons décidé d'utiliser la spécification WSQF d'OASIS pour l'amélioration de la définition des services par la prise en considération des propriétés non-fonctionnelles.

II.3. Rationalisation de la gestion des services et de leurs propriétés non-fonctionnelles : gouvernance SOA

Avec la prolifération des services au sein du système d'information, il est devenu essentiel aux entreprises de rationaliser leur gestion ainsi que la gestion de leurs propriétés non-fonctionnelles. La problématique de la gouvernance permet d'adresser ces besoins. Nous présentons dans cette section un des composants clés de l'architecture SOA : *la gouvernance SOA* ainsi qu'une étude sur les cadres de gouvernance existants.

II.3.1. Qu'est-ce que la gouvernance SOA?

Le terme *gouvernance* vient du grec ancien *kubernân* qui signifie piloter un navire ou un char [De Oliveira Barata, 2003]. Dans le monde des entreprises, la gouvernance d'entreprise (*Corporate Governance*) représente une fonction essentielle pour les systèmes d'information et peut être définie comme étant un processus que l'entreprise met en place pour veiller à ce que les choses soient faites conformément aux règles préalablement définies. Dans [Boutillier et al., 2002], les auteurs définissent la gouvernance d'entreprise par *l'ensemble des processus, réglementations, lois, et institutions influant la manière dont l'entreprise est dirigée, administrée et contrôlée*.

S'inspirant de la gouvernance d'entreprise, la gouvernance informatique (*IT Governance*), appelée aussi gouvernance des systèmes d'information, garantit le pilotage du SI afin d'assurer l'accomplissement des objectifs de l'entreprise et d'accroître ses intérêts [Marks, 2008]. À ce titre, la gouvernance informatique fait partie intégrante de la gouvernance d'entreprise. Les responsables du système d'information sont alors concernés par la prise des décisions et la spécification des droits, des règles et des responsabilités concernant l'ensemble du SI afin d'accroitre son efficacité [Afshar et al., 2007].

La problématique de la gouvernance SOA est une particularisation de la gouvernance informatique en se focalisant sur les éléments spécifiques à la mise en œuvre de l'architecture SOA

[Bennett et al., 2011]. La Figure 12 permet de positionner la gouvernance SOA par rapport à la gouvernance informatique et à la gouvernance d'entreprise.

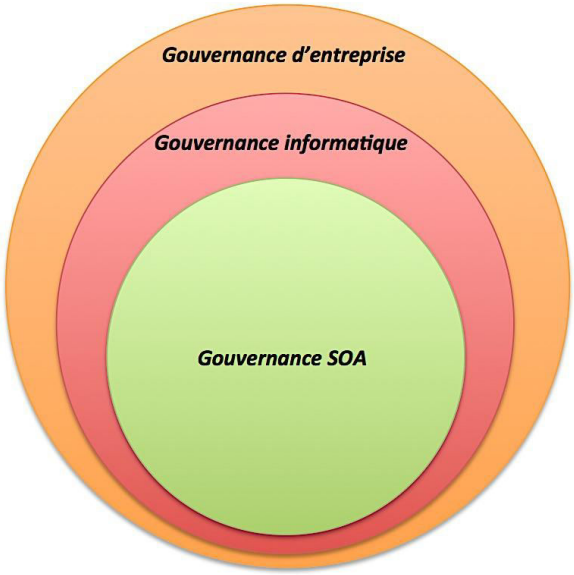

Figure 12 : *Gouvernance d'entreprise, gouvernance informatique et gouvernance SOA.*

La gouvernance SOA joue un rôle fondamental pour la réussite de toute architecture SOA. En effet, le manque de gouvernance peut être un sérieux obstacle et constitue la raison la plus commune de l'échec de cette architecture [Afshar et al., 2007]. Elle intervient sur tous les niveaux du cycle de vie depuis la phase de la conception jusqu'à la phase d'exécution afin de permettre aux entreprises de bénéficier pleinement de l'architecture SOA [Derler et al., 2007] et [Matsumura, 2007].

Les auteurs, dans [Papazoglou et al., 2007], définissent la gouvernance SOA comme étant un domaine de recherche majeur pour la conception et l'exécution d'une SOA. Néanmoins, elle n'est pas clairement définie dans la littérature. Nous essayons donc de s'appuyer sur quelques définitions afin de cerner les principes de la gouvernance SOA :

- Dans [Brown et al., 2006], les chercheurs d'IBM définissent la gouvernance SOA comme un sous-ensemble de la gouvernance informatique en attribuant des droits, des politiques et des mesures autour des services, des processus et du cycle de vie d'une SOA. Ceci est dans le but de répondre aux différentes préoccupations telles que la publication des services, la gestion des versions, le financement, la surveillance, la découverte, etc.
- Dans [Afshar et al., 2007], les chercheurs d'Oracle définissent la gouvernance SOA comme une interaction entre les politiques et règles (Quoi?), les responsables (Qui ?) et les processus (Comment ?) afin d'assurer la réussite de l'architecture SOA. La gouvernance SOA permet de veiller au maintien de l'alignement stratégique et de démontrer la valeur

des investissements SOA réalisés. Elle couvre tous les niveaux du cycle de vie : la conception, le développement, le déploiement, et l'exécution des services. Dans le même principe, les chercheurs de SUN [Wheaton, 2007] présentent la gouvernance SOA comme la capacité d'organiser, d'exécuter et de reconfigurer les interactions de services dans une architecture SOA.

- Dans [Marks, 2008], l'auteur regroupe plusieurs définitions de la gouvernance SOA afin de présenter la sienne : la gouvernance SOA comporte la définition, la mise en œuvre et l'exécution d'une approche SOA afin de l'aligner en permanence avec les objectifs métier de l'entreprise. L'exécution de cette stratégie se fait conformément aux directives et aux contraintes définies par un ensemble de principes et de règles.

En se basant sur les définitions précédentes, la gouvernance SOA peut être révélée, comme l'illustre la Figure 13, dans la trilogie suivante :

« SOA Governance is doing the right SOA things the right way (policies) for the SOA stakeholders (decision-makers) »
[Marks, 2008]

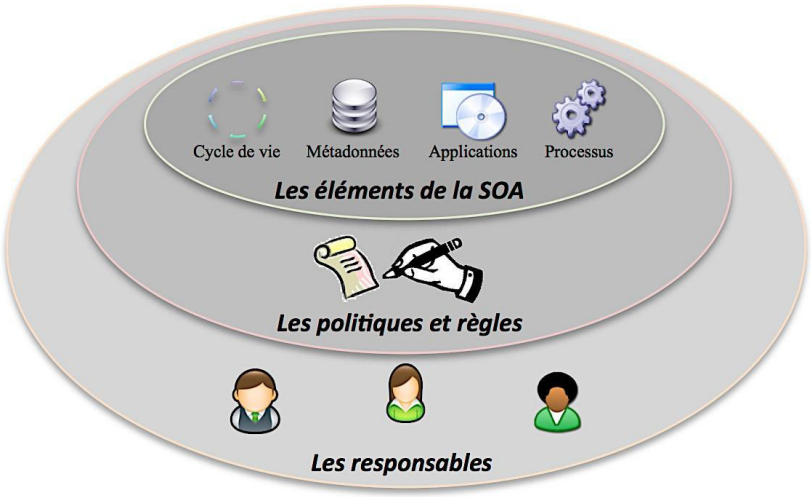

Figure 13 : *Gouvernance SOA selon la trilogie de Marks* [Marks, 2008].

La gouvernance SOA, alors, est la définition, la mise en œuvre, et le contrôle d'un modèle de décisions pour les responsables SOA. Ce modèle s'assure du bon suivi d'une stratégie SOA appropriée et alignée avec les objectifs métier de l'entreprise, et que l'exécution de cette stratégie est conforme aux directives et contraintes définies par un ensemble de politiques et règles pour chacun des éléments de la SOA.

II.3.2. Les cadres de gouvernance SOA

Les entreprises souhaitant migrer leur SI vers une approche SOA, ont besoin de définir et de mettre en place leur propre cadre de gouvernance SOA qui traduit les exigences métier et les transpose dans un ensemble de règles et de normes guidant l'évolution de la SOA. Cette section propose une étude sur des approches et cadres de gouvernance SOA.

