SV3R : un framework pour la gestion de la variabilité des services

Boutaina CHAKIR
Equipe Al Qualsadi, ENSIAS, Université Mohammed V - Madinat Al Irfane, Rabat, Maroc.
chakir.boutaina@um5s.net.ma

Mounia FREDJ
Equipe Al Qualsadi, ENSIAS, Université Mohammed V- Madinat Al Irfane, Rabat, Maroc.
fredj@ensias.ma

Résumé

L'apparition et l'expansion du paradigme de développement basé sur les approches orientées services a créé un besoin de modèles et méthodes de réutilisation de services pouvant permettre de tenir compte de différents contextes d'utilisation. Cette exigence nécessite de munir systématiquement les services de plusieurs réalisations possibles. Une des pistes prometteuses permettant d'atteindre cet objectif est la gestion de la variabilité. Cette dernière a été largement adoptée par de nombreuses disciplines de l'ingénierie logicielle, avec l'objectif de faciliter l'adaptation et la configuration des artefacts de manière systématique. C'est dans ce contexte que s'inscrit ce travail qui propose un cadre méthodologique pour le développement (for et by reuse) de services supportant la variabilité, baptisé «SV3R» *(Service Variability Representation and Resolution for Reuse)*. Cet article introduit tout d'abord le concept de variabilité. Il présente ensuite les différentes approches traitant de la variabilité de service, avant d'exposer une vue globale du framework SV3R et de décrire ses différentes composantes.

Abstract

The emergence and expansion of the development paradigm based on service-oriented approaches have been behind the elaboration of new models and methods that aim at facilitating the reuse of services within multiples context of use. This requires providing systematically services with several possible realizations. Among the promising approaches that achieve this objective is the management of variability which has been widely adopted by the software engineering disciplines and whose objective is to facilitate the adaptation or the configuration of software artifacts in a systematic way. Hence, in this work we provide a framework for the development (for and by reuse) of services supporting variability, called «SV3R» (Service Variability Representation and Resolution for Reuse). This paper introduces at first the concept of variability. Afterwards, it presents some related work, before giving an overview of the framework SV3R and describing its components.

Mots-clés

Réutilisation, Gestion de la variabilité, SOA, Web services, Ingénierie des modèles.

Keywords

Reuse, Variability management, SOA, Web services, Model engineering.

1. Introduction

Ces dernières décennies ont été marquées par la mutation du mode de déploiement des Systèmes d'Information (SI), passant d'un mode centralisé à un mode distribué. Par conséquent, les organisations sont plus que jamais interpellées à coopérer avec les SI des partenaires d'une manière transparente et flexible, tout en surpassant les contraintes technologiques. Le paradigme du développement basé sur l'approche orientée service est donc né dans ce contexte, afin de faciliter l'interopérabilité et la coopération des SI distribués et surtout de permettre à l'activité des organisations de piloter la technologie et non l'inverse.
Avec l'apparition de services, le nombre croissant d'utilisateurs de ces modules logiciels appartenant à différents domaines, a mis l'accent sur l'importance de munir les services de plusieurs réalisations possibles ainsi que de mécanismes d'adaptation lors de leur réutilisation. Ceci a suscité un vif intérêt à introduire la variabilité dès les premières phases de développement et d'élaboration de services. En effet, le concept de variabilité trouve ses origines dans l'ingénierie des lignes de produits. Cette dernière a pour objectif de minimiser le coût de développement de logiciels dans un domaine d'application particulier, et d'améliorer la réutilisation en ne se concentrant pas seulement sur le développement d'un produit unique, mais sur la production de familles de produits partageant des propriétés communes (Pohl, Bockle *et al.*, 2005), moyennant l'exploitation de la variabilité. Cette dernière se définit en général comme l'aptitude d'un système à s'adapter, à se spécialiser et à se configurer en fonction du contexte de son utilisation (Sinnema, Deelstra *et al.*, 2004). Pour cela, il est nécessaire, d'une part, d'identifier les parties fixes (ou communes du système) et les parties variables qui peuvent changer selon le besoin, et d'autre part, de représenter explicitement cette variabilité. Cette représentation peut couvrir les différentes phases de développement du système, de la conception à la réalisation. Un autre point important est celui de la résolution de la variabilité à l'utilisation, ce qui consiste à obtenir une variante possible du système réutilisable. Ces trois étapes (identification, représentation et résolution) font partie de ce que l'on appelle communément la gestion de la variabilité (Schmid et John, 2004). L'importance de ce concept a engendré son utilisation dans plusieurs disciplines d'ingénierie logicielle qui s'articulent autour de la réutilisation, telle que l'approche orientée services. Dans cette discipline, la variabilité est utilisée selon la nature du service, qui peut être atomique ou composite.
Cependant, la plupart des travaux de gestion de la variabilité de services se sont généralement contentés de se pencher sur la phase de réalisation, alors qu'avec l'apparition de nouveaux langages de modélisation de services, il est devenu intéressant de prendre en charge la variabilité de services pour les différents artefacts de développement de services, dès les premières phases de développement, afin d'améliorer sa réutilisation. Dans cette optique, nous proposons un cadre méthodologique pour le développement (*for* and *by reuse*) de services, baptisé SV3R (Service Variability Representation and Resolution for Reuse) (Chakir, 2014).
La deuxième section introduit les concepts liés à la gestion de la variabilité de services. L'état de l'art, traçant les travaux étudiés dans la littérature et justifiant l'intérêt de notre proposition, est traité dans la troisième section. La quatrième section est consacrée à la présentation du framework SV3R. finalement, un outil facilitant l'exploitation des services variables est présenté dans la cinquième section. Nous concluons ce papier par des perspectives de notre travail.

2. Concepts de la gestion de la variabilité de services

Nous présentons dans cette section la définition d'un ensemble de principes de la variabilité, qui s'avèrent indispensables pour la conception d'un service supportant la variabilité.

