Défis à l'utilisation d'une LPL dans le cadre du projet YourCast

Simon Urli
I3S, CNRS
Université Nice Sophia Antipolis
urli@i3s.unice.fr

Abstract

La construction des logiciels par assemblage de composants issus de différents domaines est une pratique courante et plébiscitée. Les lignes de produits logiciels (LPL) permettent de simplifier les tâches de production des logiciels et rendre ceux-ci plus robustes en représentant la variabilité dans une famille de logiciels. Cependant, les supports actuels aux LPL supposent que nous puissions représenter dans un seul système l'ensemble des composants impliqués, ce qui introduit une très grande complexité.

Sur un cas concret, apparemment simple, de réalisation d'un système de diffusion d'information, nous montrons dans cet article que l'utilisation usuelle de LPL présente différents problèmes. Nous amorçons une solution par composition de LPL dans une démarche IDM.

Mots-Clés : Ligne de Produits Logiciels, Système de Diffusion, IDM

Abstract

Building software by assembling components is usual and encouraged. The software product lines (SPL) allow to ease the task of software engineering and to make software safer by defining the variability in a set of softwares. However, current SPL tools assume that we can represent all the involved components in one system, which quickly raises its complexity.

On a concrete case, seemingly simple, of realizing a broadcasting system, we show in this article that the usual usage of SPL raises different problems. We are entering a solution by composing SPL in a MDE approach.

Keywords: Software Product Line, Broadcasting System, MDE

1 Introduction

La construction des logiciels repose aujourd'hui souvent sur l'assemblage de composants logiciels. L'utilisation de lignes de produits logiciels (LPL) permet de simplifier les tâches de production de logiciels sur mesure en définissant les points communs et les points de variabilité d'une famille de logiciels.

Le projet ANR Emergence YourCast[1] vise à la réalisation d'un outil de création de systèmes de diffusion d'informations à la portée de tous. Dans ce contexte, l'utilisation d'une LPL semble particulièrement pertinente de par l'importance de la variabilité latente aux systèmes de diffusion. Fort d'une expérience de plusieurs années sur un système pré-existant (JSeduite), nous avons abordé la construction d'une LPL par analyse des produits déjà déployés. Cependant, nous avons pu constater que les LPL et le formalisme des modèles de variabilité souffrent de certaines limitations auxquelles nous avons été confrontés.

Nous verrons ainsi en section 2 en quoi consiste précisément un système de diffusion d'information et nos cas d'utilisations associés, puis nous montrerons en section 3 quels sont les défis posés par les lignes de produits logiciels pour la modélisation de notre outil de création de diffuseurs d'information. Nous terminerons par une planification de nos futurs travaux en section 4 avant de conclure.

[1] http://yourcast.unice.fr

2 Etude de cas

Le projet ANR EMERGENCE YourCast portant sur la réalisation d'un système de diffusion d'information personnalisable par un utilisateur final fait suite à JSeduite, un système actuellement déployé dans différentes institutions.

2.1 De JSeduite à YourCast

JSeduite[2] a été écrit à l'origine afin de répondre aux besoins des institutions académiques. Il supporte la diffusion d'informations provenant de différentes instances (par exemple des réseaux de transports ou du restaurant universitaire), vers de multiples dispositifs (smartphone, PDA, écrans d'ordinateurs, écrans publiques). Le système a été utilisé comme plateforme de validation par le projet Faros[3]. La dernière version stable a été réalisée en Février 2010, après 4 ans de développement, et représente environ 70 000 lignes de code.

JSeduite est actuellement déployé dans trois institutions : l'école d'ingénieur Polytech'Sophia et deux instituts spécialisés pour les personnes présentant un handicap visuel : Clément Ader, dédié aux enfants et l'IRSAM, association pour les adultes.

Cependant, si ce système fonctionne à l'heure actuelle, il n'offre que peu de possibilités d'évolution, et n'est pas destiné à être adapté par l'utilisateur. Par ailleurs, le système n'a été déployé que pour une utilisation à petite échelle de l'ordre d'une dizaine d'écrans de même contenu. YourCast a pour objectif de permettre (i) une construction du système entièrement dirigée par un utilisateur final sélectionnant les modules qu'il souhaite utiliser, et (ii) une utilisation de l'ordre de la centaine d'écrans proposant des contenus différents.

Les choix des différentes parties du logiciel devront porter sur des concepts aussi différents que :

- *Les sources d'informations* : il peut s'agir de services tels que Twitter, FlickR, des Flux RSS, des calendriers, ou encore des services internes à des entreprises.
- *Les rendus d'affichage* : il s'agit d'un processus traitant les informations afin de les afficher de la façon souhaitée par l'utilisateur ; par exemple, un album photo affiché sous forme de mosaïque ou en diapositives.
- *Le layout* : il s'agit d'un squelette de l'écran d'affichage, contenant les différentes zones d'affichage et le style général de la page ; le layout est adapté à un support.
- *Les politiques* : il s'agit d'un processus effectuant des opérations sur les sources d'informations ; par exemple, une politique de filtrage d'informations, ou de tri sur des photographies.