Dans [Derler et al., 2007], les auteurs proposent un modèle générique et deux outils pour la gouvernance. Les services sont décrits à partir de leur cycle de vie tout en prenant en compte à chaque étape les rôles des acteurs. Les auteurs identifient trois rôles majeurs : (i) le développeur du service : celui qui se charge de l'implémentation du service, sa structure, son architecture technique, etc. (ii) le chef de produit : détermine les exigences du client, définit les spécifications métier du service, etc. et (iii) l'administrateur : attribue les rôles et les droits à chacun des acteurs. Les deux outils que les auteurs proposent sont : ***Service Repository Console*** et ***Service Browser***. Le premier représente un annuaire contenant les fonctionnalités du service, sa description et ses dépendances. Quant au Service Browser, il est utilisé pour la recherche des services préalablement publiés dans le Service Repository. Cette approche propose une méthode standard et un modèle assez solide pour établir une gouvernance SOA. Néanmoins, les auteurs n'abordent pas la gouvernance dans le cas où les services ont besoin de collaborer ensemble.

Une ***approche méthodologique*** est présentée dans [Schepers et al., 2008] où les auteurs affirment que la gouvernance SOA est plus qu'un processus que met en place l'entreprise mais permet aussi de veiller en permanence à son alignement sur les objectifs stratégiques. Cette approche repose sur six phases : (i) définition d'une stratégie SOA : cette phase vise à aligner le comportement SOA sur les objectifs métier de l'entreprise, (ii) répartition des pouvoirs décisionnels et attribution des responsabilités aux principaux acteurs impliqués dans la gouvernance (les actionnaires, la direction, conseil d'administration, etc.) afin de faire prendre conscience à chacun de ses droits et de ses obligations, (iii) gestion de portefeuilles de services d'une manière à optimiser la réutilisation des services existants avant de développer des nouveaux, (iv) gestion et contrôle complets du cycle de vie de services (de la conception à la production des services), (v) intégration de procédures de suivi et vérification : cette phase permet de s'assurer de la conformité du déroulement des services par rapport aux règles et politiques préalablement définies et enfin (vi) gestion du contrat de service : elle spécifie les accords établis entre le fournisseur et celui qui va l'utiliser (le temps de réponse, le prix, les pénalités en cas de non conformité, etc.). Dans cette approche, si les auteurs présentent des directives assez claires sur la mise en place d'une gouvernance SOA, aucun détail n'est donné sur sa réalisation.

L'approche d'IBM est présentée dans [Brown et al., 2006] comme une approche de cycle de vie de gouvernance en quatre phases. Il se compose d'(i) une phase de planification au cours de laquelle les besoins sont établis, et les procédures et mécanismes existants sont évalués afin de voir comment

les améliorer, (ii) une phase de définition, pendant laquelle le cadre de la gouvernance SOA est défini en incluant les structures organisationnelles, les rôles et les règles à mettre en place durant tout le cycle de vie des services, (iii) une phase de validation durant laquelle le cadre de gouvernance retenu est mis en place au sein de l'entreprise et enfin (iv) une phase de déploiement, de suivi et de vérification du processus de la gouvernance où les politiques et les mécanismes mis en place sont contrôlés afin de vérifier leur respect par rapport aux accords engagés dans les contrats de services et d'analyser l'efficacité des mesures établies, etc. S'appuyant sur ce cadre, IBM propose une solution de gouvernance SOA appelée ***IBM Web Sphere Service Registry and Repository*** [WebSphere, 2010]. Cet outil de gouvernance offre, entre autres, les fonctionnalités de découverte et de publication de services, la gestion du cycle de vie, l'interfaçage avec leur infrastructure de déploiement, etc.

WSO2 Governance Registry [WSO2, 2010] est un outil de gouvernance SOA open-source de l'entreprise WSO2. Il permet l'intégration des fonctionnalités avancées de la gouvernance telles que la publication et la recherche des services, la gestion du cycle de vie, la gestion des dépendances et des relations entre les services, la gestion des utilisateurs et du contrôle d'accès aux services, la gestion du cycle de vie et la gestion de la gouvernance à la phase de l'exécution. Le registre de gouvernance WSO2 permet le stockage, la gestion des métadonnées liées aux services et la gestion des processus métier. Il offre également les fonctionnalités de gestion du cycle de vie, de gestion des politiques et règles, de gestion des relations et de gestion dépendances. Enfin, WSO2 fournit des fonctionnalités avancées comme la possibilité de gérer le registre de gouvernance à distance grâce à l'utilisation du protocole APP[2] (*Atom Publishing Protocol*).

CentraSite™ [CentraSite, 2010] est une solution de gouvernance SOA co-développée par l'entreprise Software ^{AG} et Fujitsu pour soutenir l'architecture SOA. Elle se base sur des standards, garantit une qualité constante, maintient les services en ligne avec les besoins stratégiques de l'entreprise et optimise la réutilisation des ressources existantes. Ce registre de gouvernance offre d'une part un contrôle et une visibilité métier pour la SOA et d'autre part la gestion de la découverte et de la publication de services et des processus, la gestion des versions, la gestion du cycle de vie des services, la gestion des règles et des processus, et l'analyse des dépendances. Cette offre de gouvernance SOA apporte une plus grande agilité métier et une flexibilité technique dans la mesure où elle permet l'intégration avec d'autres infrastructures techniques. Toutefois, elle ne couvre pas certaines fonctionnalités essentielles pour la gouvernance telles que la gestion des contrats de services.

HP SOA Systinet [Systinet, 2009] est un cadre de gouvernance SOA implémenté par l'entreprise Hewlett-Packard (HP). Il fournit un registre permettant de cataloguer et de rechercher les données concernant les services et les processus métier. Il permet aussi la gestion des dépendances entre les

[2] Atom Publishing Protocol (APP) est un protocole de publication basé sur http et permet de gérer les ressources Web.

services individuels et entre les compositions de services. De même que la majorité des cadres de gouvernance SOA, Systinet couvre la fonctionnalité de la gestion du cycle de vie.

Petals Master [Master, 2009] est un outil de gouvernance SOA du consortium OW2[3] développé par l'entreprise EBM WebSourcing (Petals Link). C'est un cadre open source conçu pour gérer les services, les indexer, les documenter, les retrouver, gérer leur cycle de vie, les dépendances entre eux, leurs évolutions et leurs versions. Petals Master peut être déployé en tant qu'un outil autonome ou peut être intégré dans une infrastructure de services. Toutefois, Petals Master ne couvre pas les fonctionnalités de gestion des contrats de services et des propriétés non-fonctionnelles.

Mule Galaxy [Galaxy, 2009] est une plateforme de gouvernance SOA développée par l'entreprise Mule Soft. Elle fournit un registre de gouvernance qui facilite la gestion de la SOA en proposant des fonctionnalités telles que la gestion du cycle de vie, la recherche de services, le stockage des documents décrivant les services, la gestion des changements de version, la gestion de la dépendance, la découverte de services, la gestion des artefacts et le déploiement des services. Bien que Mule Galaxy soit un produit prometteur, il reste similaire à d'autres cadres de gouvernance open source et peut être amélioré par la gestion des propriétés non-fonctionnelles et la par la réconciliation des services.

Il existe d'autres approches et cadres de gouvernance SOA sur le marché, nous pouvons citer : Cosminexus (l'outil de Hitachi, [Cosminexus, 2010]), et Governance Interoperability Framework (GIF, [GIF, 2010]). Ils fournissent généralement les fonctionnalités de découverte de services, de publication de services et de gestion du cycle de vie.

II.3.3. Synthèse sur les cadres existants

Plusieurs cadres de gouvernance SOA existent et particulièrement dans la communauté industrielle et offrent majoritairement les fonctionnalités de gouvernance, telles que : la publication de services (enregistrement de la description du service dans le registre de gouvernance), la découverte de services (la recherche d'un service dans le registre parmi ceux publiés), la gestion des contrats de services (le contrat est un document réalisé par le fournisseur et l'utilisateur et spécifie les accords établis entre eux pour l'utilisation du service), la gestion de cycle de vie (la gestion du service durant toutes les phases de son cycle de vie), la gestion des utilisateurs et des droits d'accès (la gestion du qui fait quoi) et le déploiement (la synchronisation du registre de gouvernance directement avec une infrastructure de déploiement de services). Dans le Tableau 1, nous présentons une analyse comparative de ces cadres. Chaque colonne représente un cadre et chaque ligne une fonctionnalité.

[3] OW2 est une association internationale à but non lucratif dédiée au développement d'intergiciels Open Source de qualité industrielle. Elle regroupe des entreprises et des organismes de recherche de premier plan tels que l'INRIA, Bull, France Télécom, Linagora, Thales Group ou Red Hat.

