RESOLVING TRACEABILITY ISSUES IN PRODUCT DERIVATION FOR SOFTWARE PRODUCT LINES

Saad Bin Abid
Lero-The Irish Software Engineering Centre, University of Limerick, Limerick, Ireland

Keywords: Traceability, Change management, Model-driven development, Software product lines, Product derivation.

Abstract: Dealing with traceability management issues during model based product derivation in large complex industrial SPL is error prone due to the lack of tool support. As a result traceability management between connected models emerges as an important research topic. In this position paper, we discuss research challenges as scenarios from developed example product line and give recommendations on resolving traceability issues during product derivation. We also discuss initial ideas about our proposed approach for resolving traceability issues for efficient change management. It is foreseen that the proposed traceability management recommendations will help to understand the traceability issues during product derivation and as a result of implementing them will help us to get a bit closer to our ultimate goal of, 1) efficiently automate the product derivation, 2) reduce the production cost, 3) improve productivity and 4) improve change management in SPL.

1 INTRODUCTION

Software Product Lines (SPLs) allows companies to realize significant improvements in time-to-market, cost, productivity, and system quality (Clements et al., 2002). One major difficulty with SPL engineering is to deal with thousands of variation points in the industrial size product line. These variant points need special attention as they add complexity during product configuration (PC) and product derivation (PD).

In a large SPL, software artefact traceability is an important factor when it comes to effective development and maintenance of software system; due to lack of automation support for traceability, maintaining links between artefacts is a tedious and time consuming job. According to (Antoniol et al., 2006) traceability links between related artefacts need to evolve synchronously and reflect current changes and dependencies across multiple artefact types. Traceability management facilitates the SPL artefacts to remain in synchronous state and ensures the consistency of derived products. This research paper discusses and elaborates on our ideas of doing traceability management during the product derivation and motivates where traceability can be useful and recommend the SPL community challenges which need attention.

This paper is structured into following sections. Section 2 discusses the basic terminology of traceability and motivates the use of traceability in Model Driven Software Development (MDSD) and SPL. Section 3 elaborates PD tasks. Section 4 discusses how the introduction of traceability can facilitate the product engineer and automate the PD. Section 5 provides related work. Section 6 is discussion and future work. Section 7 discusses the research evaluation plan and some limitation of proposed research prototype tool suite. Section 8 concludes the paper.

2 MOTIVATION OF USING TRACEABILITY IN MDD AND SPL

In this section, we are going to discuss the traceability terminology and motivates use of traceability in both model-driven software development (MDSD) and software product lines (SPLs). The term traceability has different meanings in different contexts. In the context of software artefact traceability is “the ability to relate the different artefacts created in the development life cycle with one another” (Ajila et al., 2004).

Abid S. (2009).
RESOLVING TRACEABILITY ISSUES IN PRODUCT DERIVATION FOR SOFTWARE PRODUCT LINES.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 99-104
DOI: 10.5220/0002239400990104

Traceability is an important challenge in model driven development (MDD) (ECMDA website, 2006). One critical challenge in MDSD processes is the traceability of the requirements throughout the development life cycle and on different levels of abstraction (Aleksy et al., 2008). Despite the fact MDD paradigm involves automatic transformations. Traceability is still an open issue due management of large number of software development artefacts, relationships and dependencies among them, these factors make traceability more time consuming and error prone (Egyed, A., 2003).

The need of maintaining traces among artefacts to support change management in software development is well documented in the literature (Ramesh et al., 2001). Prior Literature also describes that poor traceability leads to adverse impact (e.g. decrease in system quality, increase in number of changes, loss of crucial knowledge due to turnover, erroneous decisions, misunderstanding and miss communication) on project cost and schedule (Ralf et al., 1998).

Traceability helps identifying relationships and dependencies among artefacts. Traceability between requirements and their representation in the models is crucial to ensure that the relevant set of requirements is accurately elicited and eventually implemented in the code (Aleksy et al., 2008). Not only traceability ensures identifying related artefacts and elements but it also can facilitate the change impact analysis during software development.

