

Une approche orienté aspect allant du modèle d'exigences au modèle de conception

Définition d'un processus itératif d'ingénierie logicielle

Sébastien Mosser¹, Gunter Mussbacher², Mireille Blay-Fornarino³, and Daniel Amyot⁴

¹ INRIA Lille–Nord Europe, LIFL (UMR CNRS 8022), Univ. Lille 1, France

² SCE, Carleton University, Canada

³ I3S (UMR CNRS 6070), Université Nice–Sophia Antipolis, France

⁴ SITE, University of Ottawa, Canada

Résumé Des approches orientées aspects sont aujourd'hui disponibles à chaque phase du développement d'un logiciel : analyse des exigences, conception, ou encore implémentation. Passer d'une phase à l'autre en conservant les aspects identifiés au préalable reste un défi majeur, pourtant peu étudié. Nous proposons dans [2] une approche itérative et dirigée par les préoccupations, permettant de transformer un modèle d'exigences orienté aspect en un modèle de conception lui aussi orienté aspect, et ceci de manière automatique. Cette approche est mise en œuvre en utilisant AoURN ("use case maps") pour le modèle d'exigence et ADORE pour le modèle de conception (orchestrations SOA). Elle permet l'encapsulation continue des préoccupations identifiées lors des exigences, transformées en artefact de conception. Nous proposons de plus une boucle de rétro-action permettant de remonter dans les modèles d'exigences des défauts constatés dans le modèle de conception, supportant ainsi un processus de développement itératif.

Introduction. De nombreuses méthodologies "aspects" (AO*) ont été proposées dans la littérature pour supporter la séparation des préoccupations au cours du cycle de vie du logiciel. Cependant, peu de travaux ont lié ces approches entre elles, alors que par essence l'AO* vise à conserver une telle séparation tout au long du processus de développement. Nous proposons d'étudier ce problème dans le contexte des Architectures Orientées Services (SOA), en prenant pour cible deux phases du processus : (i) analyse des exigences (au travers de modèles d'exigence AoURN [3]) et (ii) conception (au travers de modèles d'orchestration de services ADORE [1]). Les objectifs de ce travail sont illustrés en figure 1 : (1) fournir une approche automatisée (*i.e.*, implémentée par un algorithme) permettant de transformer les aspects identifiés au niveau des exigences en aspects utilisables lors de la phase de conception, et (2) remonter dans le modèle d'exigences les modifications apportées (*e.g.*, pour corriger des défauts) lors de la conception.

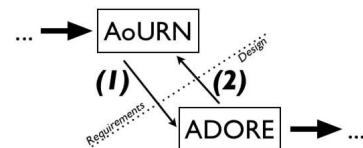


FIGURE 1. AoURN \rightleftharpoons ADORE

Étude de cas. Nous illustrons cette approche à l'aide de PICWEB, une application SOA de référence utilisée par plusieurs universités (Nice, Lille 1, Univ. Du Texas à Austin) comme support d'enseignement. L'objectif de cette application est représenté en terme de modèle d'exigence en figure 2(a) : lors de la réception d'un `tag` et d'un `seuil`, PICWEB va interroger le service Flickr à la recherche d'images associées à ce `tag`, puis restreindre le jeu d'images récupérées au `seuil` donné avant de le renvoyer. Une nouvelle "préoccupation" est l'utilisation du service de stockage Picasa en addition du service rendu par Flickr. Une telle préoccupation est représentée en figure 2(b) : en parallèle de l'invocation de Flickr, nous ajoutons une invocation à Picasa, suivi de la concaténation des résultats obtenus avant de poursuivre le comportement pré-préexistant. En terme SOA, les modèles d'exigences définis précédemment peuvent être mis en œuvre sous la forme d'orchestrations de services. Nous utilisons la plate-forme ADORE pour modéliser l'orchestration PICWEB (figure 3(a)) ainsi que l'aspect d'ajout du service Picasa (figure 3(b)).