Rappelons notre objectif initial qui est la rationalisation de la gestion des services et de leurs propriétés non-fonctionnelles. Ceci dans le but d'assurer et d'optimiser la réconciliation non-fonctionnelle entre les activités métier et les services techniques à partir d'un processus modélisé graphiquement. Les deux dernières lignes montrent qu'aucun cadre, parmi ceux étudiés, ne couvre ni la gestion des propriétés non-fonctionnelles (d'une façon assez complète et en utilisant un standard) ni la réconciliation non-fonctionnelle. La dernière colonne présente, notre vision d'un cadre de gouvernance, appelé EasierGov-NFR permettant à la fois de gérer :

- la publication de services ;
- la découverte de services ;
- la gestion des contrats de services ;
- la synchronisation avec un bus de services pour assurer le déploiement ;
- la gestion des propriétés non-fonctionnelles en se basant sur le standard WSQF ;
- la liaison avec un outil de modélisation graphique de processus métier collaboratifs (comprenant l'annotation non-fonctionnelle) afin d'assurer la réconciliation métier / technique.

Tableau 1 : *Synthèse sur les cadres de gouvernance SOA.*

	Web Sphere Service Registry and Repository	WO2 Governance Registry	CentraSite	Systinet	Petals Master	Mule Galaxy	EasierGov-NFR
Publication de services	X	X	X	X	X	X	X
Découverte de services	X	X	X	X	X	X	X
Gestion des versions			X	X	X	X	
Gestion des dépendances		X	X				
Gestion du cycle de vie	X	X	X	X	X	X	
Gestion des contrats de services	X						X
Gestion des utilisateurs et des droits d'accés	X					X	
Déploiement	X				X		X
Gestion des propriétés non-fonctionnelles (en se basant sur un standard)							X
Réconciliation non-fonctionnelle activités métier / services							X

II.4. Gestion des processus métier (BPM)

Dans cette section, nous présentons les différents concepts de base liés aux processus métier, à leur gestion ainsi qu'à leur modélisation graphique.

II.4.1. Les processus et les processus métier

Processus : le terme processus peut se traduire comme « avancer » et « améliorer ». Ce concept est utilisé dans plusieurs domaines. Dans les domaines industriels, Morley dans [Morley et al., 2005] définit un processus comme étant « *l'organisation d'un ensemble finalisé d'activités effectuées par des acteurs et mettant en jeu des entités* ». Théroude [Théroude, 2002] enrichit cette définition en mettant en évidence, d'une part, un aspect objectif lié à une finalité industrielle et, d'autre part, un aspect structurel des processus :

> *« Un processus est un enchaînement partiellement ordonné d'exécution d'activités qui, à l'aide de moyens techniques et humains, transforme des éléments d'entrée en éléments de sortie en vue de réaliser un objectif dans le cadre d'une stratégie donnée. »*
> [Théroude, 2002]

L'approche processus est recommandée aux entreprises par la norme ISO9001 (*International Organization for Standardization)* afin de réaliser une démarche de qualité [Brandenburg et al., 2006]. Elle consiste donc à décrire et à concevoir d'une façon plus pragmatique l'organisation de l'entreprise dans le but de satisfaire les besoins de ses clients. Il existe plusieurs types de processus, mais généralement ils sont classés dans trois grandes familles [Cattan, 2000], [Debauch et al., 2004] :

- Processus de pilotage : appelés aussi processus de management ou processus décisionnels et ont pour fonction d'organiser les objectifs de l'entreprise ;
- Processus opérationnels : ont pour but de contribuer directement à la réalisation du produit ou de la prestation : de la phase de détection du besoin client jusqu'à sa satisfaction ;
- Processus de support : appelés aussi processus de soutien et ont pour mission de fournir les périphériques et les ressources (humaines, matérielles, informationnelles, etc.) nécessaires à tous les processus pour accomplir les objectifs métier.

Processus métier : les processus métier constituent de plus en plus la colonne vertébrale des SIs modernes. Ils sont au cœur de la réorganisation des entreprises pour une meilleure intégration de

leurs départements et de leurs personnels [Godart et al., 2009]. Un processus métier consiste à orchestrer des activités qui visent, grâce à leur déroulement, à atteindre l'objectif souhaité. Les activités sont des éléments d'action atomiques qui transforment une ressource d'entrée en une ressource de sortie. Ces activités peuvent (i) être exécutées manuellement, partiellement ou complètement automatisées sans aucune intervention humaine, (ii) être exécutées en parallèle ou de façon séquentielle et (iii) être déclenchées par des événements ou produire des événements [Weske, 2012]. Un processus métier peut être mesuré par des contrôles de conformité mis en œuvre selon deux types de propriétés :

- Les propriétés fonctionnelles mises en place sur les entrées et les sorties du processus ;
- Les propriétés non-fonctionnelles qui sont relatives à la performance du processus..

Un processus métier peut être affecté entièrement à une seule organisation ou peut interagir avec d'autres processus métier appartenant à diverses organisations afin de répondre à un objectif plus global. Dans ce cas, il est appelé : ***processus métier collaboratif*** (ou aussi processus métier inter-organisationnel).

II.4.2. Gestion des processus métier

La gestion des processus métier (*Business Process Management* - BPM) est une discipline collaborative qui consiste à gérer les processus métier d'une entreprise. La gestion des processus métier est définie par le Pr. Van der Aalst comme suit :

> *« La gestion des processus métier (BPM) inclut les méthodes, les techniques et les outils pour modéliser, exécuter, contrôler et analyser les processus opérationnels métier en s'appuyant sur des acteurs tels que les personnes, les organisations, les applications, les documents et toute autre source d'information. »* [Van der Aalst et al., 2003]

Les deux objectifs majeurs de cette approche sont : (i) avoir une meilleure vue globale de l'ensemble des processus métier de l'entreprise et optimiser les interactions entre ses différentes activités voire les automatiser (dans le cas où ceci est possible) et (ii) gérer les processus métier en s'intéressant aux différentes étapes du cycle de vie des processus depuis la modélisation jusqu'à l'établissement du produit final.

Le BPM tend à unifier toutes les activités de l'entreprise. Crusson, dans [Crusson, 2003], définit trois niveaux pour modéliser les processus métier : le niveau métier, le niveau fonctionnel, et le niveau technique. Ceci permet de découpler la modélisation métier et fonctionnelle de la modélisation

technique d'un processus afin de limiter l'impact de toute modification. Toutefois, pour faire évoluer les processus il est nécessaire de suivre l'une de ces deux approches : l'approche *top-down* et l'approche *bottom-up* (cf. Figure 14) [Crusson, 2003].

- L'approche *top-down* est utilisée lorsque le processus est modifié par le niveau métier ou fonctionnel.
- L'approche *bottom-up* est utilisée lorsqu'une modification intervient dans l'implémentation des services techniques. Dans la majorité des cas, une modification à ce niveau n'a pas vraiment d'impacts sur les niveaux métier et fonctionnels.

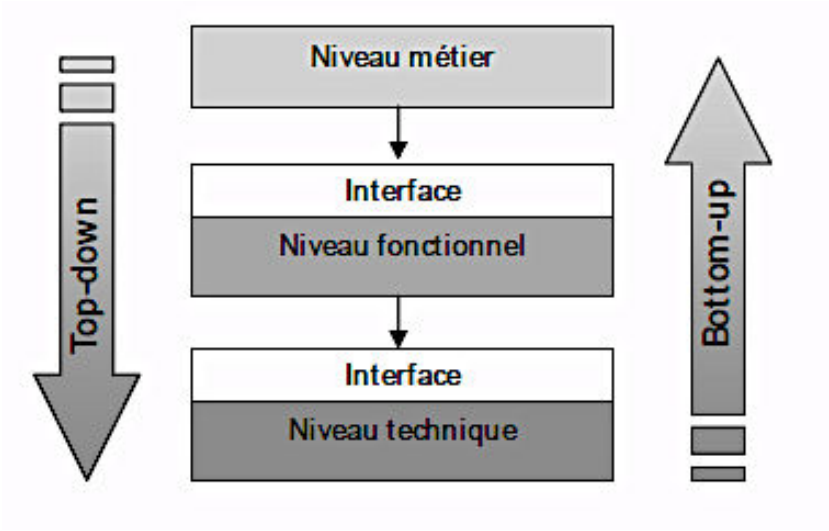

Figure 14 : *Niveaux d'abstraction d'un processus BPM [Crusson, 2003].*

II.4.3. Classification des processus métier

Il existe deux types de classements des processus métier : les processus abstraits et les processus exécutables (connus aussi sous le nom des processus concrets).

