CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Open Repository and Bibliography - Luxembourg

A Change Management Approach in Product Lines for Use Case-Driven Development and Testing

Ines Hajri, Arda Goknil, and Lionel C. Briand

SnT Centre for Security, Reliability and Trust, University of Luxembourg
{ines.hajri, arda.goknil, lionel.briand}@uni.lu

Abstract. In this paper, driven by industrial needs, we present a change management approach for product lines within the context of use case-driven development and testing. As part of the approach, we first provide a modeling method to support variability modeling in Product Line (PL) use case diagrams, specifications, and domain models, intentionally avoiding any reliance on feature models and thus avoiding unnecessary modeling and traceability overhead. Then, we introduce a use case-driven configuration approach based on the proposed modeling method to automatically generate Product Specific (PS) use case and domain models from the PL models and configuration decisions. Building on this, we provide a change impact analysis approach for evolving configuration decisions in PL use case models. In addition, we plan to develop a change impact analysis approach for evolving PL use case models and an automated regression test selection technique for evolving configuration decisions and PL models.

Keywords: Product Line Engineering, Use Case-Driven Development, Change Impact Analysis, Regression Test Selection

1 Introduction

Product Line Engineering (PLE) is becoming widely adopted in many software development environments to develop complex systems for multiple customers with varying needs. In many of these development environments, use cases are central development artifacts and are used for communicating requirements among stakeholders and for system test case generation [1] [2]. We face the same situation with our industrial partner IEE S.A. [3], a leading supplier of embedded systems in the automotive domain.

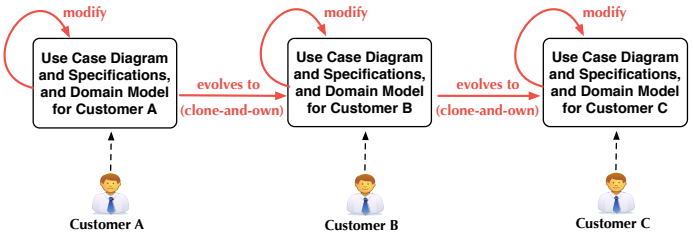

Fig. 1. Clone-and-Own Reuse at IEE for a Product Family

The current use case-driven development practice at IEE, like in many other environments, is based on clone-and-own reuse [4] (see Fig. 1). IEE starts a new project with an initial customer. The product requirements are elicited from the initial customer and documented as a use case diagram, use case specifications, and a domain model (i.e., a conceptual model that represents domain entities). For each new customer in the product family, the IEE analysts need to clone the current models, and negotiate variabilities with the customer to produce a new use case diagram, new specifications, and a new domain model (see *clone-and-own* in Fig. 1). As a result of the negotiations, the IEE analysts make changes in the cloned models (see *modify* in Fig. 1). They derive and select new system test cases from the modified use cases (i.e., regression test selection for system test cases). With such practice, variants and variation points (i.e., where potential changes are made) are not documented and the IEE analysts, together with the customer, need to evaluate the entire use cases, domain model and test cases.

There are approaches in the literature that address the need for supporting PLE in the context of use case-driven development and regression test selection (e.g., [5] [6] [7] [8] [9]). Most of the use case-driven PLE approaches require that feature models be traced to use case diagrams and specifications (e.g., [10] [11] [12] [13] [14] [15]). The analysts need to establish traces between feature models and use cases. The evolution of feature models and use cases also requires these traces to be maintained manually [16]. There are approaches [17] [18] [19] [20] that study the evolution of feature models without addressing the impact of changes of variability information on use case models. In addition, due to limited resources, many companies, such as IEE, find the traceability and maintainability effort incurred with feature models to be impractical.

This paper outlines our approach providing automated support for the change management of use case models and regression test selection in a product family.

2 Overview

We present an overview of our proposed approach in Fig. 2. The analyst produces three artifacts: a Product Line (PL) use case diagram, PL use case specifications, and a PL domain model (see *Elicit Product Line Use Case Models* in Fig. 2). It may not always be possible to model PL use cases without starting from product use cases; the analyst elicits use cases of a specific product, and then identifies variabilities and commonalities for the product family. We observe that most of the projects in industry start with an initial customer for which the product is designed and produced. Other potential customers are typically engaged after the release of the initial product. At this phase of product development, the analyst starts identifying commonalities and variabilities of the product family based on the use cases of the initial product.