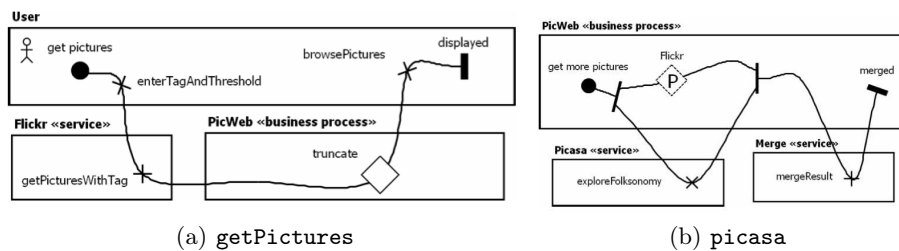


FIGURE 2. Modèle d'exigence orienté-aspect ("Use Case Maps" AoURN)

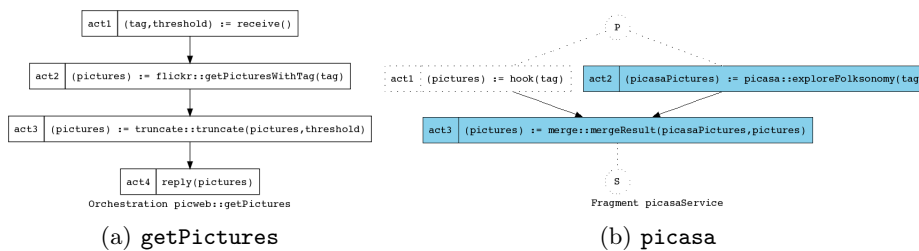


FIGURE 3. Modèle de conception orienté-aspect (Orchestration ADORE)

Mise en œuvre et Bénéfices. Nous proposons dans [2] un algorithme supportant la transformation automatique de modèles AoURN en modèles ADORE. Grâce à cet algorithme, nous bénéficions des avantages suivants dans le cadre du développement des applications :

Exigences → Conception : Les modèles de conception sont générés automatiquement à partir des modèles d'exigences. De fait, il n'y a pas de divergences entre les différents artefacts, ce qui augmente la cohérence entre les différentes phases de développement. De plus, les analyses effectuées sur les modèles d'exigences permettent d'assurer des propriétés sur les modèles de conception (*e.g.*, respect de règles métiers validées en termes d'exigences). Pour finir, les règles d'ordonnement d'aspects identifiées à un très haut niveau d'abstraction sont automatiquement adaptées pour être prise en compte par les aspects de composition.

Conception → Exigences : Le grain plus fin des modèles de conception permet d'identifier des interactions invisibles lors de la phase précédente. Ces informations peuvent être remontées dans les modèles d'exigences afin d'en renforcer la sémantique. Dans le même esprit, des flots de données inconsistants peuvent être identifiés, illustrant des oublis dans les modèles d'exigences. Pour finir, des choix de conception (*e.g.*, enrichissement d'interface de services) peuvent être réinjectés au niveau précédent pour être discutés en terme d'exigences avec le client.

Références.

- [1] Mosser, S. : Behavioral Compositions in Service-Oriented Architecture. Ph.D. thesis, Université Nice - Sophia Antipolis, ED STIC, Nice, France (Oct 2010), <http://nyx.unice.fr/publis/mosser:2010.pdf>
- [2] Mosser, S., Mussbacher, G., Blay-Fornarino, M., Amyot, D. : From Aspect-oriented Requirements Models to Aspect-oriented Business Process Design Models. In : 10th international conference on Aspect Oriented Software Development (AOSD'11) AR=20% , long paper : 1st round. , ACM, Porto de Galinhas (Mar 2011)
- [3] Mussbacher, G., Amyot, D., Araújo, J., Moreira, A. : Requirements Modeling with the Aspect-oriented User Requirements Notation (AoURN) : A Case Study. In : Katz, S., Mezini, M., Kienzle, J. (eds.) Transactions on Aspect-Oriented Software Development VII. Lect. Notes Comp. Sci., vol. 6210, pp. 23–68. Springer (2010)