- Les processus abstraits constituent une représentation haut niveau des processus métier [Crusson, 2003]. La définition des processus abstraits est indépendante des aspects techniques. En d'autres termes, nous ne précisons pas quelles actions sont effectivement réalisées par les activités (e.g. Comment une demande de devis est reçue ? Quelle application est utilisée pour vérifier la disponibilité des produits ?, etc.). Le processus abstrait est la base utilisée par les équipes techniques pour créer un processus exécutable ;
- Les processus exécutables sont obtenus par la spécialisation du processus métier abstrait et se focalisent sur l'implémentation privée et propre à chacun des participants. Ils spécifient un certain nombre d'informations techniques (e.g. les services impliqués dans le processus, la transformation des données effectuées, l'intégration des utilisateurs comme

participants dans le processus, le format des messages échangés, les protocoles de transport utilisés, etc.).

II.5. BPM dans un contexte SOA

Pour rester compétitive, une organisation doit pouvoir analyser l'efficacité de ses processus métier et les améliorer sans être contrainte par le système d'information sous-jacent. Le BPM ne peut suffire à lui seul pour créer un système d'information agile. Il ne se préoccupe pas de lier les processus métier avec le système d'information (certains processus métier sont d'ailleurs intrinsèquement décorélés). C'est là qu'intervient l'architecture SOA. En faisant évoluer le système d'information d'un paradigme d'applications à un paradigme de services, l'architecture SOA offre une interface parfaite pour le BPM. Elle permet d'associer les étapes des processus métier à des appels de services, sans se soucier de l'implémentation effective de ces derniers. En effet, les processus BPM assurent l'interopérabilité au niveau métier, tandis que l'utilisation des standards et des services Web réutilisables sont préconisés par la SOA afin d'assurer l'interopérabilité au niveau technique. Cette universalité facilite la composition de plusieurs services hétérogènes permettant de répondre à un objectif commun. Dans cette section, nous présentons les approches pour la description de la composition de services ainsi que les modèles et langages pour leur représentation graphique au niveau métier.

II.5.1. Composition de services Web

La composition de services Web est l'un des défis majeurs du paradigme émergent SOA. Elle permet de créer de nouvelles fonctionnalités en faisant collaborer les fonctionnalités des services Web existants afin de répondre à une finalité commune et d'apporter une valeur ajoutée. La modélisation d'une telle technique dans un flot de processus métier peut se faire principalement selon deux approches : (i) une approche statique et centralisée : l'*orchestration*, ou (ii) une approche dynamique et complètement décentralisée : la *chorégraphie*.

Orchestration : l'orchestration de services Web définit un schéma de services abstraits qu'un moteur d'orchestration (appelé aussi moteur de composition) instancie et exécute en fonction des services concrets disponibles. Ce moteur d'orchestration coordonne et contrôle l'exécution des différentes opérations sur les services participants. Par conséquent, l'orchestration est centralisée et définit d'une façon explicite les opérations, l'ordre d'invocation des services Web, ainsi que les interactions [Juric et al., 2006]. Ces interactions peuvent être inter-applications et/ou inter-organisations, et peuvent résulter en un modèle d'exécution transactionnel de longue durée. BPEL (*Business Process Execution Language*) [OASIS, 2007] est reconnu comme un langage standard de l'orchestration de services.

Chorégraphie : contrairement à l'orchestration, la chorégraphie ne dépend pas d'un orchestrateur central. Elle décrit la collaboration entre une collection de services afin d'atteindre un objectif commun. La chorégraphie est basée sur la collaboration et est principalement utilisée pour l'échange de messages entre plusieurs services (potentiellement appartenant à des partenaires hétérogènes) au sein des processus métier publics [Peltz, 2003]. Tous les services Web mêlés dans la chorégraphie, doivent connaître le processus métier, les opérations à exécuter, les messages à échanger, ainsi que le moment de diffusion des messages. WS-CDL (*Web Services Choreography Description Language*) [W3C, 2005] est l'un des langages recommandés pour la description de chorégraphie de services.

II.5.2. Modélisation graphique des processus métier orientés SOA

Une modélisation précise du processus est primordiale car c'est d'elle que dépend tout le déroulement du processus tel qu'il est voulu par l'auteur du modèle. Il existe de nombreux types de modélisation des processus métier mais dans ces travaux nous nous intéressons particulièrement à la modélisation graphique des processus métier. Dans ce qui suit, nous présentons quelques modèles et langages de ce type de modélisation.

Les réseaux de Petri (*Petri nets* - PN) ont été présentés par Carl Adam Petri [Petri, 1962]. Ils représentent une notation graphique plus formelle et plus abstraite que toutes les autres notations [Murata, 1989]. Elle est définie sous la forme d'un flux d'informations qui est décrit, analysé, et contrôlé dans les systèmes où les activités sont asynchrones et concurrentielles, distribuées, parallèles, non déterministes et/ou stochastiques. Cette notation graphique permet également de mettre en place des équations d'état, des équations algébriques et d'autres modèles mathématiques qui gèrent le comportement d'un système. L'exemple de la Figure 15 illustre une modélisation graphique d'un processus à l'aide des réseaux de Petri [De Backer et al., 2005]. La transition A renvoie le déclenchement en parallèle des transitions B ou NOT B, et C ou NOT C (selon la vérification de l'équation). La fin du déroulement de ces transitions constitue le début de la transition D.

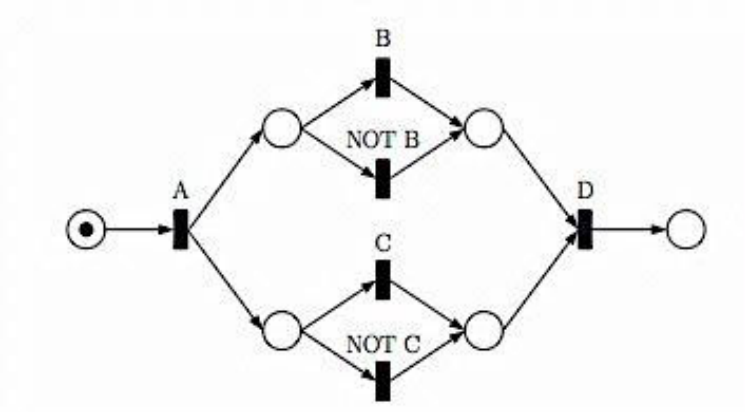

Figure 15 : *Modélisation de processus avec les réseaux de Petri* [De Backer et al., 2005].

Toutefois, la modélisation de processus métier avec les réseaux de Petri n'inclut pas une représentation organisationnelle (les acteurs) ni une représentation de la prise en considération des évènements qui peuvent être déclenchés au cours de l'exécution du processus.

IDEFØ - *Icam (Integrated computer-aided manufacturing) DEFinition Ø* connu aussi sous le nom commercial SADT (*Structured Analysis and Design Technique*) est une méthode orientée flux et conçue pour modéliser graphiquement les processus métier. Cette méthode a connu une diffusion importante dans les milieux industriels et est centrée sur la représentation des activités d'une organisation ou d'un système [Greeff et al., 2004]. Elle consiste en la décomposition du processus d'une façon hiérarchique. La représentation du diagramme global, appelé aussi diagramme de contexte, se traduit par une boite (l'activité) et les quatre flèches situées à ses points cardinaux : les flux entrants (à gauche), les flux sortants (à droite), les contraintes de contrôle (en haut) et les mécanismes (en bas). Ce diagramme peut être décomposé en plusieurs diagrammes « enfants », chacun correspond aux détails d'une boite [Bia-Figueiredo et al., 2011]. Dans la Figure 16, nous présentons un exemple de diagramme de modélisation avec IDEFØ. Le diagramme principal, nommé A-0, représente le processus global. Le diagramme du premier niveau de décomposition (A0) est composé d'une suite d'activités. Seule l'activité 3 a fait l'objet d'une deuxième décomposition (d'où la mention du numéro A3 sous la boite 3). Cette méthodologie a le mérite d'être bien structurée mais elle fournit des graphiques à l'aspect un peu technique, visuellement pas assez faciles de compréhension et très clairement orientés sur les flux de données.