Nous avons à l'heure actuelle plusieurs cas d'utilisation de tels écrans de diffusion dans des contextes très différents. L'étude de mise en place d'un outil de réalisation de tels écrans par un utilisateur final répond donc à une problématique très concrète.

2.2 Cas d'utilisation

Les systèmes de diffusion d'information deviennent omniprésents : qu'il s'agisse d'afficher des horaires de bus à un arrêt, ou de faire de la publicité dans les grandes surfaces, nous voyons des écrans en permanence. Le projet YourCast cible à court terme un certain nombre de cas d'utilisation spécifiques qui nous guident dans les choix de construction de notre outil.

[2] `http://www.jseduite.org`
[3] `http://www.lifl.fr/faros`

Un premier cas d'utilisation porte sur l'affichage d'informations dans le milieu académique. Dans ce contexte, la priorité est donnée aux *sources d'information*. Il s'agit alors d'afficher des informations qui peuvent être aussi diverses que les emplois du temps, la météo, les horaires de bus, les professeurs absents, etc. Le système doit également être prévu pour diffuser les informations au sein de *layouts* très différents en fonction des supports : écrans larges, écrans d'ordinateurs, smartphone, etc.

Notre deuxième cas d'utilisation, très spécifique, porte sur des écoles spécialisées pour les personnes présentant un handicap visuel. Ici le contexte oriente clairement la priorité sur les *layouts*. Les problématiques sont alors très différentes : les supports sont uniquement des grands écrans et l'affichage doit être adapté avec un très fort contraste et des mécanismes de répétition des informations. De plus, certaines sources ne font pas sens pour ce type d'institution.

Un dernier cas d'utilisation concerne les grands rassemblements associatifs : les supports et les sources peuvent ici être très différents d'un rassemblement à un autre. Il n'y a pas de priorité clairement établie sur l'un ou l'autre des concepts. En revanche, d'autres problèmes, plus techniques, sont à prendre en compte comme l'infrastructure réseau d'un rassemblement s'effectuant en pleine campagne.

Nous pouvons constater que dans certains cas la donnée importante pour la construction du système de diffusion concerne les sources à sélectionner (milieu académique) alors que dans d'autres cas, l'accent est mis sur le type d'affichage (instituts spécialisés). Il est donc indispensable que l'outil conçu puisse être abordé par l'un ou l'autre de ces aspects.

3 Défis à l'utilisation d'une LPL

L'objectif du projet YourCast est donc de fournir un outil permettant à un utilisateur final de sélectionner et lier les éléments qu'il souhaite voir apparaître dans son système en fonction des choix qu'il a précédemment effectués. Afin de pouvoir exprimer et gérer toute la variabilité des différents concepts présentés avant, nous souhaitons utiliser des lignes de produits logiciels (LPL).

En effet, les LPL ont pour but la construction de systèmes par sélection et assemblage de composants choisis. Nous avons basé nos travaux sur les modèles de fonctionnalités (*Feature Model* - FM)[4] tels que définis dans [6] mais nous étudions la possibilité d'utiliser d'autres modèles [5, 2].

Nos FM sont représentés sous forme d'arbres où chaque concept présentant un degré de variabilité constitue un noeud et chaque choix une feuille. Par ailleurs, il est possible d'exprimer au sein des FM des contraintes entre les différents éléments, restreignant les choix de l'utilisateur en fonction de ses sélections. Un choix valide de l'utilisateur correspond finalement à un sous-ensemble de l'arbre pour lequel les contraintes sont vérifiées.

Cependant, la modélisation de notre système sous forme d'une LPL a soulevé de nombreux problèmes quant à son utilisation concrète pour permettre la construction d'un diffuseur d'information valide.

3.1 Variabilité à plusieurs niveaux

> *Un système de diffusion est bien construit si à chaque source d'information correspond une ou plusieurs méthodes de rendu.*

[4] Afin de conserver le vocabulaire habituellement utilisé dans le domaine, nous conserverons dans la suite du document le terme de *feature* pour parler d'un élément du FM.

Dans notre ligne, il est donc nécessaire d'exprimer des relations entre nos features. Cependant, des relations vont également s'exprimer entre les configurations.