With new customers (see 'evolves to' in *b* and *d* in Fig. 2), the analyst is asked to input configuration decisions regarding variation points captured in PL use case and domain models to automatically configure the product line into a product (see 'configure' in *b*, and *d* in Fig. 2). The configuration of Product Specific (PS) use case and domain models is an automated, iterative, and interactive decision-making activity. When a decision is made, the consistency of the decision with prior decisions is checked. There might be contradicting decisions in the PL use case diagram such as two decisions re-

sulting in selecting variant use cases violating some dependency constraints. These need to be automatically determined and reported *a posteriori* and the analysts can backtrack and revise their decisions.

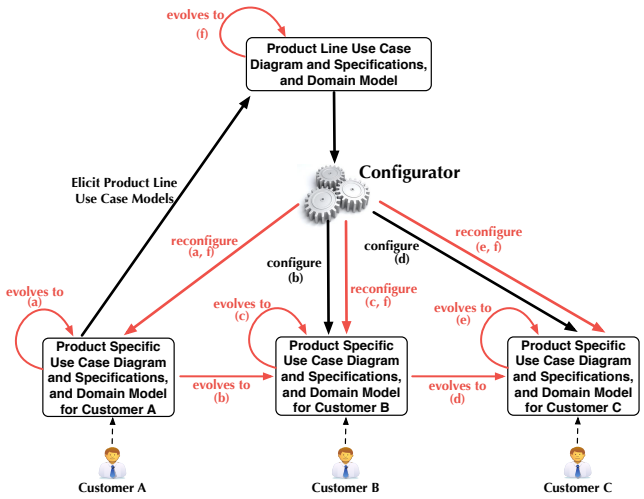

Fig. 2. Overview of the Approach

Configuration decisions may frequently change, resulting in the reconfiguration of PS use case models (see 'evolves to' in *a*, *c* and *e* in Fig. 2). Impacted decisions, i.e., subsequent decisions to be made and prior decisions cancelled or contradicting when a decision changes, are identified to incrementally reconfigure the generated use case models (see 'reconfigure' in *a*, *c* and *e* in Fig. 2).

Reconfiguration is needed not only for changes in configuration decisions (see 'evolves to' in *a*, *c*, and *e* in Fig. 2) but also for changes in PL use case models (see 'evolves to' in *f*). The former requires reconfiguration only for the product concerned with the decisions, while the latter needs an impact assessment method to analyze change impact on PL use cases before reconfigurating use case models for all products in the product family (see 'reconfigure' in *f* in Fig. 2).

When requirements evolve in a product family (see 'evolves to' in *a*, *b*, *c*, *d*, *e*, and *f* in Fig. 2), the change impact on the execution of system test cases derived from these requirements need to be assessed. We automatically choose, from an existing test set, test cases that can and need to be rerun to ensure existing, unmodified functionalities are still working correctly.

Most parts of the proposed approach in Fig. 2 have already been implemented and assessed (see Section 3), while some other parts are planned to be developed as a future work (see Section 4).

3 Current Results and Tool Support

Product line use case modeling method. We proposed, applied, and assessed our Product line Use case modeling Method (PUM) [21] to support variability modeling in PL use case diagrams, specifications, and domain models, without making use of feature

models, thus avoiding any modeling overhead (see *Elicit Product Line Use Case Models* in Fig. 2). PUM adopts existing PL extensions [22] [23] for use case diagrams and domain models. For modeling variability in use case specifications, we introduced new product line extensions for the Restricted Use Case Modeling method (RUCM) [24].

Use case-driven configuration approach. We proposed, applied, and assessed a use case-driven configuration approach [25] using PUM (see 'configure' in *b* and *d* in Fig. 2). The approach supports three activities. First, the analyst is guided to make configuration decisions in an appropriate order. Second, the consistency of configuration decisions is ensured by automatically identifying contradicting decisions. Third, PS use case and domain models are automatically generated from PL models and decisions.