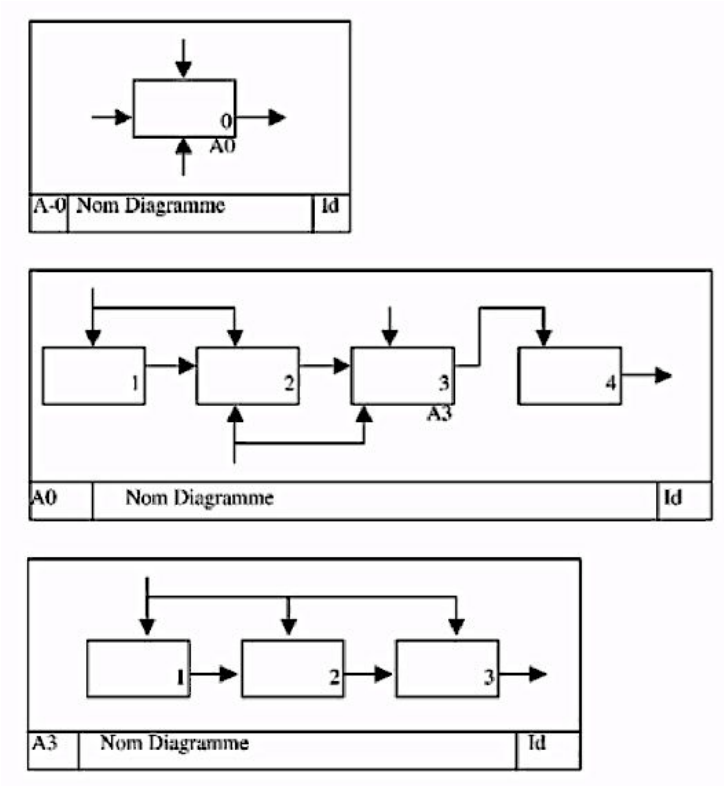

Figure 16 : *Modélisation de processus avec IDEFØ* [Bia-Figueiredo et al., 2011].

UML *Unified Modeling Language* [Booch et al., 1998] propose un ensemble de diagrammes spécifiques, tels que les diagrammes d'activités, les diagrammes de séquences, les diagrammes de classes, etc. Pour la modélisation des processus métier, UML propose un mécanisme d'extensibilité permettant de spécialiser les diagrammes d'activités. IBM propose un profil UML appelé « *UML2 Activity Diagram* » permettant de prendre en compte une telle extension [Koehler, et al., 2007]. La Figure 17 illustre un exemple de processus modélisé à l'aide de ce profil. Le processus commence par un flux de début (nœud gris) et un flux de fin (nœud blanc contenant une croix). L'élément qui représente une activité est appelé *Call Behavior* et est représenté par un rectangle. La *Decision* représente un état exclusif (seule l'activité qui vérifie la condition s'exécute). Les *Fork* et *Join* indiquent respectivement le début et la fin d'une exécution parallèle des activités (Subprocess 1 et Task2). Le petit rectangle gris au début du branchement des activités 3 et 4 définit le branchement inclusif (Task 3, Task 4, ou les deux qui s'exécutent).

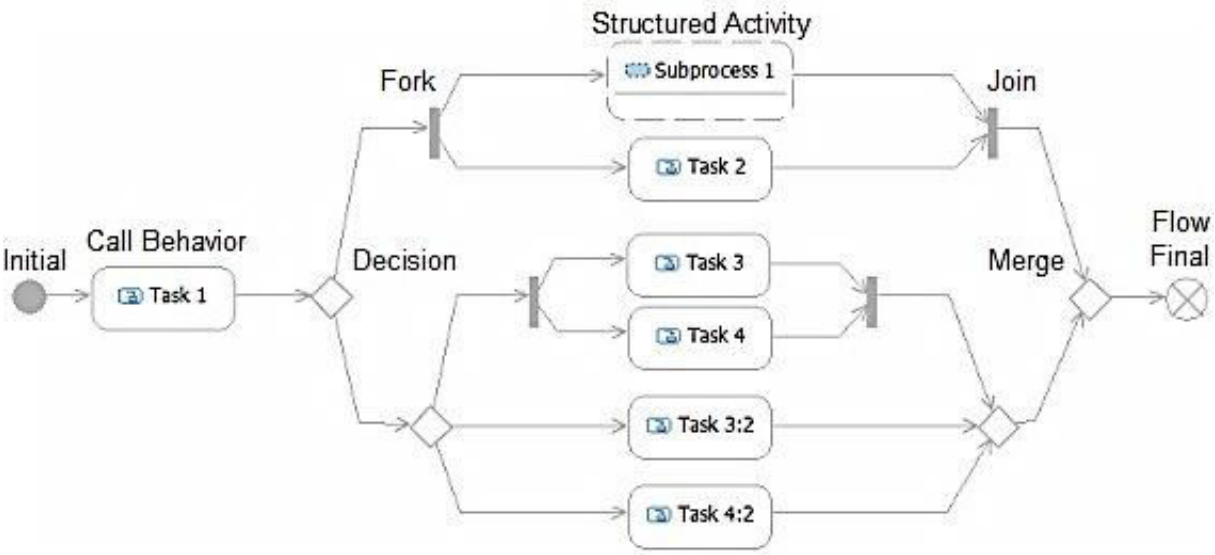

Figure 17 : *Modélisation de processus avec le profil UML* [Koehler, et al., 2007].

Cette approche est propriétaire à l'équipe IBM et consiste à définir un profil UML pour chaque outil. Toutefois, cela ce fait en dépit de l'interopérabilité puisqu'il devient impossible dans un outil de travailler sur un processus modélisé avec un autre outil.

BPMN *Business Process Model and Notation* [Stephen, 2004] est un standard développé par le groupe BPMI (*Business Process Management Initiative*) et intégré par la suite par OMG (*Object Management Group*). Il vise à fournir une représentation graphique normalisée et complète des processus métier. La dernière version de BPMN a été publiée en Février 2011 sous le nom de BPMN 2.0 [OMG, 2011]. Cette nouvelle version est devenue le standard principal pour la modélisation des processus métier comme elle permet non seulement de modéliser une très grande variété de processus métier, mais aussi de modéliser les diagrammes qui sont à la fois faciles à comprendre et peuvent être

exécutables. Ainsi, il comble le passage entre la modélisation des processus métier et leur exécution automatique. La modélisation graphique est au cœur de la spécification BPMN 2.0 qui propose un ensemble d'éléments assez intuitif et permet de gérer les évènements qui peuvent être déclenchés au cours de l'exécution du processus. BPMN 2.0 permet également de modéliser deux types d'activités : (i) les activités dites interactives (qui nécessitent une intervention humaine pour leur réalisation) et les activités automatiques (qui leur exécution n'a besoin d'aucune intervention). Dans la Figure 18, nous présentons un exemple de modélisation avec BPMN 2.0 d'un processus métier de contrôle de tweets avant leur publication. La soumission d'un tweet déclenche le début du processus, si le responsable de communication décide que le contenu du tweet est conforme aux politiques de l'entreprise (tâche humaine), il est donc publié par le système (tâche automatique), sinon des modifications sont demandées. Le premier nœud (à gauche) constitue l'événement du début du processus (déclenché pour chaque soumission de tweet), le nœud de droite constitue la fin du processus. Les rectangles avec des coins arrondis symbolisent les activités et l'indicateur en haut à gauche définit le type de l'activité : humaine (un être humain) ou automatique (un rouage). Enfin, le losange représente le branchement, dans cet exemple il s'agit d'un branchement exclusif.

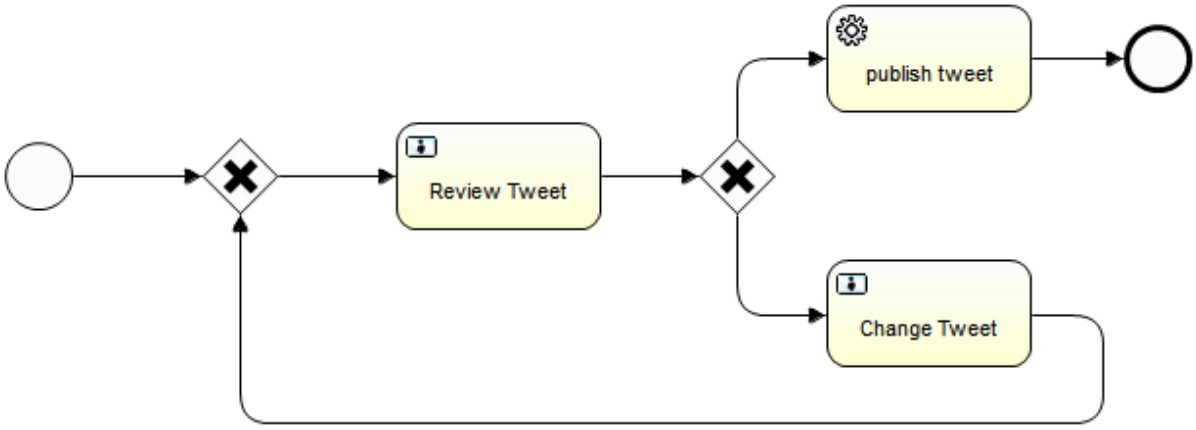

Figure 18 : *Modélisation de processus avec BPMN 2.0* [Meyer, 2011].

