Modeling and Managing Tacit Product Line Requirements Knowledge

Reinhard Stoiber, Martin Glinz
Department of Informatics, University of Zurich, Switzerland
{*stoiber* | *glinz*} *@ifi.uzh.ch*

Abstract

The success of very large product lines systems with globally distributed stakeholders often builds significantly on the implicit knowledge of individuals. Final products are typically built by integrating numerous detailed specifications of subsystems. But how exactly all these parts can and need to be integrated to build valid end products is often left unspecified and to numerous discussions, reviews and the expertise of senior architects and product managers.

Building a high-level product line requirements model that explicitly and formally specifies common and variable requirements, their precise integration semantics and the constraints for selecting variable features helps significantly to manage this crucial and often tacit requirements knowledge. Based on an industrial exemplar we motivate and demonstrate such an approach and discuss our early findings regarding knowledge and rationale management in product line requirements engineering.

1. Introduction

The complexity and functionality of software in today's software product line environments is growing faster than ever. Requirements engineering is the key activity in specifying product lines on the one hand abstract enough, so that negotiations with customers can be performed on them, and on the other hand detailed enough, so that they provide a sound basis for implementing solutions.

In current product line requirements practice and research, most approaches build upon many different, separate requirements models and documents. Working with multiple separate diagrams and/or documents for describing the requirements requires frequent, time-consuming switching of views and context. Additionally, for product lines, an orthogonal specification of the variability is needed as well, using e.g. feature trees [8] [1], OVM [11], or decision modeling [13]. Specifying the variability orthogonally introduces the need for traceability between variants in the orthogonal variability model and the corresponding requirements that specify these variants.

For software product lines, which typically are long-living systems, explicit documentation of design rationale is beneficial, especially in maintenance and evolution [2]. Further, past research has shown that capturing and sharing other tacit knowledge about requirements is also beneficial for reuse, traceability, evolution and collaboration in distributed projects [12] [18].

In our ongoing research, we are developing a new product line requirements engineering approach, where we explore a combination of aspect-oriented requirements modeling with table-based boolean decision modeling [14]. The latter is a significant enhancement over our previous work [15]. We are developing an experimental implementation of this approach based on ADORA, an integrated requirements and architecture modeling language and tool that allows modeling a requirements specification with one single and coherent graphical model [5]. Modularizing the variable requirements with aspects makes it possible to integrate the variability specification also into that single model, thus not requiring any additional traceability or mappings. With table-based boolean decision modeling we further specify the details of the variability and variability constraints.

In this paper we introduce an industrial product line exemplar and motivate the need for a more explicit product line commonality and variability specification, which previously was typically implicit and subject to various meetings and requirements reviews when new systems and products were built. With modeling a high-level product line requirements specification we show how our approach could explicitly model such product line requirements knowledge. This knowledge included identifying common and varying requirements, integration semantics of how and where variable requirements can be integrated with other artifacts, necessary constraints for building valid systems, and the documentation of rationale for variability and constraints.

In section 2 we introduce the industrial background and motivate. Section 3 describes the product line exemplar and how we modeled it. In section 4 we discuss in detail how

Proceedings of the Second International Workshop on Managing Requirements Knowledge (MaRK'09), Atlanta, USA, September 2009.

such tacit product line knowledge could be expressed explicitly with our modeling. Section 5 critically discusses potential threats and limitations and section 6 concludes.

2. Industrial Background and Motivation

This work has been motivated by challenges identified at a global Fortune-500 company with a tradition in developing industrial software-intensive systems and development sites spread globally. The systems the company produces are typically built on a three-level composition hierarchy: systems are built of products, and products are built of components. Components are systematically developed to be reusable. Products are assembled with using as many of the existing components as possible and already appear on price lists of the company. And systems, finally, are built from implementing products in order to provide services and applications. The development units typically organize catalogues of available engineering artifacts as spreadsheets at the different levels of the composition hierarchy and maintain portfolios of components, products and/or systems they offer.

As we have identified in a previous study [4], this composition hierarchy in the company does not consist of a company-wide product line variability specification. Thus, constraints between artifacts were negotiated as part of the requirements reviewing activities and hence much of the understanding of commonality, variability and variability interdependencies between products was in fact tacit and distributed in the organization. When a new system was built, the high-level conceptual design, i.e. which feature combination the system could finally implement, or, respectively, which combination of components and products, was subject to numerous reviews, personal knowledge and expertise of many individuals, down- and upwards the composition hierarchy.