A change impact analysis approach for evolving configuration decisions. We proposed, applied and assessed a change impact analysis approach, on top of our use case-driven modeling and configuration techniques, to support the evolution of configuration decisions (see 'evolves to' and 'reconfigure' in *a*, *c* and *e* in Fig. 2). Our approach supports three activities. First, the analyst proposes a change but does not apply it to the corresponding configuration decision. Second, the impact of the proposed change on other configuration decisions for the PL use case diagram, the PL use case specifications and the PL domain model are automatically identified. Third, the PS use case and domain models are incrementally regenerated only for the impacted decisions after the analyst makes all the required changes. We implemented a model differencing and reconfiguration pipeline [26] where we identify the changed decisions for which we need to regenerate the PS use case and domain models.

The current results have been implemented as a tool, PUMConf [27]. PUMConf uses GATE (`http://gate.ac.uk/`), an open source NLP framework, to annotate PL use case specifications to be used for (re)configuring PS use case specifications. PUMConf relies upon: (i) *IBM DOORS* to model PL use case specifications and (ii) *Papyrus* to model and save PL use case diagrams as a UML file. To load use cases from IBM DOORS, it uses DOORS Document Exporter, an API that exports the DOORS content as text files. The (re)configuration of PS use case models has been implemented as a Java application. The DOORS eXtension Language (DXL) is employed to load the configured PS specifications into DOORS. PUMConf is approximately 25K lines of code, excluding comments and third-party libraries. PUMConf has been evaluated over an industrial case study. The evaluation showed that our tool is practical and beneficial to (re)configure PS use case and domain models in industrial settings [25] [26]. Additional details about PUMConf, including executable files and a screencast covering motivations, are available on the tool's website at `https://sites.google.com/site/pumconf/`.

4 Future Work

A change impact analysis approach for evolving PL use case and domain models. Change can occur in variability aspects of PL use case and domain models, e.g., adding a new variation point (see 'evolves to' in *f*). Changes on PL use case models require impact assessment on other parts of PL models and configuration decisions for each individual product, and may entail reconfiguration (see 'reconfigure' in *f* in Fig. 2). To

this end, we plan to develop a change impact analysis approach to identify impacted configuration decisions and parts of PS models that need to be reconfigured when PL use case and domain models evolve.

A regression test selection approach for product lines. Regression test selection is a particular application of change impact analysis, that consists in choosing, from an existing test set, test cases that can and need to be rerun to ensure existing, unmodified functionalities are still working correctly [28]. We plan to provide an automated regression test selection approach for system test cases derived from use case models. For system test cases and their traces to use case models, we plan to rely on the Use Case Modeling for System Tests Generation approach (UMTG) [2] [29], that automatically generates executable system test cases from PS use case and domain models.

5 Conclusion

We presented a change management approach for product lines within the context of use case-driven development and testing. As part of the approach, we provided (i) a product line modeling methodology centered around use case modeling for documenting variability in PL use case diagrams and specifications, and associated domain models, (ii) a configuration approach that automatically generates PS use case and domain models from PL models and configuration decisions, and (iii) a change impact analysis approach for evolving configuration decisions in PL use case models. In the future, we plan to develop a change impact analysis approach for evolving PL use case models and a regression test selection technique for evolving PL use case models and configuration decisions. We further plan to conduct more case studies to better evaluate the practical utility and usability of our techniques.

Acknowledgments. Financial support was provided by IEE and FNR under grants FNR/P10/03 and FNR10045046.

References

1. C. Nebut, Y. L. Traon, and J.-M. Jezequel, "System testing of product families: from requirements to test cases," in *Software Product Lines*. Springer, 2006.
2. C. Wang, F. Pastore, A. Goknil, L. C. Briand, and M. Z. Z. Iqbal, "Automatic generation of system test cases from use case specifications," in *ISSTA'15*, 2015, pp. 385–396.
3. "IEE (International Electronics & Engineering) S.A., http://www.iee.lu/."
4. P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.
5. V. Alves, N. Niu, C. Alves, and G. Valença, "Requirements engineering for software product lines: A systematic review," *Information and Software Technology*, vol. 52, pp. 806–820, 2010.
6. E. Engstrom and P. Runeson, "Software product line testing - a systematic mapping study," *Information and Software Technology*, vol. 53, pp. 2–13, 2011.
7. E. Engstrom, "Regression test selection and product line system testing," in *ICST'10*, 2010, pp. 512–515.
8. E. Engstrom, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