Par ailleurs, pour assurer l'interopérabilité, chaque diagramme BPMN 2.0 correspond à une sérialisation XML unique contenant toutes les informations sémantiques liées au processus métier lui-même. Une telle sérialisation unique pour la modélisation du processus métier permet son utilisation dans d'autres outils de modélisation. BPMN 2.0 fournit également un standard appelé BPMN DI (*BPMN Diagram Interchange*) qui est basé aussi sur une sérialisation XML [Bray et al., 1997]. Ce standard est utilisé pour compléter la notation textuelle de BPMN 2.0 en ajoutant des informations graphiques. De plus, il permet d'importer des diagrammes de type BPMN 2.0 directement dans un outil de modélisation graphique s'ils respectent les règles définies dans la spécification. De plus, le standard BPMN 2.0 est compatible avec des langages basés sur XML pour l'exécution des processus

et fournit une connexion directe avec le langage standard BPEL (*Business Process Execution Language*) pour une exécution automatique des processus.

II.5.3. Synthèse sur la modélisation des processus métier orientés SOA

Les processus métier occupent une place centrale dans la mise en œuvre et l'évolution des SIs. Ils se présentent comme un outil clé à la compréhension du fonctionnement de l'entreprise d'une part, et d'autre part à l'organisation des activités et l'amélioration des interactions entre elles.

La modélisation des processus métier représente une étape essentielle à travers laquelle les concepteurs définissent les nouveaux processus métier d'une façon abstraite ou détaillée, ou améliorent des processus déjà existants.

Nous avons défini dans cette section, les langages de modélisation graphique du BPM les plus utilisés : les réseaux de Petri, IDEF0, UML et BPMN 2.0.

Les deux premiers prennent en compte seulement une représentation de la gestion comportementale du processus. La modélisation des processus métier est donc réduite à une simple séquence d'activités.

Les deux derniers (à savoir, UML et BPMN 2.0) représentent des langages plus avancés qui incluent une représentation informationnelle (les données) et organisationnelle (les acteurs). Cette catégorie fournit une typologie riche pour modéliser les activités, les événements, les flux, etc. L'utilisation de ces formalismes avancés lors de la modélisation des processus métier permet de couvrir les différents aspects et concepts des processus. La principale différence entre ces deux langages est que BPMN 2.0 offre la possibilité d'être lié directement à un environnement d'exécution (et ce par exemple grâce au standard BPEL). Tandis que pour UML, il est nécessaire de créer une correspondance entre la représentation graphique des processus et un langage d'exécution afin de permettre un déploiement direct des processus modélisés.

En conséquence, BPMN dans sa version la plus récente (2.0) semble être le meilleur choix pour la modélisation graphique des processus métier.

II.5.4. Annotation du BPMN avec des exigences non-fonctionnelles

L'importance des exigences non-fonctionnelles (appelées aussi contraintes) pour les processus métier a été fortement reconnue durant ces dernières années. Plusieurs définitions d'exigences non-fonctionnelles existent, mais nous retenons celle présentée par Chung :

> *« Non-functional requirents describe not what the service will do, but how the service will do it, for example, software performance requirements, software external interface requirements, design constraints, and software quality attributes. »* [Chung et al., 2000]

Les exigences non-fonctionnelles (*Non-Functional Requirements* - NFR) ne décrivent pas ce que le service fait (les fonctionnalités), mais plutôt comment il les remplit. Dans la littérature, il existe de multiples travaux autour de la modélisation des exigences non-fonctionnelles dans les processus métier ([Mylopoulos et al., 1992], [Chung et al., 1995], [Bresciani et al., 2002] et [Lu et al., 2006]). Toutefois, ceux qui traitent particulièrement les processus métier de type BPMN 2.0 sont beaucoup plus limités même s'il existe plusieurs ouvrages abordant ce sujet ([Demirors et al., 2003], [Cysneiros et al., 2004], [Gorton et al., 2006] et [Hepp et al., 2007]).

Bien qu'il y ait une couverture quasi complète des caractéristiques fonctionnelles lors de la modélisation des processus métier, les exigences non-fonctionnelles ne sont généralement pas prises en considération. Cela peut conduire à la non prise en compte (voire à l'exclusion) d'informations importantes et particulièrement dans un contexte des architectures SOA où les processus sont amenés à collaborer. Les exigences non-fonctionnelles ne sont pas explicitement traitées par le processus BPMN. L'approche envisagée est plutôt d'étendre le formalisme en associant une approche ou un modèle pour annoter les activités du processus métier.

Les auteurs dans [Rodriguez et al., 2007] présentent une solution d'annotation du BPMN à l'aide d'exigences non-fonctionnelles. Ils proposent dans leur approche deux méta-modèles : (i) un méta-modèle appelé *Business Process Diagram* (BPD) qui reprend les différents éléments du processus BPMN et détaille les relations entre eux et (ii) un méta-modèle de sécurité (*Security Requièrent*) qui étend le BPD en intégrant les aspects de sécurité suivants : la non-répudiation (*non-repudiation*), la détection des risques et des attaques (*attack harm detection*), l'intégrité (*integrity*), la confidentialité (*privacy*) et le contrôle d'accès (*access control*). La Figure 19 permet d'illustrer ces deux méta-modèles et montre la relation entre eux.

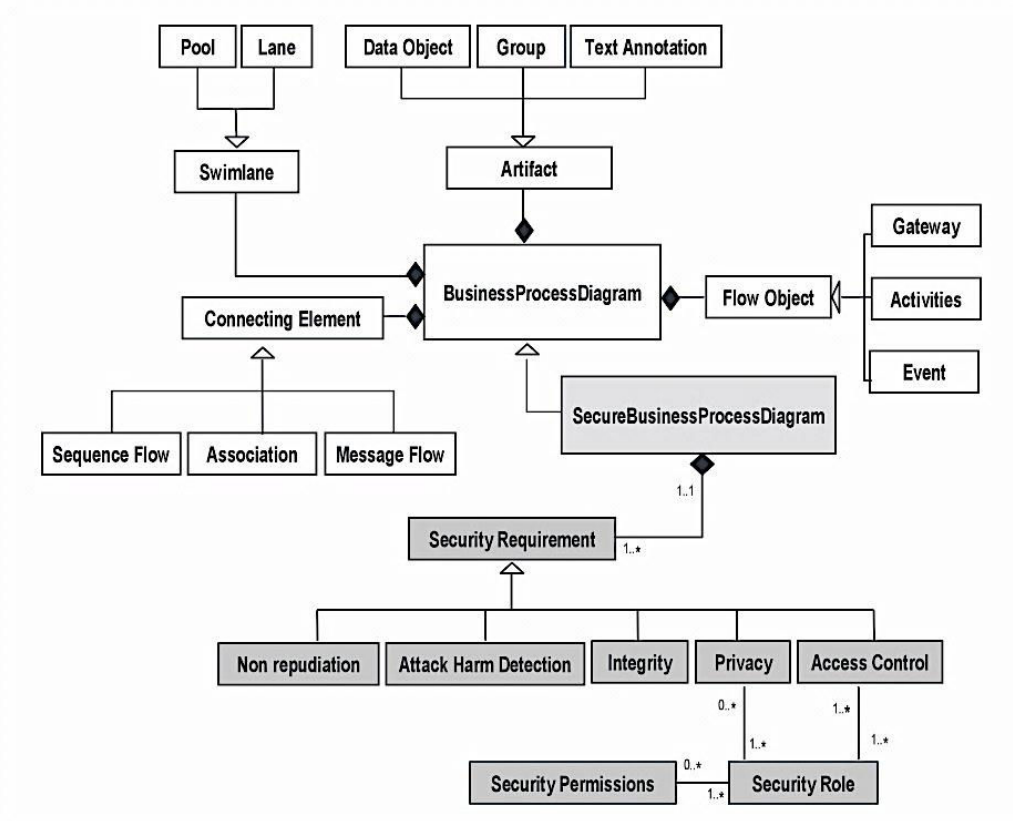

Figure 19 : *Extension de BPMN : intégration des exigences de sécurité [Rodriguez et al., 2007].*

Cette approche permet aux analystes métier d'exprimer les contraintes de sécurité à partir de leur perspective. Cependant, elle se limite qu'à la modélisation de la sécurité. Certes, ces propriétés sont assez importantes et particulièrement dans un contexte collaboratif. Mais, ceci n'est pas suffisant pour couvrir les autres exigences des utilisateurs telles que le temps de réponse, le prix et la disponibilité qui sont également du même ordre d'importance.