We have studied one of the company's product lines in detail and, together with two domain experts, built a high-level product line requirements specification with our approach.

3. An Industrial Product Line Exemplar

The product line we modeled specifies industrial automation devices at the product and components level, with globally distributed stakeholders involved.

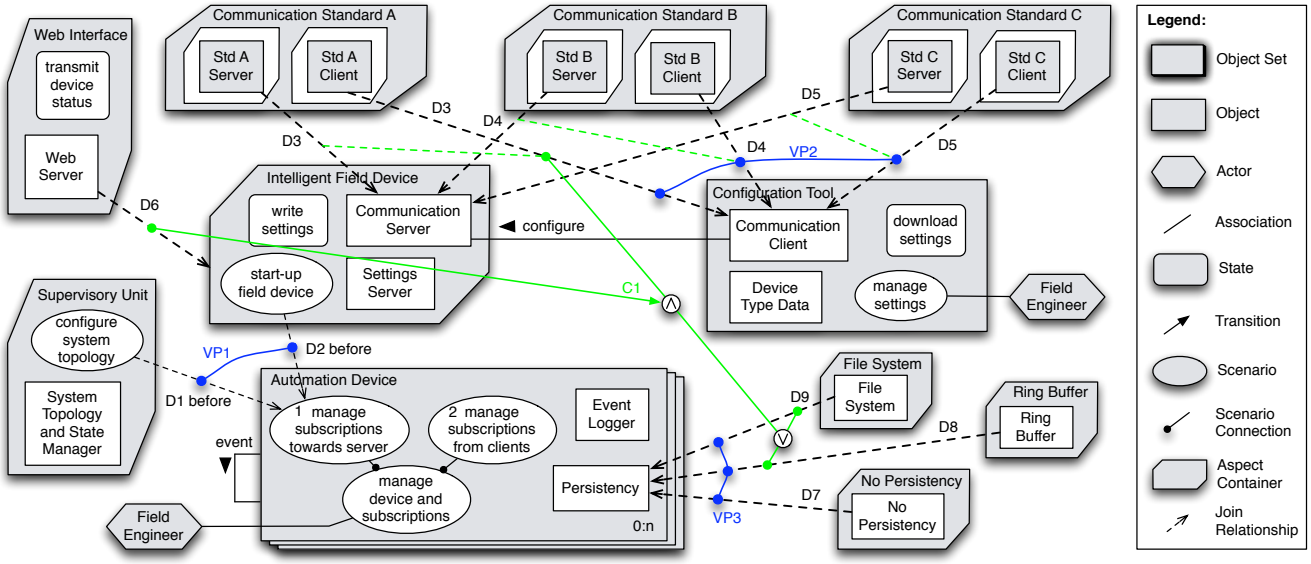
Figure 1 shows the graphical requirements specification and the decision and constraints tables of this product line. The graphical notation is briefly explained in the figure legend. The commonality is modeled with abstract objects or object sets and the variable requirements are modularized with aspect containers and join relationships. Every variable join relationship is annotated with a decision item and the details and constraints of these decision items are modeled in the decision and constraints tables.

Common to all configurations of the product line are a set of automation devices, each consisting of an event logger and a persistence mechanism, and a configuration tool. The automation devices can send events to each other and a field engineer can manage the client- and server-side subscriptions for these events. There are three different options for persistency, one of them being mandatory: no persistency, a ring buffer, or a file system. The automation devices can be of two types: supervisory unit or intelligent field device. For a supervisory unit, the field engineer can configure the system topology and for an intelligent field device, he can start up the device. The intelligent field device is configureable by the configuration tool. For that, one or more of the three available communication standards A, B, and C[1] needs to be supported both at server- and client-side. The configuration tool can download field device settings and also knows all device type data. An intelligent field device can also have an optional web server for remote monitoring. Deploying a web interface requires both the communication standard A installed and either a file system or a ring buffer persistency.