2.1. Définitions

La variabilité a fait l'objet de plusieurs applications dans plusieurs domaines, tels que l'ingénierie des domaines, des lignes de produits, des composants, des services, etc. Selon le domaine d'application, ce concept peut faire l'objet de différentes interprétations.
Ainsi, dans l'ingénierie des domaines, la variabilité est considérée comme la représentation des caractéristiques d'un domaine, exprimées en fonction de ses aspects essentiels visibles du domaine (Berger, She *et al.*, 2013).
Dans l'ingénierie des lignes de produits, le concept de la variabilité est utilisé pour regrouper les caractéristiques qui différencient les produits de la même famille (Ziadi, 2004).
Dans l'ingénierie des composants, les auteurs de (Kim, Her *et al.*, 2005) définissent la variabilité par la distinction des parties fixes des parties variables du composant. Ces parties variables concernent les niveaux suivants :
niveau des attributs : cela sous-entend les attributs utilisés dans les fonctions du composant.

- niveau logique : cela concerne les algorithmes des fonctions atomiques du composant ainsi que les post-conditions et la gestion des exceptions.
- niveau du workflow : la variabilité se situe au niveau des séquences d'appels de méthodes réalisant des fonctions de grande granularité offertes par un composant.

- niveau persistance de données : la variabilité se situe au niveau du schéma physique de persistance des attributs du composant.
- niveau interface : la variabilité apparaît au niveau des interfaces exposées du composant, et plus spécifiquement, au niveau des signatures de méthodes contenues dans ces dernières.

Dans l'ingénierie de services, les auteurs de (SHAPE, 2009) définissent la variabilité comme une technique centrale permettant la réutilisation de services dans différents scénarii d'exécution ainsi que la simplification de leur utilisation par les consommateurs. Ils distinguent la variabilité dans l'espace de la variabilité dans le temps. La première est appelée adaptabilité, la seconde, évolution. Dans notre travail, nous nous intéressons à l'adaptabilité. Ainsi, concernant ce type de variabilité, les auteurs de ce projet proposent deux mécanismes de mise en œuvre appliqués au niveau de la spécification de services, qui sont la configuration et l'extension. Le premier mécanisme concerne l'adaptation du service par le consommateur, qui sélectionne les caractéristiques souhaitées parmi les caractéristiques variables. Pour le second mécanisme, il s'agit de permettre au client de modifier les fonctionnalités existantes ou d'en ajouter. D'autres auteurs, à l'instar de (Sun, Rossing *et al.*, 2010), considèrent la variabilité de services comme la variabilité d'un artefact logiciel qui se définit comme l'aptitude de l'artefact à être étendu, changé, adapté ou configuré selon le contexte. Elle est appliquée au niveau de l'architecture des services, répartis en plusieurs vues de modélisation. D'autres la définissent en la liant à la notion de famille de services. À ce titre, selon (Mohan et Ramesh, 2002), une famille de services est un ensemble de services qui partagent des aspects communs et qui possèdent des aspects variables prédéfinis, et une architecture de famille de services est constituée de blocs pouvant être configurés donnant lieu à différentes variantes de services.

2.2. Propriétés

Les propriétés de la variabilité représentent l'ensemble des éléments nécessaires à la représentation de la variabilité dans les artefacts. Parmi les propriétés les plus utilisées dans la littérature, nous citons :

- **Unités de la variabilité** : elles ont été définies initialement dans les lignes de produits pour différencier des produits de la même famille. Deux concepts sont employés selon (Czarnecki et Eisenecker, 2000) : point de variation et variante. Selon ces auteurs, un point de variation est une partie du système où des choix doivent être faits afin d'identifier les variantes à utiliser. Pour (Jacobson, Griss *et al.*, 1997), un point de variation est défini comme *«one or more locations at which the variation will occur»* et une variante représente une réalisation spécifique d'un point de variation.
- **Portée de la variabilité** : cela sous-entend la portée du point de variation. Elle signifie les contraintes fournies par ce dernier sur le choix de ses variantes. (Bachmann et Bass, 2001) définissent quatre portées, **Optionnelle**, **Alternative**, **Alternative optionnelle** et **Ensemble d'alternatives** :
  - **Optionnelle** : un point de variation de portée optionnelle fournit des variantes qui peuvent être sélectionnées ou non. Par exemple, un système de e-commerce peut proposer ou non la vérification de l'éligibilité des clients demandeurs de produits, en restreignant la liste des pays d'appartenance des clients éligibles à l'utilisation du système. Ainsi, dans ce cas, le système peut contenir un point de variation «Vérification commande» optionnel, lié à une variante «Vérification éligibilité».
  - **Alternative** : un point de variation de portée alternative permet le choix d'une seule variante parmi n choix possibles (XOR). Par exemple, dans un système de e-commerce, on peut admettre seulement un des deux types d'envois de commande aux clients : expédition directe de commande ou envoi de commande par transporteur. Ainsi, nous avons un point de variation «Envoi de commande» de portée alternative, lié à deux variantes «Expédition directe de commande» et «Envoi de commande par transporteur».
  - **Alternative optionnelle** : un point de variation de portée alternative optionnelle est un point de variation de portée optionnelle qui offre des variantes alternatives. Ce type de variation permet le choix d'aucune ou d'une seule variante parmi les n choix possibles. Par exemple, le paiement cash dans un système de e-commerce peut contenir un point de variation optionnel sur la gestion de garantie. Ce point de variation offre deux alternatives possibles : paiement avec dépôt de garantie et paiement sans dépôt de garantie. Le concepteur peut choisir une ou aucune variante parmi ces deux alternatives.
  - **Ensemble d'alternatives** : un point de variation de type ensemble d'alternatives signifie qu'il existe de multiples réalisations de cette variation et qu'au moins une doit être sélectionnée. Par exemple, un système de e-commerce peut intégrer plusieurs moyens de paiement alternatifs : paiement par carte visa, par carte fidélité ou cash.
  - **Contraintes de dépendance de la variabilité** : le choix d'un point de variation ou d'une variante peut influencer sur le choix d'autres points de variation ou variantes. Ce type de relation est appelé contrainte de dépendance de la variabilité. Les auteurs de (Berger, She *et al.*, 2013) distinguent deux types de contraintes, l'**inclusion** et l'**exclusion mutuelle**.