Les auteurs dans [Pavlovski et al., 2008] présentent une approche qui permet d'étendre la modélisation d'une activité d'un processus BPMN en lui associant des contraintes non-fonctionnelles. Pour réaliser ceci et comme l'illustre la Figure 20, ils proposent deux nouveaux artefacts « *operating condition* » et « *control case* » :

- état de fonctionnement (*operating condition*) : cet artefact est associé directement à une activité du processus métier. Il permet d'indiquer les exigences non-fonctionnelles qui sont associées à l'activité en question. Ces exigences peuvent être : la performance, les politiques de sécurité, la disponibilité et la fiabilité. Cette extension représente une vue de haut niveau sur les contraintes potentielles à définir ;
- cas de contrôle (*control case*) : cet artefact est en outre optionnel et est associé à *l'operating condition*. Il consiste en un ensemble d'informations détaillées concernant les

contraintes non-fonctionnelles. Il permet également de définir les contrôles à mettre en place afin d'atténuer les risques métier associés aux contraintes non-fonctionnelles.

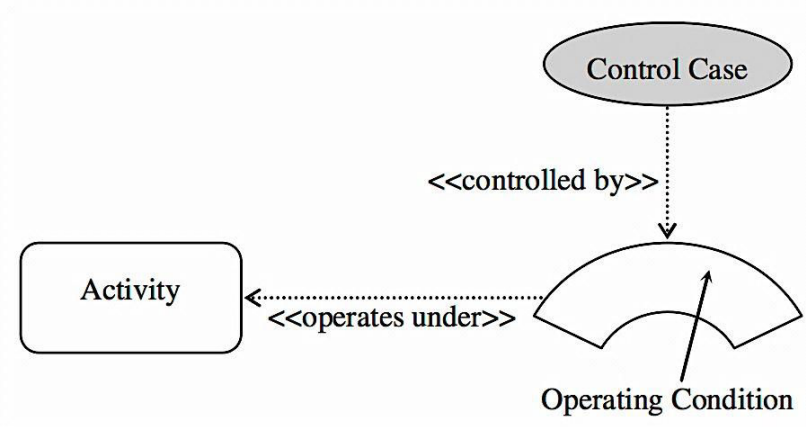

Figure 20 : *Extension de BPMN : control case et operating condition [Pavlovski et al., 2008].*

Toutefois, la liste des contraintes non-fonctionnelles que les auteurs proposent dans *operating condition* est de très haut niveau. Ils ne spécifient pas, par exemple, quelles propriétés non-fonctionnelles sont couvertes par les politiques de sécurité (le cryptage, l'authentification, la non répudiation, etc.). De plus, si cette approche présente une solution d'extension du processus BPMN en permettant l'annotation des activités métier par des contraintes non-fonctionnelles, elle ne présente pas de méta-modèle permettant de modéliser ces contraintes.

Les travaux d'Heinrich [Heinrich et al., 2011] optent également pour l'extension du standard BPMN en lui considérant la gestion des contraintes non-fonctionnelles. Ils présentent dans leur approche un méta-modèle intitulé BPQRM (*Business Process Quality Reference-Model*). Ce méta-modèle permet d'associer un large ensemble de propriétés non-fonctionnelles à chacun des composants du processus métier (contexte (*context*), activité (*activity*), ressource (*resource*), acteur (*actor*), informations sur les objets (*information object*), données des objets (*data object*)) et les relations entre eux. La Figure 21 présente le méta-modèle BPQRM.

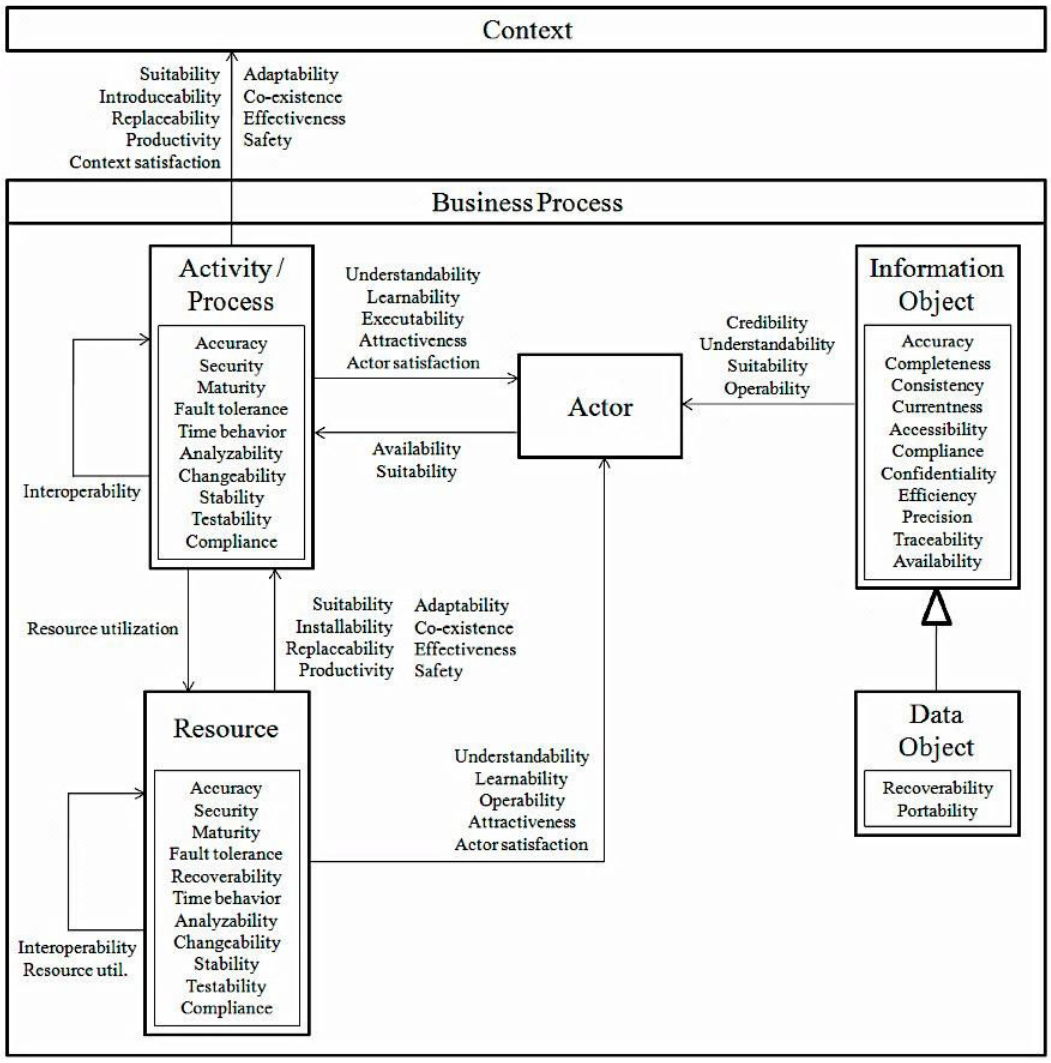

Figure 21 : *Extension de BPMN : le méta-modèle BPQRM [Heinrich et al., 2011].*

Bien que BPQRM identifie plusieurs propriétés non-fonctionnelles, ce méta-modèle ne représente pas un standard et ne fait pas référence également à l'utilisation des standards (entre autres pour une question de terminologie). Or, implémenter des standards est particulièrement recommandé dans le sens où les processus sont amenés à collaborer avec d'autres provenant de différentes entreprises et réalisés à partir des technologies hétérogènes.

II.5.5. Synthèse sur les annotations du BPMN à l'aide d'exigences non-fonctionnelles

La prise en compte des exigences non-fonctionnelles au sein de la modélisation des processus métier est essentielle pour satisfaire au mieux les attentes des utilisateurs. Il est donc utile d'avoir une notation graphique simple, facile et complète qui permettra de gérer cet ensemble de concepts lors de la modélisation graphique du processus. Le standard BPMN 2.0 nous offre la possibilité de gérer quelques uns parmi ces concepts (par exemple la sécurité) mais à un niveau d'abstraction qui, à notre

avis, n'est pas suffisant. En effet, l'approche BPMN 2.0 ne considère pas explicitement l'annotation d'une activité métier par des exigences non-fonctionnelles.