The tables in the lower half of Figure 1 specify the details of the product line variability. The first table, the *decision table*, builds upon the decision items that are defined for all join relationships of variability. Based on these decisions the products can be built by weaving the selected variability into the commonality, using a slightly extended form of the weaving semantics defined in [9]. The second and the third table below formally specify the variability constraints: first the *variation points* which are defined with minimum and maximum number of decisions required to be true, and then the *global constraints*, which are defined with boolean algebra, typically with logical implications. The ADORA tool is also capable of visualizing these constraints graphically in the requirements model [16]. For example, the solid line connecting the join relationship arrows labeled D1 and D2 in Figure 1 visualizes that these two decision items constitute variation point VP1. Further, the ADORA tool is also capable of analyzing the constraints and checking the satisfiability of all currently interesting decision configurations using a boolean satisfiability (SAT) solving tool. The resulting constraint propagations are listed in the columns *ifTrue* and *ifFalse* in the *decision table*.

For every decision item we also document a description or derivation question, which helps analysts derive products from the product line. Further, we document design rationale for all decision items and constraints, thus describing the reasons why the variants and constraints were designed

[1] anonymized upon request by the organization



Decision Table

ID	Description / Derivation Question	Design Rationale	Constraints	ifTrue	ifFalse	Decision
D1	Should the automation device be a supervisory unit?	The automation device platform is used for cost and quality reasons.	VP1	¬D2	D2	*undecided*
D2	Should the automation device be an intelligent field device?	The automation device platform is used for cost and quality reasons.	VP1	¬D1	D1 ¬D6	*undecided*
D3	Should the intelligent field device support communic. std. A?	Standard A is the newest and most performant one.	VP2, C1		¬D6	*undecided*
D4	Should the intelligent field device support communic. std. B?	Is a must in some countries.	VP2			*undecided*
D5	Should the intelligent field device support communic. std. C?	C is still required by many legacy systems.	VP2			*undecided*
D6	Should the intelligent field device a web interface?	The web interface allows access over the world wide web.	C1			*undecided*
D7	Is there no memory in the automation device?	Might be interesting for non-critical systems.	VP3	¬D8 ¬D9 ¬D6		*undecided*
D8	Is there a ring buffer in the automation device?	Ring buffers are cheap to implement and fast.	VP3, C1	¬D7 ¬D9		*undecided*
D9	Is there a file system in the automation device?	File system offer huge amounts of storage space.	VP3, C1	¬D7 ¬D8		*undecided*

Variation Points Table

ID	Description / Rationale	minCard	maxCard	Decisions Involved
VP1	An automation device always can be either a supervisory unit or an intelligend field device.	1	1	D1, D2
VP2	There are three different communication standards to configure a field device: standard A, B, and C. At least one must be chosen.	1	3	D3, D4, D5
VP3	There exist three different persistency types: no persistency, a ring buffer, or a file system. One of these must be selected.	1	1	D7, D8, D9

Constraints Table

ID	Description / Rationale	Antecedent	Operator	Consequent
C1	A web interface always requires the communication standard A and a local persistency installed (either ring buffer or FS).	D6	=>	D3 and (D8 or D9)

Figure 1. The automation device product line specified in the ADORA language.

like this. This helps better understand the model and is beneficial when evolving a product line, particularly in dynamic and multi-stakeholder environments.

4. How Tacit Product Line Variability Knowledge Became Explicit

Modeling product line requirements with our approach allows the explicit specification of information which cannot be expressed explicitly in other modeling approaches and thus would remain tacit knowledge. Subsequently, we will pinpoint these novel concepts (in bold) and discuss how we modeled this previously tacit requirements knowledge.

Modeling common and variable requirements in a single diagram, with aspects. This is a capability that is unique to compositional (additive) approaches to product line engineering. It allows modeling the commonality with a simple conventional model, which only shows the really common requirements (i.e. in contrast to subtractive product line approaches where models of commonality and variability are mingled). The same is true for variable features: every variant is modeled as an aspect, showing only the requirements it may add to the system specification. With aspects, commonality and variants can be shown in the same model. The aspectual join relationships also show explicitly and precisely where and how a variability can impact other variants or the commonality.