  - La contrainte d'**inclusion** entre deux éléments variables indique que le choix d'un élément variable exige la présence d'un autre élément variable lors de la réutilisation de l'artefact comportant ces éléments. La contrainte d'**exclusion mutuelle** entre deux éléments variables spécifie que la présence de l'un de ces éléments prohibe la présence de l'autre lors de la réutilisation de l'artefact comportant ces éléments.
  - À titre d'exemple, soit un système de e-commerce comportant un point de variation «Paiement» lié à deux variantes : «Paiement à crédit» et «Paiement sans crédit», et un second point de variation «Mode de paiement» lié à deux variantes : «Paiement à la réception» et «Paiement en ligne». Dans ce système, le choix de la variante «Paiement à la réception» **inclut** le choix de la variante «Paiement sans crédit» et **exclut** la présence de la variante «Paiement à crédit».
- **Types de la variabilité** : ils permettent de classifier les points de variation. Plusieurs classifications sont proposées selon la dimension, la visibilité ou le niveau de représentation. En effet, les auteurs de (Pohl, Bockle *et al.*, 2005) classifient la variabilité en deux types selon sa dimension :
  - Variabilité **dans le temps** : elle concerne le fait qu'il y ait différentes versions d'un artefact à différents moments. On parle de l'évolution de l'artefact.
  - Variabilité **dans l'espace** : elle est mise en évidence quand un artefact capture différentes formes en même temps.
  - Selon ces mêmes auteurs, la variabilité peut aussi être classifiée selon la visibilité des variantes associées aux points de variation, en variabilité interne et variabilité externe :
  - Variabilité **interne** : les variantes associées ne sont visibles qu'au développeur et non à l'utilisateur.
  - Variabilité **externe** : les variantes associées sont visibles aux deux acteurs.

  D'autres travaux classifient la variabilité selon le niveau de représentation des points de variation dans l'artefact (Kim, Her *et al.*, 2005) ou dans le cycle de développement (Saidi, 2009).

Après avoir traité les différentes propriétés de la variabilité, nous allons aborder dans la section suivante le processus de gestion de la variabilité.

2.3. Processus de gestion de la variabilité

La gestion de la variabilité est une technique qui a été utilisée dans plusieurs disciplines d'ingénierie logicielle, afin d'améliorer la réutilisation des artefacts. Les auteurs de (Schmid et John, 2004) donnent la définition suivante au processus de gestion de la variabilité :
«Variability management encompasses the activities of explicitly representing variability in software artifacts throughout the lifecycle, managing dependences among different variabilities, and supporting the instantiation of the variabilities».
Ces auteurs distinguent ainsi les étapes suivantes dans un processus de gestion de la variabilité :

- La représentation de la variabilité à partir des artefacts de base.
- La définition et la gestion des contraintes de dépendances de la variabilité.
- La spécification de l'instanciation des différentes variantes, c'est-à-dire la sélection des variantes utiles.

Concernant la représentation de la variabilité, nous avons identifié dans la littérature deux types de représentations :

- **Intégré à un formalisme** : dans ce cas, la représentation de la variabilité utilise des notations existantes telles que le langage UML (Ziadi, 2004).
- **Orthogonal** ou **séparé des modèles de développement** : certaines approches optent pour une représentation séparée des modèles de développement (Pohl, Bockle *et al.*, 2005) et (Ghaddar, Tamzalit *et al.*, 2012).

Dans ce travail, nous nous intéressons à la gestion de la variabilité de service. Ainsi, dans la section qui suit, nous exposons un état de l'art des approches de l'ingénierie de services qui ont traité de ce concept.

3. État de l'art

Avec l'élargissement de l'adoption de l'approche orientée services, les mondes industriel et académique s'intéressent de plus en plus à la modélisation des services et à la définition des formalismes, et aux notations nécessaires pour décrire les solutions SOA, indépendamment des technologies et des normes. Plusieurs méthodes de modélisation ont ainsi vu le jour. Ces méthodes peuvent être divisées en deux parties : méthodes de développement centrées processus et méthodes de développement dirigées par les modèles. La première catégorie de méthodes se focalise sur les différentes phases qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. La deuxième catégorie s'intéresse aux modèles de services et adopte la démarche MDA.