3 PRODUCT DERIVATION TASKS

In this section we are going to discuss PD tasks in a form of scenarios and identify the potential areas for use of traceability. Many SPL research approaches (Czarnecki et al., 2004) focus on single development artefacts. In order to exploit real benefits of a product line we need to connect these isolated models (Botterweck et al., 2008). The authors have identified potential PD tasks in a form of scenarios. The scenarios are illustrated with a sample product line (RESCU product line (Botterweck et al., 2008)). Decision, Feature and Component models are modeled artefacts from the RESCU product line for automotive restraint systems. The following sections discuss few scenarios in detail as well as identify potential areas for traceability to help the PD tasks.

3.1 Change Analysis and Evolution of SPL (Scenario 1)

Change can arise during PD. If the introduced change is not catered it can lead to less productivity. Introduced change needs special attention when it comes to understanding the consequences of the change during PD.

In the next sections, we discuss introduction of change Impact analysis, artefacts evolution and consistency, configured product variant validation and testing identification of unstable product variants and 3rd part components during PD and also identify potential area use of traceability.

3.1.1 Impact Analysis (Scenario 1.1)

During PD change can occur due to the following reasons i) Inclusion of newly built artefacts into the reusable assets platform (product line level), ii) Change in customer requirements (product level), iii) Updating the artefacts in a form of versions (product level).

In a product line above mention changes can occur in a form of addition of new decisions, features, and components. The impact of change needs to be analyzed for change management during PD.

3.1.2 Related Artefacts Evolution and Consistency (Scenario 1.2)

The newly built artefacts need to be added to the core assets platform for future reuse. Evolving related artefacts synchronously is still an open issue as current traceability schemes are focusing on one time snapshot of related artefacts (Murta et al., 2008). For instance in example application product line (RESCU) evolution is required when changes are introduced to any of the modelled artefacts in a form of either adding new decision in the decision model, addition/ deletion of features (functional requirement) in feature model or addition/ deletion/updating of components in component model.

Consistency on both product line and model level is necessary for derived products to be consistent and PD task to be less error prone. Modeling level consistency corresponds to the consistency between instantiated decision, feature and component models. And elements level corresponds to consistency that all the related decision, feature and component models elements are realized/implemented by their corresponding elements.

3.1.3 Configured Product Variant Validation and Testing (Scenario 1.3)

During PD product variants need to be validated and tested accordingly with respect to the customer's requirements. One scenario can be when new variants are generated and reused again during PD. In this situation the related test cases needs to be identified for their testing and validation.

3.1.4 Identification of Unstable Product Variants and 3rd Party Components (1.4)

During PD the reusable product variants are tailored by resolving the variability according to customer's requirements. Change is introduced in product variants in a form of variants generation or updating of variation points. Highly unstable component can be defined as those components that require multiple changes. During the PD planning the product engineer might also want to analyze the highly unstable components which are of critical importance during PD.

Introduction of commercially of the shelf components introduce change during PD. One scenario could be when a customer wants to introduce either a hardware (e.g. navigation system) or software component (e.g. Oracle database) in the final product variant. As a result of introduction of new component the product engineer wants to analyze which hardware or software components are affected.

3.2 Resolution of Variability Across Related Artefacts (Scenario 2)

During PD which is an application engineering process, a product variant is derived from core platform artefacts/ core assets. The problem arises when different versions of same artefact elements are developed and updated and this variability point (representing variability in an artefact) is not implemented in corresponding related artefact.

3.3 Identification of Related Documentation (Scenario 3)

In this scenario, during PD in a product line the customer gives his/her requirements to the product engineer and PD slows down because product engineer doesn't understand particular feature or related component functionality. In this situation the product engineer wants to refer to feature or components documentation.

4 FACILITATING PRODUCT DERIVATION TASKS USING TRACEABILITY

In the previous section we discussed some of the potential candidate tasks during PD. In this section we give suggestions as to how introduction of the traceability can help in automating the PD tasks.

4.1 Interactive Visualisation of Traceability Links (Approach 1)

The visualization of traceability links facilitates the issue of visualization of related artefacts elements. For example, when during PC only the related elements and rest of the artefacts elements hide might be one visual effect that can facilitate the product engineer to stay focused and understand the consequences of the decisions made by product engineer. Providing a focus to a certain area in the model also enhances the understanding of dependencies between related elements. Visualizing and reporting an error and providing a guideline to solve an error can increase the productivity level.