This is particularly useful for managing distributed development that involves different teams. Typically, for every single product feature a dedicated development team is responsible. Every team needs to know how exactly and where their artifacts will be integrated with other system artifacts. As this information is modeled explicitly in our approach, development teams working on single components or features can easily comprehend which technical interdependencies might occur with other system artifacts and with which other development units they may need to coordinate

their development.

Our aspect modeling approach is equally useful for maintaining and evolving the product line. On a fine-grained development level (components, or below) artifacts are often reused in various parts of the system. In order to maintain and evolve such artifacts, one needs to know all places where they are used. Without modeling such relationships explicitly, development teams for different lower-level features often don't really know where and how their components actually are and will be applied. Neither do they know whether their artifacts were integrated seamlessly, or whether there wasn't an ideal fit and workarounds had to be built. Explicitly modeling features and how they integrate is thus crucial for evolving high-quality components and products.

Explicit and visual decisions and constraints, in the graphic requirements model. Product managers and solution architects working on more coarse-grained levels further need to know which combinations of integrating variable requirements are legal to form a working product or system. Therefore constraints that are required for technical feasibility of the products must be formulated and also other, non-technical constraints, such as, e.g., long-term portfolio or marketing strategies, should be formulated explicitly. With our table-based decision and constraints modeling we provide the necessary framework to model constraints formally and explicitly. In our industrial case, a lot of knowledge about feature interdependencies and other constraints that previously was distributed in the organization could now be made explicit this way.

Rationale descriptions for variability and constraints. An explicit documentation of rationale is beneficial, also for variability in software product lines [17]. Figure 1 shows how our approach supports the documentation of rationale for every variability decision item and constraint. Thus, engineers can easily document why the variability and its constraints were designed in a specific way in a product line. Such rationale is especially helpful when it comes to maintenance, optimization and evolution of the product line. The separate column for rationale information encourages requirements engineers to document also their design rationale knowledge explicitly.

5. Critical Discussion

For our argumentation in this paper we have essentially assumed that all stakeholders involved in the product line are working on one single and shared central product line requirements model. This assumption implies that all stakeholders have access to this shared model and are working on the most recent version, so that no versioning conflicts will occur. In our research environment we are currently using a subversion repository to store and share our models. Ideally, for such a multi-stakeholder and distributed product line, additional version control mechanisms and access rights management should be implemented. This way the stakeholders can work on their parts of the central product line model concurrently. An authorization mechanism would also be beneficial, to guarantee that only authorized stakeholders are able to maintain and change the concerned requirements. In our current research, we do not focus on version control for product line models. However, a real industrial application of our approach would require an implementation of versioning and access right mechanisms.

Every stakeholder in the system should have the possibility to suggest issues for improvement, which could then be reviewed and criticized by the responsible architects and development teams and eventually be approved and integrated. Doing so would essentially require an integration of a groupware support system, like for example EasyWinWin [7]. Integrating such groupware with our product line approach would address a so-called many-to-many requirements negotiation constellation [3]. New requirements to the product line, for example demanded by markets or various customers, would need a negotiation with all the involved development teams on the one hand. Changes in more detailed existing technical requirements on the other hand would need to be negotiated with potentially all market- and customer-responsible stakeholders as well. Considering further that both of these situations might happen severalfold and probably concurrently with different parts of the product line, a sophisticated many-to-many negotiation constellation support would be required. Implementing such groupware support for evolving product line specifications might possibly be a future research area within our project.

Another limitation is that ADORA is a language for modeling requirements and high-level architecture. Consequently, it only supports model simulation as a means of validating and evolving models [6], but currently neither supports detailed design nor code generation. In terms of model-driven engineering capabilities, UML [10] is superior to ADORA, hence. The reason why we don't use UML is UML's fundamental inherent weakness of spreading a model over dozens of loosely coupled diagrams which dramatically complicates integrated product line modeling and also makes model understanding difficult, because a model reader needs to switch between numerous diagrams and has to integrate the contents of different diagrams mentally. So in all cases where an integrated model is essential such as in our product line modeling approach, as well as for understanding complex models, where a tool can generate views at any level of abstraction from an integrated model, we consider ADORA to be the better choice than UML.

