JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2005**

Vol. 4, No. 9, November–December 2005

Ideas from SPLC

John D. McGregor, Clemson University and Luminary Software LLC, U.S.A.

Abstract

Software product lines continue to provide adopters with gains in many areas such as faster time to market and increased productivity. The recent Software Product Line Conference (SPLC) provided a forum in which a spectrum of results were reported. In this issue of Strategic Software Engineering I will use the activities at SPLC to focus on a couple of interesting areas that have strategic importance.

1 INTRODUCTION

Software product lines is a strategy for improving a company's product production capability. They do this by differentiating between those features that are common across a set of products and those that appear in only some products. This analysis is used to define a set of core assets that are structured to support building the set of products.

The Software Product Line Conference (SPLC) is the gathering place for the software product line community. Like most international technical conferences, SPLC brings together industry and academic professionals to discuss research and practice. Many organizations recognize the strategic importance of participating in conferences, sharing their results, and listening to the latest information in their field. Participants can see trends in the industry reflected in the topics presented in the formal sessions and the discussions in the coffee breaks.

For this issue I have focused on three themes that received attention at SPLC 2005. I will give a taste of each and invite you to explore further through the proceedings of the conference available from Springer and the others works cited here. Whether you are a member of the product line community or not, these are issues that most organizations need to consider.

2 PRODUCT SPECIFICATION AND PRODUCTION

The coordinated development of a set of products provides the opportunity to explore more powerful means of product production. One production problem is the difficulty in crafting a complete, correct, and consistent product specification. A couple of answers

Cite this article as follows: John McGregor: "Ideas from SPLC", in *Journal of Object Technology*, vol. 4, no. 9, November-December 2005, pp. 23-29, http://www.jot.fm/issues/issue_2005_11/column3.

were presented at SPLC that center around understanding the domains relevant to the product line.

Domain Modeling

Product line organizations must understand the domains relevant to their products. There are essentially two different perspectives for conducting a domain analysis and I find both perspectives useful for different reasons. The first approach is to do what I will call a conceptual domain analysis. In this approach the content of the model are concepts from the domain, such as the concept of *score* in a gaming product line. The model includes a class diagram that identifies concepts and the relationships between them, sequence diagrams that capture standard algorithms used in the domain, and state diagrams that capture the life cycles of instances of the concepts. This model is useful as the initial object model for an object-oriented design.

This conceptual domain analysis is useful for establishing the vocabulary to be used in the statement of the product requirements. The conceptual analysis provides a basis for examining the requirements for correctness and consistency. The feature-oriented domain analysis identifies constraints and other relationships among the individual features. This second perspective, using product features instead of concepts, is discussed in more detail in the following sections.

Feature Modeling

Feature modeling is emerging as a fundamental technique for product line modeling and this is reflected by several papers on the topic at SPLC 2005. A feature is a “prominent aspect of something” according to web definitions. It is a useful unit of specification when the specification is created by a client or a domain expert. A feature is at the correct grain size to specify the functional variations among a set of products.

The basic notation for a feature model is a simple graphical representation, see Figure 1 for examples of one such notation. The darkened circle above the “event loop” feature indicates a required feature. The light circle indicates the “scoreboard” feature is optional. There are several different notations for feature graphs, although most are closely related. The feature model for a product line must be able to model a range of variability concepts including inclusive and exclusive or and selection of potentially multiple members from a set of choices.

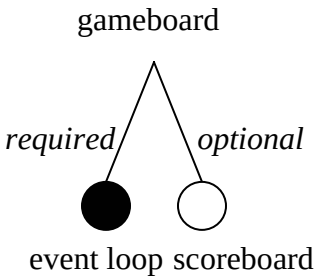


Figure 1 - Feature model notation

Formalizing Feature Models

The graphical notation used in feature models makes for an easy to understand, but impossible to automate, specification. In a session at SPLC Don Batory showed an equivalence from this graphical notation to an attribute grammar [Batory 05]. For the very small model in Figure 1, the grammar would be stated as:

gameboard ::= eventLoop [scoreboard];

This grammar makes it easier to automate product derivation, but only partially addresses the need to evaluate the consistency and completeness of the model. Batory then showed that the attribute grammar can be translated into the propositional logic formula:

$$gameboard \Leftrightarrow eventLoop \wedge scoreboard \Leftrightarrow gameboard \wedge gameboard = true$$

The advantage of getting to this form is the ability to automate the consistency check for the model. As the model evolves, each new version can be checked for consistency relatively quickly.

Using the feature model

A product line feature model shows the complete range of features that are within the scope of the product line. Each product team then specifies their product by making all the decisions necessary to make all the choices called for at alternative and optional branches in the feature model. Essentially the product-specific model contains only those features that are required for the final product.

Depending upon the production method for the product line the specification technique may be the only portion of the production method that requires human intervention. It is fairly easy to design a decision tool that guides the product specifier through the choices needed to specify the product. The product is then automatically generated from the specification.