4.2 Semantically Enriched Traceability Meta-Model

It is hard to define the term ``semantics'' of any language. Semantics gives the meaning to the language constructs which define the models in MDD. Semantics of traceability links is responsible for defining the constructs of the traceability links. Semantically correct traceability links ensure the consistency of related product line artefacts by capturing more meaningful traceability information. There are different languages for defining the semantics for instance Meta modeling languages (OMG group, 2005; Eclipse website, 2009), Higher order logic (Janota et al., 2007; Mannion, 2002), Propositional logic (Batory, 2005).

Figure 1 shows traceability Meta-model established by us. Meta-model contains "Traceable Artefact" class. Traceable artefact acts as core platform containing different artefacts. Different artefacts are traceable to each other via the class "Artefact Trace". "Artefact Trace" class has attributes like "TId" (identifying unique artefacts), "TFreq" (for calculating frequency of artefact

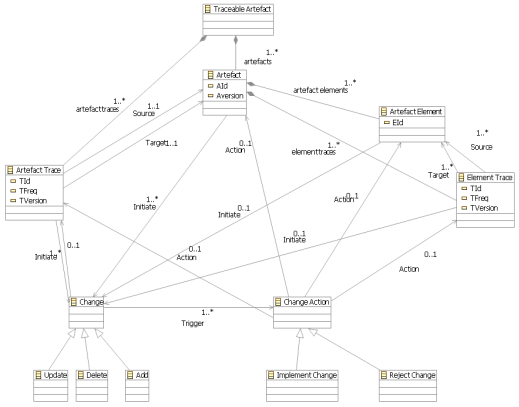

Figure 1: Suggested traceability Meta model.

traces), "TVersion" (which artefact versions are traceable). Artefact consists of different artefact elements, these artefact elements are traceable with each other via the class "Element Trace". Classes "Artefact", "Artefact Trace", "Artefact Element" and "Element Trace" can initiate change which is implemented in class "Change". Change can be of different types. For simple version of traceability Meta model we have change types as Update, Delete and Add. Initiating the change leads to trigger the Change Action class. Change Action class is responsible for analyzing the change and can either implement the change (Implement Change class) or reject the change (Reject Change). This change can lead to either implementing change in a form of adding, deleting or updating related element or just rejecting the change after analyzing the change request.

Attributed traceability links means that traceability links should not be just acting like pointers to relate source and target artefact elements and vice versa but also should contain additional information (e.g., probability of variant selection, uncertainty factor for related artefacts elements). During PD in large complex product line the probability can be determined by frequency of reuse of a hardware or software component. This frequency of reuse can be added as additional information to the traceability links.

4.3 Documenting Traceability Links (Approach 3)

Documentation of traceability links facilitates the product engineer to analyze which artefacts elements are connected to each other along with their frequency of reuse during previous variant derivations. It is also possible that documenting traceability links might increase the overhead for maintaining the product line. But during PD in large complex product lines where thousands of artefacts elements are connected the product engineer might want to click the traceability link and get the information of related artefacts elements. Documenting traceability links will not only save his time for analysis but also can serve as planning for next reuse, as will save all the information captured in traceability links (e.g. Versioning, updating source destination artefacts.

5 RELATED WORK

Work by (Murta et al., 2005) presents architecture to implementation traceability links evolution. The approach has provided a nice start to manage the traceability links between architecture to implementation but still the introduction of new artefacts and traceability management among them is not yet elaborated as in our case we are dealing with traceability management of multiple models over time. Work by (Asikainen et al., 2007) supports both the Domain Engineering process and the Application Engineering Processes. Kumbang does link the features to the architecture. Unfortunately, there exists no direct visual, interactive representation of the relationships between these models. The guidance to resolve the conflict during PD is not provided, which makes PD error prone and human intensive. Pure::Variants (Beuche, 2004) is a commercially available feature modelling tool. It supports various views which provide different approaches for different stakeholder tasks but does not support cardinality. Pure::Variants is restricted to only two models and it is not mentioned that if other SPL artefacts can be traceable and managed. It is also not possible to navigate to change and analyze the consequences of introduced change. The GenArch (Cirilo et al., 2007) approach is model-based tool support for PD. The approach is lacking traceability between feature and architecture models and relies on Java annotations to provide the traceability information which is not a flexible and robust approach. It is not possible to identify, navigate and analyze the change. Also introduction of new SPL artefact is not elaborated. Work by (Satyananda et al., 2007) identifies traceability between feature model and architecture model using Formal Concept Analysis (FCA) approach. There is no visualization or tool support available for the approach. Hence identification and analysis of introduced change during PD is not possible.