> *Dans une configuration d'un système de diffusion, (i) il existe au moins une source d'information ; (ii) une source d'information peut être sélectionnée plusieurs fois ; (iii) à une source d'information doit correspondre une source de rendu.*

Cela signifie (i) qu'il est nécessaire d'exprimer des cardinalités sur nos features, (ii) qu'une même feature peut apparaître plusieurs fois dans une configuration, (iii) que nous devons pouvoir exprimer des relations au sein d'une configuration.

La notion de cardinalité, bien qu'absente dans la plupart des formalismes de FM, a été décrite dans quelques travaux [3, 7]. Ainsi [7] introduit la notion de clone. L'identification de clones en tant que participants à une relation donnée au sein d'une configuration n'est cependant pas possible. Pour cela, nous avons besoin d'identifier les features sélectionnées : nous parlerons alors d'*instances*.

> *Dans une configuration, à une instance de sources doit correspondre une unique instance de méthode de rendu. Plusieurs instances de politiques peuvent en revanche être associée à une instance de source.*

Nous avons donc également besoin d'être en mesure d'exprimer des cardinalités sur les relations entre instances de features.

La validité d'une configuration repose donc à la fois sur la cardinalité des features au sein de la configuration et la cardinalité de la relation entre les instances au niveau de la configuration.

La variabilité s'exprime donc à plusieurs niveaux, les cardinalités des relations étant différentes suivant le niveau observé.

3.2 Multiplicité des processus de réalisation

> *Un directeur d'établissement construit son système en choisissant ses sources, puis les méthodes de rendu. Le responsable d'un institut pour personnes atteinte d'un handicap visuel construit son système en choisissant le design de l'affichage, puis les sources.*

Un FM est un arbre hiérarchisant des fonctionnalités, des concepts, impliquant donc de fait un ordre : on part du sommet et on descend les noeuds jusqu'à atteindre les feuilles. Cet ordre correspond aux sens des liens suivis par l'utilisateur pour la conception de son système : par exemple, il sélectionne une source qui est un noeud pour les méthodes de rendu valides pour cette source. Cependant, nos liens pouvant être parcourus dans un sens ou dans un autre selon la manière dont l'utilisateur effectue ses choix, cette hiérarchisation semble très problématique. Si l'utilisateur souhaite commencer par choisir une méthode de rendu qui est une feuille de l'arbre, il sélectionnera automatiquement la source parente correspondante. Si l'arbre est construit en considérant les sources comme des feuilles, le même problème apparait. La taille des arbres est si grande qu'ils sont très difficiles à maintenir, de plus les informations redondantes rendent cette solution peu convaincante.

De fait, nous avons pu constater qu'en fonction du contexte les choix de l'utilisateur se font simultanément sur plusieurs concepts (sources, rendus, design, etc) sans ordre pré-établi. Nous avons envisagé de construire dynamiquement ces arbres en utilisant des références de FM [4] afin de contourner cette difficulté. Cependant, la notion d'instance de feature, telle que définie ci-avant, pose des problèmes notamment pour la sélection des instances.

3.3 Interaction et Composition de features

> *Un utilisateur souhaite récupérer 10 photographies récentes, correspondant à un tag, sur FlickR, mais les afficher dans un ordre toujours différents. Il associe donc à sa source FlickR la politique Troncature pour récupérer 10 photographies et associe ensuite la politique Mélange pour mélanger les photographie.*
> *Un autre utilisateur souhaite récupérer 10 photographies au hasard parmi toutes les photographies d'un certain tag. Il associe donc également à sa source FlickR les politiques Troncature et Mélange...*

Une configuration mettant en jeu plusieurs features peut exiger la donnée d'informations supplémentaires qu'il convient de capturer pour guider les choix de l'utilisateur.

Il s'agit d'un problème de composition logicielle pour lequel de nombreuses recherches sont menées [8], mais nous avons également un problème d'expressivité pour l'utilisateur. En effet, la sémantique de la composition globale n'est absolument pas la même suivant l'ordre dans lequel chaque opération est effectuée, mais il est pourtant nécessaire de conserver au sein du FM chaque politique de manière séparée car elle peut être utilisée de manière unitaire.

L'utilisateur doit donc pouvoir conserver la capacité de ne sélectionner les politiques qu'une par une, tout en étant en mesure d'exprimer des opérations complexes qui font sens pour lui, et être guidé dans cette expression.

4 Agenda

Nous partons d'une étude de cas qui, si elle semble simple, s'avère particulièrement riche en variabilité. Si les LPL constituent un instrument spécifique à la gestion de la variabilité, le fait que celle-ci s'exprime dans notre système à plusieurs niveaux et en l'absence de processus de réalisation pré-établi, il semble difficile de les utiliser telles quelles.