Parmi les méthodes de la première catégorie, nous distinguons : SOUP (Service Oriented Unified Process) (Mittal, 2010), Erl (Erl, 2007), IBM Service-Oriented Analysis and Design (SOAD) (Zimmermann, Miksovic *et al.*, 2012), IBM Service Oriented Modeling and Architecture (SOMA) (Arsanjani, Ghosh *et al.*, 2008) et SCDD (Service Component-oriented Design and Development) (Bendekkoum et Boufaida, 2013).
Concernant la deuxième catégorie, plusieurs méthodes ont adopté MDA à l'instar des travaux de (Kenzi, 2010), (Amsden, 2010) et (Elvesæter, Mohagheghi *et al.*, 2011).
Par ailleurs, la multitude des clients potentiels de services a mis en évidence l'importance de développer des services réutilisables possédant plusieurs réalisations selon le contexte d'utilisation. Dans ce sens, quelques méthodes ont intégré le concept de réutilisation. Par conséquent, nous avons procédé à l'évaluation de l'ensemble des méthodes en nous basant sur la prise en compte de la réutilisation et la possibilité d'adaptation selon le contexte. Nous avons ainsi constaté que l'axe réutilisation n'est pas pris en considération par toutes les méthodes. De plus, ces méthodes ne couvrent pas toutes les phases de développement, hormis la méthode SOMA, et la plupart d'entre elles ne traitent que de la vue producteur de services, alors qu'il est important de prendre en compte la vue consommateur, afin d'assurer l'adaptation des services selon le contexte d'utilisation. De plus, ces méthodes ne proposent pas de mécanismes permettant de fournir des services avec plusieurs réalisations possibles afin de permettre leur réutilisation dans plusieurs contextes. Parmi les pistes importantes permettant de fournir de manière systématique plusieurs réalisations de services, figure la gestion de la variabilité.
En effet, de nombreux travaux se sont intéressés à la gestion de la variabilité de services. Parmi les approches étudiées traitant de la variabilité des services atomiques, nous citons l'approche VOE de (Narendra, Ponnalagu *et al.*, 2008) qui propose trois étapes, permettant l'identification des variations, leur représentation et la génération des variantes associées. La première étape est Variation-Oriented Analysis (VOA) qui consiste en l'analyse de la variabilité des solutions. La deuxième étape est Variation-Oriented Design (VOD) qui concerne l'élaboration d'un modèle de variation associé au système étudié. finalement, la troisième étape, Variation-Oriented Implementation (VOI), permet l'élaboration des implémentations des solutions variantes sur la base d'un module de composition, qui lie la solution de base et le modèle de variation établi pour les variantes en question. Les auteurs de (Chang et Kim, 2007b) utilisent les notions de «variation point», «variant», «types of variation point», et «dependency» pour représenter la variabilité au niveau des contrats WSDL et BPEL (Business Process Execution Language) associés aux services. Ces éléments sont inclus dans un schéma XML pour la variabilité, qui sera ultérieurement utilisé dans les spécifications BPEL et WSDL. Quant aux travaux de (Stollberg et Muth, 2009), ils s'intéressent à la variabilité des services atomiques en proposant un méta-modèle pour la représentation de la variabilité des interfaces de services. Ce méta-modèle est une extension de SOAML et permet la spécification des opérations, des messages et des propriétés (obligatoires et optionnelles), ainsi que les différentes dépendances entre les éléments variables. L'approche de (Nguyen, Colman *et al.*, 2013) capture et représente la variabilité métier de manière intégrée dans la réalisation de services atomiques, sans tenir compte de leurs détails techniques. Cette approche s'articule autour d'un langage appelé «Web service variability language» (WSVL) qui associe les descriptions de la variabilité à la spécification WSDL. La spécification résultante est divisée en quatre briques d'information : variabilité de services, fonctionnalités de services, correspondances avec les caractéristiques de la variabilité et point d'adaptation.
Concernant la variabilité des services composites, l'approche SOAPLE (Abumatar et Gomaa, 2013) propose une méthode de modélisation de la variabilité multi-vues pour la conception et l'implémentation de familles de systèmes orientés services variables. Cette approche intègre les concepts des lignes de produits, notamment en ce qui concerne la modélisation des caractéristiques du système et les techniques d'analyse de la variabilité. Elle propose aussi des règles de dérivation qui permettent de générer les membres des familles d'applications, ainsi qu'un outil qui permet de prendre en charge cette dérivation. La réalisation des services se passe au niveau du client en s'appuyant sur la démarche MDA. L'approche de (Koning,Sun *et al.*, 2009) s'intéresse également à la variabilité des services composites. Elle propose une extension BPEL appelée VxBPEL permettant la représentation des points de variation et des variantes.
L'étude des différentes approches de gestion de la variabilité de services (Chakir, 2014) (Chakir, Fredj *et al.*, 2012b), (Chakir et Fredj, 2011), nous a permis de noter l'absence d'une démarche exhaustive prenant en charge la gestion de variabilité aux différents niveaux d'abstraction du développement de services. Ceci justifie le besoin d'un framework de développement de services, prenant en charge la gestion de la variabilité à différents niveaux d'abstraction. La section 4 présente SV3R, notre proposition de framework.

4. Framework de gestion de la variabilité de service

4.1. Vue d'ensemble de SV3R

Afin d'assurer le développement *pour* et *par* la réutilisation de services supportant la variabilité, nous avons proposé un framework appelé SV3R (Chakir, 2014). Ce framework peut être appréhendé selon trois composants :

- **Producteur de services configurables** : ce composant représente le responsable de la production des services variables appelés dans notre approche services configurables. À ce niveau, ce framework propose une méthode de développement de services configurables, qui comprend un langage de

représentation de la variabilité de services VarSOAML (Chakir et Fredj, 2011), (Chakir, Fredj *et al.*, 2012a) ainsi qu'un processus de développement SVDev (Chakir, 2014) basé sur la démarche MDA. Ce processus est descendant, puisqu'il s'appuie sur les exigences métier pour l'identification de services. Cette méthode permet de représenter la variabilité à tous les niveaux d'abstraction du processus de développement, de la spécification à l'implémentation.

- **Registre de services configurables** (Configurable Service Registry) : ce composant permet de stocker les différentes composantes des services configurables : modèles, contrats et implémentation. Les services y sont stockés sous forme de spécifications structurées en plusieurs facettes : facette Description, facette Signature, facette Variabilité et facette Variantes.
- **Client de services configurables** : ce composant représente le développeur d'applications par réutilisation de services. À ce niveau, le framework propose un système de réutilisation de services configurables, composé d'un processus de réutilisation et guidé par un ensemble de directives de résolution de la variabilité des services configurables. Ce système est supporté par un outil appelé Configurator.
- La figure 1 illustre une vue d'ensemble du framework SV3R. Le producteur est responsable du développement (pour la réutilisation) de services configurables supportant la variabilité. Ces services sont stockés dans le registre. Le client, quant à lui, suit le processus de réutilisation moyennant l'outil Configurator pour répondre à ses besoins en matière de services réutilisables.

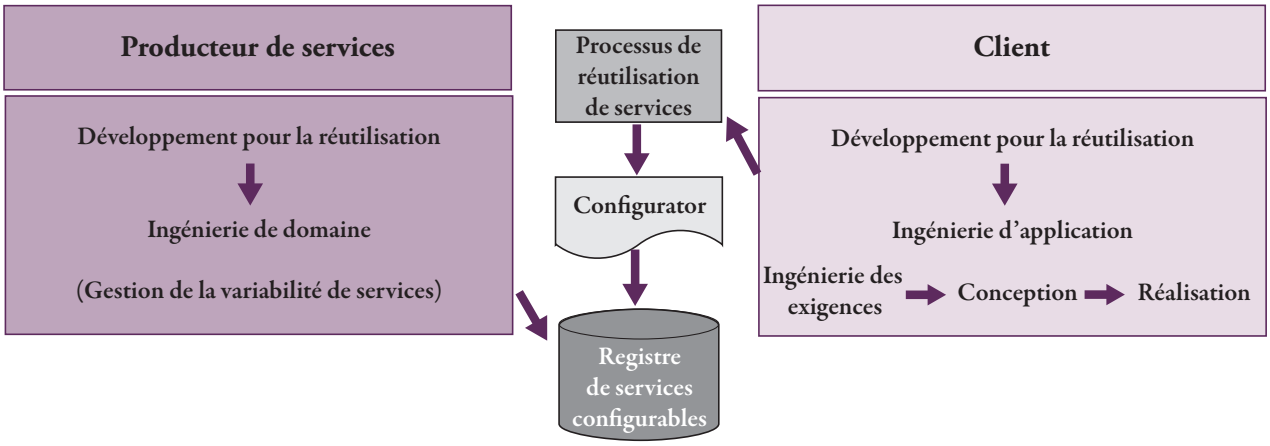

figure 1. Vue globale de SV3R (Chakir, 2014)

Dans ce qui suit, nous présentons les principales composantes de ce framework, en commençant par le processus de développement de services SVDev.