DOPLER tool suite (Rabiser et al., 2007) is integrating decision model and asset model (based on domain specific meta model) with the inclusion condition relationships between them. The complex relationships between decisions and assets are expressed in a simple rule language. Self developed engine is currently used for serving the purpose. Change identification and management using traceability is not elaborated also how to manage traceability when between pre-existing SPL artefacts (e.g. code, documentation) is not described. AMPLE project (AMPLE website, 2009) is aiming to apply MDD techniques in SPL area. AMPLE project is not treating traceability management for change management as saparete issue.

6 DISCUSSION AND FUTURE WORK

From Table 1 we can list that not every approach is supporting all the scenarios, rather subsets of the scenarios can be supported by each of the suggested approach. We can also deduce from the table that one can develop such approach which is union of all the suggested approaches in order to facilitate most of the scenarios identified by the authors during PD.Initial sample product line for Scientific Calculator (SciCalc-PL) has been developed. The proposed architecture for prototype tool will be based on the Eclipse modelling framework (EMF) and will be taking care of change management during PD in SPL. Figure 2 shows the plug in architecture of the proposed prototype. The proposed case study will lead to initial prototype development for automating PD in related artefacts for traceability links extraction and consistency checking for better change management using a semantically enriched traceability Meta model established in Section 4.2. The Prototype will consist of set of Plug-ins (AutoMapper, ConsistencyChecker, ChangeManagement, Visualization).

Table 1: Suggested approach and scenarios supported.

Suggested Approach	Scenarios Supported
Approach 1	Scenario 1.1, 1.4
Approach 2	Scenario 1.1, 1.2, 1.3, 2, 1.4
Approach 3	Scenario 1.4, 1.3

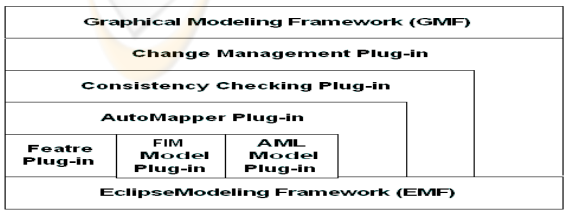

Figure 2: Prototype Architecture.

7 RESEARCH PROTOTYPE EVOLUTION PLAN AND LIMITATIONS

This section is going to discuss the initial evaluation plan and limitations of the approach discussed in Section 6. The main focus of proposed solution is to provide an EMF plug-in based tool suite prototype for traceability management during PC and PD in SPL. It is expected that the proposed approaches (Section 4) will be supported by the prototype in order to help automate the PD tasks identified in Section 3.

Tool suites (e.g., Pure::Variants etc) and research prototypes (e.g. Kumbang and GenArch etc) are available for PD but they are not mainly focusing on traceability management. Our approach will purely address and resolve the challenges of traceability management for change management during PD in SPLs. Initial set of plug-ins (Feature, FIM model and AML model plug-ins) for SciCalc-PL are developed. The proposed Meta model will be the basis for developing plug-ins (AutoMapper, Consistency Checker and Change Management). For documenting the traceability links it is in our agenda to use Model to Text transformation languages like open architecture ware (oAW). For visualising the traceability links we intent to use Graphical Modeling Language (GMF). Since our approach will result in a plug-in based prototype tool suite and each prototype will be providing different functionality. The proposed research prototype tool suite however has some limitations. Our approach will be focusing on automatically generating traceability links based on existing model based traceability extraction techniques. Also the approach will be using existing consistency checking techniques and change management for model based product lines in PD.

8 CONCLUSIONS

This position paper presented different recommendations for using traceability management

for efficient PD tasks in large scale software product lines. The proposed recommendations focus on providing interactive visual support, semantics and documentation for traceability links. The recommendations given to support PD tasks are general in nature but of importance. A traceability Meta model is also been proposed. We believe that by implementing the provided recommendations for change analysis, validation, components selection/elimination, identification of high risked feature, unstable components can be supported in a large scale PD environment. Initial plug-in architecture of proposed prototype is also provided (Figure 2). Initial ideas of proposed approach evaluation plan and limitations are also discussed.