6. Conclusion and Future Work

In this paper we have presented a new approach to product line requirements engineering and its application to an industrial exemplar from a multi-stakeholder, distributed environment. Our approach builds on (i) modularizing variable feature requirements with aspects, (ii) using explicit join relationships for their integration semantics, (iii) modeling the commonality and the variability of the product line in a single aspectual model, (iv) describing details of the variability including variability constraints in tabular form, and (v) visualizing variability constraints graphically.

We demonstrated that, with our approach, we could document various product line requirements knowledge explicitly that previously was distributed in the organization and was handled implicitly, relying on the expertise and experience of the involved stakeholders.

In our future work we plan to investigate many-to-many negotiation constellation support with our approach. We will continue developing our concepts and tool implementation. The focus will be on semi-automated requirements composition with dynamic aspect weaving and on the problem of providing more user-friendly layouts when weaving aspects graphically in such a notation. We further plan to validate our approach empirically in industry for gaining more insight and evidence about the usefulness and potential of this new kind of product line requirements modeling.

References

[1] K. Czarnecki, S. Helsen, and U. Eisenecker. Formalizing Cardinality-based Feature Models and their Specialization. *Software Process Improvement and Practice, 10(1)* , 2005. pp. 7-29.

[2] A.H. Dutoit, R. McCall, I. Mistrik, B. Paech (Eds.). *Rationale Management in Software Engineering.* Springer, 2006.

[3] S. Fricker, P. Grünbacher. Negotiation Constellations - Method Selection Framework for Requirements Negotiation. *Proceedings of RefsQ'08.* LNCS 5025, Berlin, Springer. 37-51.

[4] S. Fricker, R. Stoiber: Relating Product Line Context to Requirements Engineering Processes Using Design Rationale. *Proceedings of PiK'08 Workshop.* Munich, Germany, 2008.

[5] M. Glinz, S. Berner, S. Joos: Object-oriented modeling with ADORA. *Information Systems **27***, 6. 425-444. Elsevier, 2002.

[6] M. Glinz, C. Seybold, S. Meier. Simulation-Driven Creation, Validation and Evolution of Behavioral Requirements Models. *Dagstuhl-Workshop MBEES 2007.* Informatik-Bericht 2007-01, TU Braunschweig, Germany. 103-112. 2007.

[7] P. Grünbacher, B. Briggs. Surfacing Tacit Knowledge in Requirements Negotiation: Experiences using EasyWinWin. *Proceedings of HICSS'01.* 2001.

[8] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson: Feature Oriented Domain Analysis Feasibility Study. SEI Technical Report CMU/SEI-90-TR21 1990.

[9] S. Meier, T. Reinhard, R. Stoiber, M. Glinz. Modeling and Evolving Crosscutting Concerns in ADORA. *Proceedings 11th Early Aspects Workshop @ICSE'07.* Minneapolis, USA. 2007.

[10] Object Management Group. UML. http://www.uml.org/

[11] K. Pohl, G. Böckle, van der Linden, F.: *Software Product Line Engineering – Foundations, Principles, and Techniques.* Springer, Heidelberg, 2005.

[12] I. Rus, M. Lindvall. Knowledge Management in Software Engineering. Guest Editor's Introduction. *IEEE Software.* May/June 2002.

[13] K. Schmid, I. John. A customizable approach to full lifecycle variability management. *Science of Computer Programming*, Vol. 53, No. 3. Elsevier. 2004.

[14] R. Stoiber, M. Glinz. Software Product Line Requirements Engineering Based on Aspects (SPREBA). Research Project at the University of Zurich. URL: http://www.researchportal.ch/unizh/p12003.htm

[15] R. Stoiber, S. Meier, M. Glinz. Visualizing Product Line Domain Variability by Aspect-Oriented Modeling. *Proceedings of REV'07 Workshop.* IEEE CS. 2007.

[16] R. Stoiber, T. Reinhard, M. Glinz. Visualization Support for Software Product Line Modeling. *Proceedings of ViSPLE'08 Workshop.* 2008. pp 313-322.

[17] A. Thurimella, B. Bruegge, O. Creighton. Identifying and exploiting similarities between rationale management and variability management. *Proceedings of SPLC'08.* 2008.

[18] A. Thurimella, W. Maalej, H-J. Happel. *MaRK'09 Workshop.* http://www1.cs.tum.edu/static/mark09/