4.2 Processus de développement de services configurables basé sur MDA : SVDev

Le processus SVDev est un processus de développement de services s'appuyant sur la démarche MDA (Model Driven Architecture) proposée par l'OMG (OMG, 2001) pour supporter le développement de systèmes complexes. La démarche MDA consiste à utiliser les modèles aux différentes phases du cycle de développement d'une application et définit trois types de modèles représentant et séparant différentes préoccupations : le CIM (Computation Independant Model), le PIM (Platform Independent Model) et le PSM (Platform Specific Model).
Le CIM : permet de représenter les exigences métier d'un système.

- Le PIM : représente un modèle ayant pour objectif la description de la logique métier d'un système, indépendamment de toute plateforme ou technologie.
- Le PSM : permet de représenter une implémentation d'un système conformément à une technologie particulière (JAVA, .NET, CORBA, etc.).

Cela permet la séparation de la spécification fonctionnelle de l'implémentation sur une plateforme donnée lors du développement logiciel, et favorise l'adaptabilité aux évolutions des plateformes et des techniques. En plus de cette séparation adaptée à la démarche MDA, les modèles élaborés par le processus SVDev sont répartis en plusieurs vues de modélisation illustrées dans la figure 2.

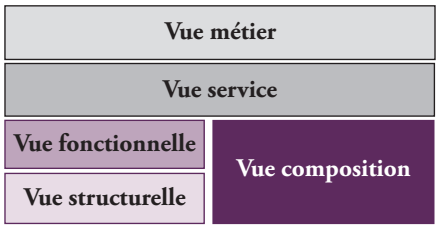

figure 2. Vues de modélisation de SVDev (Chakir et Fredj, 2011)

La figure 2 représente les vues de modélisations produites par le processus SVDev. Ainsi, nous distinguons :

- **Vue métier** : elle contient les processus métier qui régissent le système étudié, ainsi que le modèle d'information qui capture la sémantique des données échangées par ces processus.
- **Vue service** : elle permet l'identification des services associés aux processus métier.
- **Vue fonctionnelle** : elle contient les diagrammes d'interfaces ainsi que les diagrammes de types de messages associés contenant les types de données échangés entre les intervenants pour chaque service du système.
- **Vue structurelle** : elle contient les diagrammes de composants correspondants aux producteurs de services et cela pour chaque service du système.
- **Vue composition** : elle contient les diagrammes d'activités qui spécifient le comportement des différents services composites.

Une vue globale du processus SVDev est présentée dans la figure 3. Il est composé de deux sous-processus intégrant la prise en charge de la variabilité. Le premier concerne l'étude et l'analyse métier qui se caractérise par l'analyse du domaine. L'objectif est, d'une part, de produire des processus métier qui couvrent toutes les exigences d'un domaine métier, et d'autre part, d'identifier les exigences variables et de les représenter au niveau des modèles métier.
Le deuxième sous-processus concerne la spécification et l'implémentation de services. Il est structuré en plusieurs phases, qui permettent d'identifier et de spécifier les services configurables supportant la variabilité :
Phase d'analyse : elle concerne la spécification de la vue services. Ainsi, elle se base sur la vue métier pour identifier les services et produire les diagrammes d'architecture, les diagrammes de contrats ainsi que les diagrammes de séquences associés aux contrats.

- **Phase de conception** : elle concerne la spécification de la vue fonctionnelle, de la vue structurelle et de la vue composition, en se basant sur la vue service et la vue métier. Cette phase produit ainsi les diagrammes d'interfaces, les diagrammes de messages associés aux interfaces de services, les diagrammes de participants ainsi que les diagrammes d'activités de la vue composition.
- **Phase d'implémentation** : elle s'articule sur le mécanisme de transformation des modèles de spécification de services pour la production des Web services et des documents XML de spécification de la variabilité, associés à chacun des services identifiés.

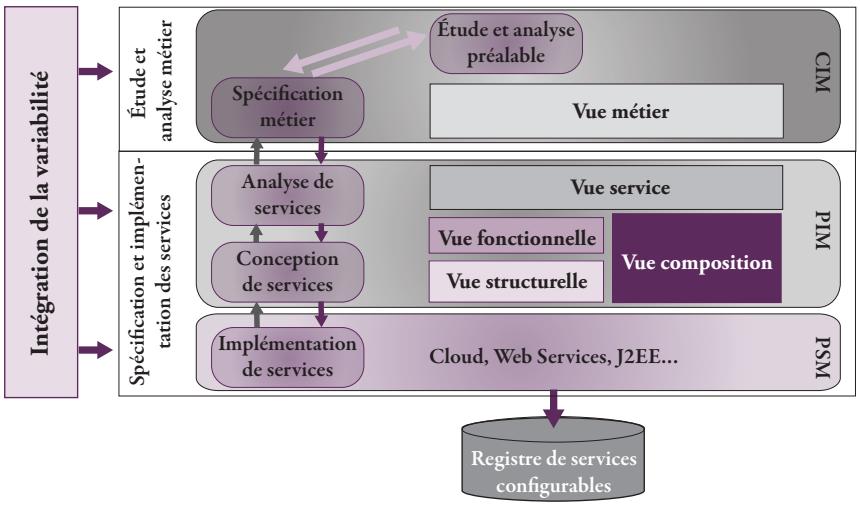

figure 3. Vue globale du processus de développement de services configurables SVDev (Chakir, 2014)

Le résultat de ce processus est un ensemble de services configurables qui seront déployés dans un registre spécifique.

4.3. Registre de services configurables

Les services configurables, issus du processus SVDev, sont déployés dans un registre que nous avons organisé en plusieurs facettes. Ce type d'organisation s'inspire de la proposition de (SECSE, 2007), qui permet de stocker les différentes spécifications et livrables associés aux services.
Selon (SECSE, 2007), une facette représente un regroupement de spécifications et de propriétés de services selon une perspective, afin de donner une description partielle des services. L'objectif est d'améliorer l'exploitation du service par le consommateur. Une facette est caractérisée par un type représentant la perspective du regroupement, et une ou plusieurs spécifications de la facette contenant une description de certaines propriétés de services, exprimée dans un langage donné (XML, OCL, WSDL, …).
Les facettes retenues dans ce registre sont :
Facette **Description** : elle contient les informations descriptives de services.