9. S. Wang, S. Ali, A. Gotlieb, and M. Liaaen, "Automated product line test case selection: Industrial case study and controlled experiment," *Software and Systems Modeling*, 2015.
10. R. Bonifácio and P. Borba, "Modeling scenario variability as croscutting mechanisms," in *AOSD'09*, 2009, pp. 125–136.
11. R. Bonifácio, P. Borba, and S. Soares, "On the benefits of scenario variability as croscutting," in *EA-AOSD'08*, 2008, pp. 1–6.
12. M. Eriksson, J. Borstler, and K. Borg, "The pluss approach - domain modeling with features, use cases and use case realizations," in *SPLC'05*, 2005, pp. 33–44.
13. M. Eriksson, H. Morast, J. Borstler, and K. Borg, "The pluss toolkit - extending telelogic doors and ibm-rational rose to support product line use case modeling," in *ASE 2005*, 2005, pp. 300–304.
14. M. Eriksson, J. Borstler, and K. Borg, "Managing requirements specifications for product lines - an approach and industry case study," *Journal of Systems and Software*, vol. 82, pp. 435–447, 2009.
15. "pure::variants, http://www.pure-systems.com/pure_variants.49.0.html."
16. G. Botterweck and A. Pleuss, "Evolution of software product lines," in *Evolving Software Systems*. Springer, 2014.
17. J. White, J. A. Galindo, T. Saxena, B. Dougherty, D. Benavides, and D. C. Schmidt, "Evolving feature model configurations in software product lines," *Journal of Systems and Software*, pp. 119–136, 2014.
18. T. Thum, D. Batory, and C. Kastner, "Reasoning about edits to feature models," in *ICSE'09*, 2009, pp. 254–264.
19. J. Burdek, T. Kehrer, M. Lochau, D. Reuling, U. Kelter, and A. Schurr, "Reasoning about product-line evolution using complex feature model differences," *Automated Software Engineering*, 2015.
20. A. Pleuss, G. Botterweck, D. Dhungana, A. Polzer, and S. Kowalewski, "Model-driven support for product line evolution on feature level," *Journal of Systems and Software*, vol. 85, pp. 2261–2274, 2012.
21. I. Hajri, A. Goknil, L. C. Briand, and T. Stephany, "Applying product line use case modeling in an industrial automotive embedded system: Lessons learned and a refined approach," in *MODELS'15*, 2015, pp. 338–347.
22. G. Halmans and K. Pohl, "Communicating the variability of a software-product family to customers," *Software and Systems Modeling*, vol. 2, pp. 15–36, 2003.
23. T. Ziadi and J.-M. Jezequel, "Product line engineering with the uml: Deriving products," in *Software Product Lines*. Springer, 2006.
24. T. Yue, L. C. Briand, and Y. Labiche, "Facilitating the transition from use case models to analysis models: Approach and experiments," *ACM Transactions on Software Engineering and Methodology*, vol. 22, no. 1, 2013.
25. I. Hajri, A. Goknil, L. C. Briand, and T. Stephany, "Configuring use case models in product families," *Software and Systems Modeling*, 2016.
26. ——, "Incremental reconfiguration of product specific use case models for evolving configuration decisions," in *REFSQ'17*, 2017.
27. ——, "PUMConf: a tool to configure product specific use case and domain models in a product line," in *SIGSOFT FSE'16*, 2016, pp. 1008–1012.
28. L. C. Briand, Y. Labiche, and S. He, "Automating regression test selection based on UML designs," *Information and Software Technology*, vol. 51, no. 1, pp. 16–30, 2009.
29. C. Wang, F. Pastore, A. Goknil, L. C. Briand, and M. Z. Z. Iqbal, "UMTG: a toolset to automatically generate system test cases from use case specifications," in *ESEC/SIGSOFT FSE'15*, 2015, pp. 942–945.
30. I. Hajri, "Supporting change in product lines within the context of use case-driven development and testing," in *SIGSOFT FSE'16*, 2016, pp. 1082–1084.