ACKNOWLEDGEMENTS

This work is partially supported by Science Foundation Ireland under grant number 03/CE2/I303-1.

REFERENCES

Clements, P. and Northrop, L. M. (2002). *Software Product Lines: Practices and Patterns*. Boston: Addison-Wesley.

Antoniol, G., Berenbach, B., Eyged, A., Ferguson, S., Maletic, J., Zisman, A., Holbrook, E. A., Sundaram, S., Zou, C. and Gotel, O. (2006). "Center of Excellence of Traceability Technical Report " Center of Excellence for Traceability.

Ajila, S. A. and Kaba, A. B. (2004), "Using traceability mechanisms to support software product line evolution," in *Information Reuse and Integration,. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pp. 157-162.

ECMDA Website (2006), "ECMDA Traceability Workshop ", Homepage: http://www.modelbased.net/ecmda-traceability/.

Ramesh, B. and Jarke, M.(2001). "Toward reference models for requirements traceability," *IEEE Transaction*.

Aleksy, M., Hildenbrand, T., Obergfell, C., and Schwind, M. (2008). "A Pragmatic Approach to Traceability in Model-Driven Development," in *PRIMIUM 2008 Process Innovation with Business Software*, Garching, Germany.

Egyed, A. (2003). "A scenario-driven approach to trace dependency analysis," *Software Engineering, IEEE Transactions on,* vol. 29, pp. 116-132.

Czarnecki, K. , Helsen, S. and Eisenecker, U. W. (2004). "Staged Configuration Using Feature Models," in *Proceedings of the Third Software Product Line Conference (SPLC 2004)*, pp. 266-283.

Botterweck, G. , Thiel, S. , Nestor, D. , Abid, S. bin and Cawley, C. (2008). "Visual Tool Support for Configuring and Understanding Software Product Lines," in *12th International Software Product Line Conference (SPLC 2008)*, Limerick, Ireland.

Murta, L. G. P., van der Hoek, A. e. and Werner, C. a. u. M. L. (2008). "Continuous and automated evolution of architecture-to-implementation traceability links," *Autom. Softw. Eng.,* vol. 15, pp. 75-107.

O M G Group (2005). "Revised submission for MOF 2.0 Query, View, Transformation version 2.0".

Eclipse website (2009). "EMF - Eclipse Modelling Framework." Homepage: http://www.Eclipse.org/

Janota, M. and Kiniry, J. (2007). "Reasoning about Feature Models in High-Order Logic," in *SPLC 2007* Kyoto, Japan.

Mannion, M. (2002). "Using First-Order Logic for Product Line Model Validation," in *Proceedings of the Second Software Product Line Conference*, 2002, pp. 176-187.

Batory, D. (2005), "Feature Models, Grammars, and Propositional Formulas," in *9th International Conference on Software Product Lines (SPLC 2005)*, Rennes, France, pp. 7-20.

Asikainen, T., Männistö, T., Soininen, T. (2007). "Kumbang: A domain ontology for modelling variability in software product families," *Adv. Eng. Inform.,* vol. 21, pp. 23-40, 2007.

Beuche, D. (2004). "Variants and Variability Management with pure::variants," in *3rd Software Product Line Conference (SPLC 2004), Workshop on Software Variability Management for Product Derivation*, Boston, MA, 2004.

Cirilo, E. , Kulesza, U. and Lucena, C. J. P. d. (2007). "GenArch A Model-Based Product Derivation Tool,".

Satyananda, T. K. , Lee, D. and Kang, S. (2007). "A Formal Approach to Verify Mapping Relation in a Software Product Line," *7th IEEE International Conference on Computer and Information Technology, 2007(CIT 2007)*, pp. 934-939.

Rabiser, R. , Dhungana, D. and Grünbacher, P. (2007) "Tool Support for Product Derivation in Large-Scale Product Lines: A Wizard-based Approach," in *1st International Workshop on Visualization in Software Product Line Engineering (ViSPLE 2007)*, Kyoto, Japan.

Ralf, D. and Klaus, P. (1998). "Adapting traceability environments to project-specific needs," *ACM Community* pp. 54-62.

Ample project (2009). Web site. http://ample.holos.pt/