- Facette **Signature** : elle contient les spécifications techniques des services, à savoir les contrats WSDL *(Web Service Description Language)* et BPEL *(Business Process Execution Language)* qui permettent de décrire les fonctionnalités techniques des services, ainsi que les modèles associés, en XMI (XML Metadata Interchange).
- Facette **Variabilité** : elle contient les spécifications de la variabilité, spécifiques au service atomique et au service composite.
- Facette **Variantes** : cette facette regroupe un ensemble de spécifications correspondant aux variantes du contrat de service, d'implémentation et de modèles du service.

Le registre proposé permet de stocker une collection de spécifications de services caractérisées par plusieurs facettes, les différentes spécifications de ces facettes et une collection de consommateurs de services.
La figure 4 montre la structure d'une spécification de service, qui inclut le nom, l'identifiant, la date de la dernière modification, les différentes facettes de description du service avec leurs différentes spécifications. Concernant une spécification de facette, elle contient généralement les informations à propos du langage, et les données de spécification représentées en XML.
Ainsi, la spécification d'un service dans ce registre prend la forme suivante (cf. figure 4) :

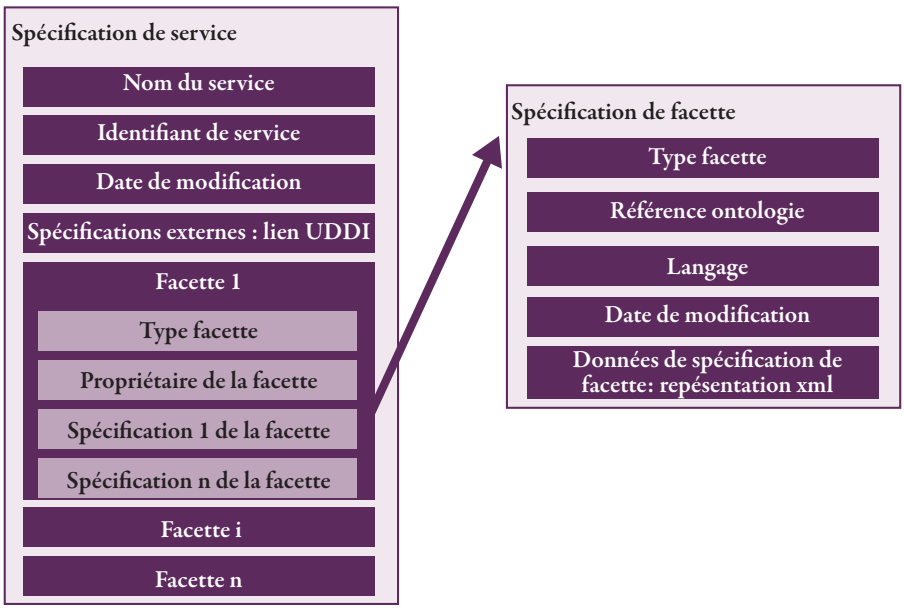

figure 4. Spécification d'un service

Après avoir donné une vue globale du registre de services configurables, et présenté sa structure et son contenu, nous exposons dans la section suivante le processus de réutilisation de services, qui constitue une des principales composantes du framework SV3R.

4.4. Processus de réutilisation de services

La réutilisation de services par un client suit les étapes suivantes :
Recherche des services configurables : ce processus de réutilisation commence par une étape de recherche d'un service configurable. La recherche d'un service du registre est liée aux informations employées pour sa description. Dans la littérature, nous distinguons principalement deux types de mécanismes de recherche : lexicale et sémantique (Zachos, Maiden *et al.*., 2007). Le premier désigne la recherche par mots-clés et le second se base sur l'analyse de la requête et utilise l'occurrence de termes similaires dans la recherche de résultats plus pertinents. Les auteurs de (Chang et Kim, 2007a) soulignent que ce deuxième type de recherche est le plus performant. Plus récemment, certains travaux sur la découverte des services Web ont proposé de combiner l'aspect sémantique et le raisonnement à partir de cas, afin d'améliorer le processus de découverte par la qualité des services identifiés [El Bitar, 2014].
Cependant, pour des contraintes de temps, nous nous sommes contentés de la recherche lexicale, à l'instar de celle qui est utilisée dans UDDI (Universal Description Discovery and Integration) (Rajasekaran, Miller *et al.*, 2004). Le plus important dans notre phase de recherche de services était de récupérer une liste de services configurables candidats, à partir de laquelle le client choisira le plus adéquat pour une éventuelle configuration. Une perspective de ce travail serait d'optimiser la découverte des services en s'inspirant des dernières contributions dans le domaine.

- **Résolution de la variabilité des services configurables** : cette étape consiste à configurer le service souhaité en choisissant des variantes des éléments variables exposés par ce dernier. Dans la littérature, nous trouvons deux types de résolution de la variabilité : explicite, où le client choisit les variantes exposées (SHAPE, 2009), ou bien implicite à l'exécution en se basant par exemple sur les «politiques» associées aux données de l'utilisateur (Tao et Yang, 2007). Dans notre cas, la résolution de la variabilité est explicite, et cible à la fois, les modèles, le contrat et l'implémentation de services stockés dans le registre de services configurables.
- **Intégration de variantes de services** : cette troisième et dernière étape vise à intégrer le service configuré dans le développement en cours du système à base de services. Certaines approches se sont intéressées à la problématique d'intégration de services dans les développements en cours. Ainsi, les auteurs de (Zachos, Maiden *et al.*, 2007) proposent la modification des exigences des applications selon les services découverts, afin de les intégrer par la suite dans les compositions de services. (Chang et Kim, 2007b) proposent un diagramme appelé «Service Integration Architecture» qui permet de documenter la phase d'intégration de services dans le processus de développement. Cependant, cette phase n'est pas prise en charge dans la version actuelle de notre système. Nous considérons que cette activité d'intégration d'une variante de service relève d'une problématique de recherche à part entière.