Nous prévoyons pour être en mesure de proposer un outil fonctionnel à un utilisateur final, de réaliser un système fonctionnant à partir d'un modèle définissant les liens entre plusieurs LPL. En effet, chacun des concepts de notre système possédant sa propre variabilité, il semble intéressant de les exprimer à travers des LPL séparées et de permettre à l'utilisateur de les configurer de manière individuelle. Cependant, pour qu'il soit possible de le guider dans ses choix, nous devons être en mesure d'exprimer des liens entre les différents concepts capables d'agir aussi bien au niveau type qu'au niveau instance. Nous avons travaillé sur la formalisation de tels liens et sur la description formelle du système global au travers d'un métamodèle.

L'objectif étant de permettre la création d'un système fonctionnel, il est nécessaire d'associer aux différents éléments du système des assets logiciels afin d'être en mesure de générer l'application finale. La génération introduit à elle seule de nouvelles problématiques, particulièrement du point de vue de la composition de certains éléments et de la prise en compte des relations entre les FM.

Par ailleurs, notre système doit également être en mesure de guider l'utilisateur en fonction de propriétés non-fonctionnelles sélectionnées par l'utilisateur, comme par exemple le fait qu'il s'agisse d'un diffuseur d'information destiné à des personnes atteintes d'un handicap visuel. Cette problématique est loin d'être triviale en ce sens qu'elle peut avoir un impact transverse à tout le système. Le guidage de l'utilisateur en fonction de ses choix au travers de multiples FM va donc se matérialiser par des opérations complexes nécessitant des outils comme FAMILIAR [1]. Il peut s'agir de déselectionner définitivement des éléments du systèmes sur les FM ou, au contraire, de forcer certaines sélections et certaines associations, par exemple pour obliger le choix d'une politique de cache associée à chaque source dans le cadre d'un milieu mal connecté.

Enfin, un dernier axe de recherche, non des moindres, porte sur la maintenabilité et l'évolutivité du système global. En effet, le projet YourCast a pour objectif d'être réalisé par incréments et d'être ensuite mis à disposition d'une communauté afin de continuer à évoluer. Il doit donc être facile d'ajouter de nouveaux éléments et de nouvelles contraintes tout en continuant à garantir la fiabilité du système.

5 Conclusion

Le projet ANR EMERGENCE YourCast est un projet qui, sur deux ans, vise à produire un outil fonctionnel capable de guider un utilisateur dans la réalisation d'un système de diffusion d'informations. Cet objectif d'apparence modeste recouvre en réalité un grand nombre de problématiques diverses, tant au niveau de la façon de guider l'utilisateur dans ses choix, qu'au niveau de la génération du produit final. Nous avons décidé de nous concentrer dans un premier temps sur la manière de guider l'utilisateur et la façon dont il pouvait exprimer ses choix à partir d'une LPL.

Cette première approche nous a amené à constater que la variabilité des systèmes de diffusions étaient très importante et qu'il ne semblait pas pertinent de la conserver dans un unique modèle. En effet, nous avons pu constater que l'utilisation d'une unique LPL était une approche trop limitée pour permettre la modélisation des différents niveaux de relations dont nous avons besoins. Par ailleurs, la hiérarchisation même des LPL impose un processus de réalisation dont nous souhaitons pouvoir nous passer en raison des trop nombreux cas d'utilisation possibles.

C'est pourquoi nous nous orientons vers une approche utilisant un ensemble de LPL liées entre elles au sein d'un modèle dont nous sommes encore en train de formaliser le méta-modèle.

References

[1] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. Separation of concerns in feature modeling: support and applications. In *Proceedings of the 11th annual international conference on Aspect-oriented Software Development*, AOSD '12, pages 1–12, New York, NY, USA, 2012. ACM.

[2] Hugo Arboleda, Rubby Casallas, and Jean-Claude Royer. Dealing with fine-grained configurations in model-driven SPLs. In *SPLC*, pages 1–10, 2009.

[3] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Formalizing Cardinality-based Feature Models and their specialization. *Software Process: Improvement and Practice*, 10(1):1–25, 2005.

[4] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models. *Software Process: Improvement and Practice*, 2005.

[5] Øeystein Haugen, B. Møller-Pedersen, Gøran K. Olsen, Andreas Franck Svendsen, Franck Fleurey, and Xiaorui Zhang. D2.1.4 – Consolidated CVL language and tool. Technical report, SINTEF, University of Oslo, 2010.

[6] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William A. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical Report November, The Software Engineering Institute, 1990.

[7] Raphael Michel, Andreas Classen, Quentin Boucher, and Arnaud Hubaux. A Formal Semantics for Feature Cardinalities in Feature Diagrams. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, pages 82–89. ACM, 2011.

[8] Sébastien Mosser, Carlos Parra, Laurence Duchien, and Mireille Blay-Fornarino. Using domain features to handle feature interactions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, pages 101–110. ACM, 2012.