Les travaux de [Rodriguez et al., 2007], [Pavlovski et al., 2008], et [Heinrich et al., 2011] présentés précédemment décrivent des approches qui étendent la modélisation des processus métier BPMN en prenant en considération les exigences non-fonctionnelles. Si ces travaux varient considérablement en complexité et en exhaustivité, ils n'implémentent pas d'une manière complète les standards d'annotation dédiés. Or, ceci est essentiel pour assurer l'interopérabilité. Cette dimension prend plus d'importance lorsqu'elle est appliquée dans un contexte collaboratif où les processus sont amenés à communiquer avec d'autres provenant de différentes entreprises, pour accomplir une finalité commune globale.

II.6. Réconciliation non-fonctionnelle

Avec la prolifération des services Web, il est devenu d'une nécessité patente de différencier les services Web qui répondent aux mêmes fonctionnalités afin de satisfaire au mieux les exigences métier des utilisateurs. Dans cette section, nous présentons quelques approches de la littérature permettant la réconciliation non-fonctionnelle entre les activités métier et les services. Par la suite, nous présentons une synthèse sur ces approches.

II.6.1. Etude des approches existantes

Dans le cadre des architectures SOA, la prise en considération des contraintes non-fonctionnelles devient particulièrement cruciale. En effet, ces contraintes constituent un critère décisif dans le choix de services parmi des offres de plus en plus compétitives et ayant des fonctionnalités quasi similaires. Il existe plusieurs approches dans la littérature qui prennent en compte les exigences non-fonctionnelles lors de la découverte et de la sélection des services.

Guo et al., dans [Guo et al., 2011], présentent une méthode pour la sélection des services selon les propriétés fonctionnelles et les propriétés de qualité de service. Leur approche pour la partie non-fonctionnelle consiste à choisir le meilleur service parmi les services candidats (qui résultent de la réconciliation fonctionnelle et sémantique). Ils proposent un algorithme de sélection en 3 étapes :

- la première consiste à quantifier les attributs de qualité de service selon leurs types : valeurs numériques, valeurs textuelles, valeurs booléennes, intervalles, etc. ;
- la deuxième étape consiste à normaliser les valeurs de chaque service candidat selon le type de la valeur de la propriété non-fonctionnelle. Les auteurs proposent une normalisation par rapport à la valeur maximale (ou à la valeur minimale selon le sens de variation de la variable) de chacune des propriétés de l'ensemble des services candidats ;

- la troisième et dernière étape consiste à calculer le degré pondéré de la QoS pour chacun des services candidats afin de les classer selon un ordre décroissant.

Les auteurs, dans la méthode qu'ils proposent, ont traité la majorité des types des propriétés non-fonctionnelles. Cependant, ils ne prennent pas du tout en considération les valeurs des propriétés non-fonctionnelles souhaitées par l'utilisateur. Leur réconciliation non-fonctionnelle consiste, en effet, à un filtre destiné à classer les services candidats entre eux selon seulement les valeurs de poids attribuées par l'utilisateur. En aucun cas les valeurs cibles pour les critères non-fonctionnels ne sont prises en compte.

Dans [Badr et al., 2008], les auteurs présentent une approche de sélection des services Web basée sur les diverses exigences non-fonctionnelles de l'utilisateur. Les préférences de l'utilisateur sont considérées en tant qu'une entrée supplémentaire vers le système de sélection de services. Ce système, à son tour, classe les services disponibles sur la base des préférences de l'utilisateur. Cette approche se base sur le calcul de la somme pondérée qui modifie les pondérations entre les fonctions objectives pour obtenir le front de Pareto [Kim et al., 2005]. Ils considèrent que le poids lui même reflète la préférence de l'utilisateur. Si l'utilisateur estime que toutes les propriétés sont importantes, alors les pondérations seront réparties d'une manière égale. Ou bien, si l'utilisateur considère que certaines propriétés non-fonctionnelles sont plus importantes que d'autres alors, les poids forts sont attribués à ces propriétés et les autres auront le même poids. Leur approche est simple et facile à utiliser. Toutefois, elle se limite à la sélection unitaire de services (un service Web pour chaque tâche) et n'aborde pas la sélection de composition de services. Cette dernière est essentielle particulièrement quand il n'existe aucun service disponible qui répond aux besoins de l'activité métier. De plus, leur approche utilise le registre de services UDDI, or ce dernier, comme nous l'avons vu précédemment, est très limité pour la gestion des propriétés non-fonctionnelles qui nécessitent d'être gérer durant tout le cycle de vie des services pour une meilleure découverte.

Ran, dans [Ran, 2003], présente également un modèle de découverte de services Web dans lequel les propriétés fonctionnelles sont traitées. L'auteur propose une extension du modèle de base (fournisseur, registre et consommateur) en rajoutant une autre entité appelée *QoS Certifier*. Cette entité vérifie les propriétés de qualité de service fournies par le fournisseur du service avant que le service ne soit publié dans le registre UDDI. Il propose aussi une extension de la structure de données de ce registre afin qu'il prenne en considération les propriétés non-fonctionnelles telles que la disponibilité, la fiabilité, la performance. Une approche similaire est présentée dans [Deora et al., 2003] et se base sur les attentes des utilisateurs. Elle recueille les attentes et les évaluations des utilisateurs d'un service. Par la suite, la qualité de service est calculée uniquement au moment où une demande de service est faite et seulement en utilisant les évaluations qui ont des attentes similaires.

II.6.2. Synthèse sur la réconciliation non-fonctionnelle

De nos jours, les organisations mettent de plus en plus en place des plateformes d'automatisation des processus impliquant un traitement collaboratif des services pour répondre à un besoin commun global. Toutefois, avec l'augmentation du nombre de services Web offrant les mêmes fonctionnalités, le problème de la découverte et de la sélection des services est devenu crucial. Les propriétés non-fonctionnelles (telles que le temps de réponse, la disponibilité, la sécurité, etc.) jouent un rôle essentiel pour différencier les services, ayant les mêmes propriétés fonctionnelles, entre eux.

Nous venons d'étudier quelques approches permettant de filtrer et de classer les services en se basant sur les propriétés non-fonctionnelles. Ces approches sont diverses et même si elles permettent la sélection de services, elles se basent sur le registre de découverte de base UDDI. Cependant, comme nous l'avons vu précédemment (cf. section II.2.2), ce registre est limité et n'assure pas une gestion globale des services et particulièrement la gestion de leurs propriétés non-fonctionnelles. Nous présentons dans le Tableau 2 une synthèse des approches étudiées. La dernière colonne expose notre proposition pour la réconciliation non-fonctionnelle.

Tableau 2 : *Synthèse sur les approches de réconciliation non-fonctionnelle.*

	[Guangjun et al., 2011]	[Badr et al., 2008]	[Ran, 2003]	Proposition
Propriétés non-fonctionnelles des services	QoS top Ontology	NFP Categorization Ontology	QoS Category Model	standard WSQF
Sélection de services	X	X	X	X
Gestions des poids	X	X		X
Extension d' un outil de gouvernance	UDDI	UDDI	UDDI	EasierGov-NFR
Sélection de compositions de services				X
Liaison avec outil de modélisation de processus pour la réconciliation métier / technique				X

II.7. Conclusion

Dans ce chapitre, nous avons introduit les architectures SOA et leur rôle pour améliorer le système d'information de l'entreprise en offrant un couplage lâche et une meilleure agilité. Nous avons aussi présenté un état de l'art sur la modélisation des propriétés non-fonctionnelles des services.

Par la suite, nous avons présenté la gestion des processus métier (BPM) collaboratifs en général avant de détailler leur symbiose avec la SOA. Par ailleurs, nous avons montré les différentes approches de la composition de services (à savoir l'orchestration et la chorégraphie) et comment les propriétés non-fonctionnelles jouent un rôle important dans ces processus (puisqu'elles constituent un élément clé pour la popularité du service). Nous nous sommes intéressés aussi à étudier les différentes approches pour modéliser graphiquement les processus métier lors de la composition et pour les annoter avec des exigences non-fonctionnelles. Dans ce contexte, nous avons, d'une part, montré aussi l'importance de la gouvernance SOA qui représente un registre de services avancé pour une meilleure découverte de services et d'autre part, étudié quelques approches pour la réconciliation et sélection de services selon les contraintes non-fonctionnelles.

Dans les chapitres suivants, nous présentons nos contributions pour l'annotation graphique des activités du processus BPM dans un contexte SOA à l'aide d'exigences non-fonctionnelles et nous détaillons notre approche d'un cadre de gouvernance SOA qui permette la gestion des services et de leurs propriétés non-fonctionnelles pour pouvoir enfin réaliser la réconciliation non-fonctionnelle de services pour les activités annotées de notre processus.