5. Mise en œuvre de SV3R

SV3R est composé de plusieurs briques, d'une méthode de développement basée sur MDA pour la production de services, d'un registre pour le stockage des services configurables et d'un système de réutilisation par les clients. Ainsi, la mise en œuvre de ce framework s'est basée sur trois modules :
SV3R-MT : «MT» *(Model Transformation)* représente le module de génération de web services configurables par application de la démarche MDA. Il consiste en la réalisation des méta-modèles des couches PIM et PSM, moyennant l'outil EMF ainsi que des règles de transformation en ATL.

- **SV3R-REG** : il consiste en l'implémentation du registre de services configurables représentant un espace de stockage et de publication de services dans SV3R. Ce module est implémenté sous eXist qui représente une base de données XML native.
- **SV3R-CONfiG** : il représente l'outil de configuration de services par les utilisateurs.

En ce qui concerne ce module, il permet d'assister les utilisateurs dans la réutilisation de services. En effet, afin de guider les consommateurs dans la configuration de services, nous avons proposé un outil baptisé «*ServiceConfigurator Tool*» réalisé avec la technologie J2EE (Java Enterprise Edition), dans l'environnement de développement Eclipse. L'outil propose aux consommateurs un ensemble d'écrans conviviaux qui permettent la recherche, la configuration ainsi que l'intégration de services.
La figure 5 illustre l'écran de configuration pour un service atomique appelé «TraitementRéception» qui contient un ensemble de fonctionnalités permettant le traitement de commandes dans un système de e-commerce. Ainsi, les éléments variables du service sont répartis en trois catégories : les opérations variables de type variation et variante, les variables messages et les variables types. L'utilisateur sélectionne les éléments qu'il veut garder dans la variante de service, un contrôle de conformité automatique est réalisé sur les éléments sélectionnés en se basant sur les contraintes de dépendances entre les éléments variables, afin de garantir la cohérence de la configuration.

fig. 5 : Écran de configuration de services

6. Conclusion

Avec l'évolution de l'ingénierie de services, la modélisation de services gagne de plus en plus d'importance. Ceci a suscité un intérêt pour la réutilisation de services. En effet, les services ont longtemps été considérés comme des boîtes noires, dont la réutilisation se réduisait à une simple invocation à l'exécution, moyennant le recours à des contrats de service. Cependant, avec l'expansion de l'adoption de l'approche orientée services, ainsi que l'apparition des langages et des méthodes de modélisation de services, le besoin de réutilisation de services, dans toutes les phases de développement d'application par réutilisation, est devenu essentiel, et cela, pour leurs différents artefacts (modèles, contrats et implémentation).

Par ailleurs, l'élargissement des clients potentiels de services a créé le besoin de les munir de plusieurs variantes, afin de favoriser leur réutilisation dans différents contextes d'utilisation. Cela justifie l'intérêt porté par la communauté qui s'intéresse à l'ingénierie de services pour la gestion de la variabilité.

Ainsi, plusieurs approches se sont intéressées à la gestion de la variabilité de services, mais l'étude de ces approches nous a permis de constater l'absence d'une démarche exhaustive prenant en charge la gestion de variabilité aux différents niveaux d'abstraction du développement de services. Ainsi, la contribution principale de cet article est la proposition d'un framework de développement de services prenant en charge la variabilité à différents niveaux d'abstraction. Pour ce faire, nous avons conçu plusieurs briques de ce framework :

- Un processus de développement, baptisé SVDev, qui intègre l'identification et la représentation de la variabilité de services à tous les niveaux d'abstraction.
- Un registre de services configurables : qui permet de répertorier les services configurables. Ainsi, il contient non seulement les contrats de service à l'instar des registres classiques, mais aussi l'implémentation, les modèles et la spécification de la variabilité.
- Un processus de réutilisation de services configurables : qui inclut les étapes de recherche, de résolution et d'intégration de services.

Afin de valider notre proposition, nous avons tout d'abord réalisé la phase d'implémentation du processus de développement de services configurables, en ciblant les plateformes technologiques des Web services. Ensuite, nous avons implémenté le registre de services configurables sous forme d'une base de données eXist. Enfin, pour assister l'utilisateur dans le processus de réutilisation de services configurables, un prototype de l'outil de configuration de services est réalisé avec la technologie J2EE.

Comme perspectives de ce travail, nous jugeons important d'introduire une ontologie de domaine. En effet, l'annotation des services par les descriptions sémantiques , combinée au raisonnement à partir de cas (El Bitar, 2014), pourrait améliorer les phases de recherche, de configuration et d'intégration de services. Une autre piste d'amélioration est la formalisation de l'approche, notamment du processus SVDev qui peut être formalisé par un langage tel que SPEM (Software and System Process Engineering), afin de permettre son intégration avec d'autres méthodes, et également son automatisation.

7. Références

Abu-Matar, M., Gomaa, H., (2013). An Automated Framework for Variability Management of Service-Oriented Software Product Lines. *Proceeding of IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*. San Francisco Bay, USA: 25-28 Mars, 260-267.

Amsden, J., (2010). Modeling with SoaML, the Service-Oriented Architecture Modeling Language, *IBM developerWorks*. **http://www.ibm.com/developerworks/rational/library/09/modelingwithsoaml-1/index.html**.

Arsanjani, A., Ghosh, S., Allam, A., *et al.*, (2008). SOMA: A method for developing service-oriented solutions. *IBM Systems journal.* Vol.47, No.3, 377-396.

Bachmann, F., et Bass, L., (2001). Managing variability in software architecture. *ACM SIGSOFT Software Engineering Notes*, Vol. 26, No. 3, ISBN: 1-58113-358-8, 126 – 132.

Bendekkoum, S., Boufaida, M., (2013). A Service Component-Oriented Design and Development Methodology for Developing SOA-based Applications. *Proceedings of the fifth International Conferences on Advanced Service Computing*. Valencia, Spain: May 27 - June 1, 13-18.

Berger, T., She, S., Lotufo, R., Wasowski, A. *et al.*, (2013). A study of variability models and languages in the systems software domain. *Software Engineering, IEEE Transactions*. Vol. 39, No. 12, 1611-1640.

Chakir, B., (2014). *Contribution à la gestion de la variabilité des services : vers l'amélioration de la réutilisation dans SOA*, Thèse de doctorat en Informatique, Rabat : ENSIAS, Université Mohammed V.

Chakir, B., Fredj, M., et Nassar, M., (2012). Promoting reuse in web services by managing variability. *The 3rd International Conference on Multimedia Computing and Systems*, Tanger, Maroc: 10-12 Mai.

Chakir, B., Fredj, M., et Nassar, M., (2012). A model driven method for promoting reuse in SOA-solutions by managing variability, *IJCSI International Journal of Computer Science Issues*. Vol. 9, No 3.

Chakir, B., Fredj, M., (2011). A model driven approach supporting multi-view services modeling and variability management. *Proceedings of the International Conference on Enterprise* Information Systems (ICEIS). Beijing, China: 8-11 Juin.

Chang, S.H., et Kim, S. D., (2007). A Systematic Approach to Service-Oriented Analysis and Design. *8th International Conference, PROFES*. Riga, Latvia: 2-4 Juillet, 374-388.

Chang, S.H., et Kim, S.D., (2007). A Variability Modeling Method for Adaptable Services in Service-Oriented Computing. *Software Product Line Conference (SPLC 2007)* .Kyoto, Japan: 10-14 Septembre, 261-268.

Czarnecki, K., et Eisenecker, U., (2000).*Generative Programming – Methods, Tools and Applications*. Addison-Wesley.

El Bitar, B., (2014). *CBR4WSD : Une approche de découverte de services Web par Raisonnement à Partir de Cas*, Thèse de doctorat, Rabat : Ecole Mohammadia d'Ingénieurs, Université Mohammed V.

Elvesæter, B., Carrez, C., Mohagheghi, P., Berre, A.J., *et al.*, (2011), Model-driven Service Engineering with SoaML. *Service Engineering - European Research Results*, Wien: Springer, 25-54.

Erl, T., (2007). *SOA Principles of Service Design*. PRENTICE HALL.

Ghaddar, A., Tamzalit, D., et Assaf, A., (2012). Gestion de la variabilité dans les applications SaaS multi-locataire, *INFORSID*, 239-256.

Jacobson, I., Griss, M., et Jonsson, P., (1997). Software Reuse: Architecture Process and Organization for business Success. New York : ACM Press.

Kenzi, A., (2010). *Ingénierie des Systèmes Orientés Services Adaptables : Une Approche Dirigée par les Modèles*. Thèse de doctorat, Rabat : École Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Université Mohammed V- Souissi, Rabat, Maroc.

Kim, S.D., Her, J.S. et Chang, S. H., (2005). A theoretical foundation of variability in component-based development, Information and Software Technology (IST), Vol. 47, No. 10, 663–673.

Koning, M., Sun, C., Sinnema, M. et Avgeriou, P., (2009). VxBPEL: supporting variability for Web services in BPEL. *Information and Software Technology*, Vol. 51, 258-269.

Mittal, K., (2010). *Service Oriented Unified Process (SOUP)*. http://www.Kunalmitt*al*.com/ html/soup.shtml.

Mohan, K. et Ramesh, B., (2002). Managing Variability with Traceability in Product and Service Families. *In the proceedings of the 35th Hawaii International Conference on System Sciences*, 1309–1317.

Narendra, N.C., Ponnalagu, K., Srivastava, B. et Banavar, G.S., (2008). Variation-Oriented Engineering (VOE): Enhancing Reusability of SOA-based Solutions. *IEEE International Conference on Services Computing*. Honolulu, Hawaii, USA: 8-11 July.

Nguyen, T., Colman, A., Han, J., (2013). A Web Services Variability Description Language (WSVL) for Business Users Oriented Service Customization. *In Web Information Systems Engineering–WISE 2011 and 2012 Workshops, Lecture Notes in Computer Science*, Vol. 7652, 321-334.

OMG, (2011). Model Driven Architecture (MDA). http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01.

Pohl, K., Bockle, G., et Linden, F.V.D., (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*. Berlin Heidelberg : Springer-Verlag.

Rajasekaran, P., Miller, J., Verma, K., Sheth, A., Enhancing Web Services Description and Discovery to Facilitate Orchestration. *Proceedings of SWSWPC (In conjunction n with ICWS'2004)*, 34-47.

Saidi, R., (2009). *Conception et usage des composants métier processus pour les systèmes d'information.* Thèse de doctorat, Rabat : Université Mohammed V-Agdal.

Schmid, K., John, I., (2004). A Customizable Approach to Full Lifecycle Variability Management. *Science of Computer Programming*, Vol. 53, No. 3, 259-284.

SECSE, (2007). *Specification Language Definition-final.* http://www.secse-project.eu/?page_id=80.

SHAPE, (2009), Semantically-enabled Heterogeneous Service Architecture and Platforms Engineering, **http://www.shape-project.eu/**.

Sinnema, M., Deelstra, S., Nijhuis, J., et Bosch, J., (2004). COVAMOF: A Framework for Modeling Variability in Software Product Families. *The Proceeding of the Third Software Product Line Conference (SPLC2004)*, Springer Verlag Lecture Notes on Computer Science, Vol. 3154 (LNCS 3154), 197–213.

Stollberg, M., Muth, M., (2009). Service customization by variability modeling. *Service-Oriented Computing. ICSOC/ServiceWave Workshops*, Springer, 425-434.

Sun, C., Rossing, R., Sinnema, M., Bulanov, P., et Aiello, M., (2010). Modelling and managing the variability of web service-based systems, Journal of Systems and Software. *Elsevier*, Vol. 83, 502–516.

Tao, A., et Yang, J., (2007). Supporting Differentiated Services With Configurable Business Processes. *In the Proceeding of the International Conference : On Business Process and Services Computing (BPSC)*, 194-213.

Zachos, K., Maiden, N., Zhu, X., et Jones, S., «Discovering Web Services to Specify More Complete System Requirements», CAiSE, 2007.

Ziadi, T., (2004). *Manipulation de Lignes de Produits en UML*, Thèse de doctorat, Université de Rennes.

Zimmermann, O., Miksovic, C., Küster, J. M., (2012). Reference architecture, metamodel, and modeling principles for architectural knowledge management in information technology services. *Journal of Systems and Software*, Vol. 85, No. 9, 2014-2033.