3 ECONOMIC MODELING

Many of the strategic decisions made during the initial planning for product line adoption and during product line operation are based on economic issues. At this year's SPLC there was a "competition" among several modeling techniques [Clements 05]. A scenario was provided and each team produced a model of the scenario using their specific technique. Several issues appeared.

Basic assumptions and goals

Even simple models involve a number of factors, not all of which can be varied in one analysis if the model is to be understandable. In the case of the product line economic modeling competition, there was a range of assumptions made in the original problem statement and by the individual models. Some of the important cost factors in a software product line are:

- The rate at which the domain is changing - This rate is not under the control of the product line organization but it affects how often the core assets must be updated. Even the product line approach does not result in large amounts of reuse if the core assets must be modified after each product is built. This is often expressed as a percentage of the core asset base that must be modified each year (or other appropriate period).
- The productivity of the asset builders producing the core assets – This value can be affected by the product line organization through hiring appropriate people, training, and adoption of appropriate tools. The models account for this factor using a LOC/hour parameter.
- The commonality among the products – This value can be controlled by adjusting the scope of the product line. The commonality allows the economic model to estimate the amount of total content needed to produce all of the products. The higher the commonality, the smaller the cost of the core asset base. One of the techniques modeled this commonality as an integral part of the cost computation.
- The timing for when the assets and products will be produced – The sooner the assets are produced, the sooner the cost is incurred. The sooner the products are produced, the sooner a cash flow is established. The sooner the products are produced, the less risk that a product will be cancelled. Several models take time into account when computing return on investment as well as actual costs.
- The certainty of the values being used in the model – A product line usually represents an investment over time. Plans can change based on market trends, economic conditions, and technology innovations. One model in the competition used probabilities that each of its chosen scenarios would actually occur to account for uncertainty. Another model used a range of possible conditions and recomputed each scenario for several selected values within the range.

Scope and depth of the models

One model provided a high-level view of the costs and benefits of the entire product line. Other models focused more on just the software to be developed and delivered. Product line organizations plan and model before committing to code. In my opinion, the costs of these assets have to be accounted for to have an accurate model. One model accounted for that by adjusting the LOC/hour productivity while other models explicitly modeled costs such as the cost of the architecture.

Fixed versus custom modeling approaches

A major difference in the models is whether they use a fixed set of equations to compute the economic value or whether they describe a modeling approach and create each model specifically for the scenario at hand. Product line organizations vary dramatically from one to another. The fixed equation models must be at a very high-level to be broadly applicable. This limits the specificity of the answers they can provide. The modeling approach allows individual variations in the product line organizations and their assumptions to be taken into account but obviously require more effort and training to use effectively.

The slides for this competition, including the models, will be available from www.splc.net shortly. This competition gave a clear survey of the state of the art in developing the economic portion of the strategic arguments for the product line business case.

4 AGILITY

Since much of product line practice is based on comprehensive planning, how do you rapidly respond to innovative ideas? Böckle raised this interesting question in a session at SPLC [Böckle 05]. Innovation that is anticipated, and therefore may correspond to a variation point, is probably not very innovative. So is it possible to anticipate that the variation points will not anticipate the full range of variation – a meta-level of variation?

The ability to respond quickly is strategically important. Software product lines are typically structured to minimize the time to market for a product that is within the scope of the product line. The question is, does that have to equate to a slower than normal time to market for products or features that were not included in the original planning for the product line? The worst case should be the time it takes to build the product as a one-off product but this may not be the case if the “reusefulness” of an asset makes it more difficult to modify than a simpler, single-use asset.

If a decision is made to expand the scope of the product line, the time for producing the newly in-scope product may be longer than in non-product line products depending upon the structure of the core assets. The more abstract the level of the assets, the more flexibility at the concrete level and the more effort required to modify at the abstract

level. Consider the Unified Modeling Language (UML). A number of experts have found the UML lacking constructs that were necessary for their purpose. Rather than invent their own language, they have extended UML. This is facilitated by the meta-model provided by the Meta-Object Facility (MOF). Structuring the core assets of a product line to have this level of flexibility can support an important level of agility.

Later in the conference, Jan Bosch, a Vice President at Nokia, alluded to a similar situation but a different type of agility: the ability to adapt products to unanticipated modifications beyond deployment [Bosch 05]. In his view, variability binding is extending well beyond deployment time. He compared systems which previously shipped as a closed product – 100% bound - with systems shipped more recently in which an intermediary might add 30 to 40% additional content through late binding after the original development group has released the product. Bosch further contrasted those situations to the currently shipping open systems in which the intermediary may supply 60% or more of the functionality using late binding mechanisms.

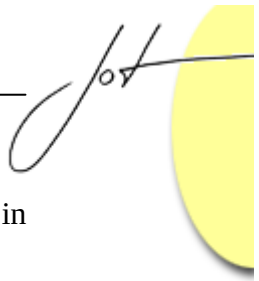
Both of these approaches to agility relate to how quickly new functionality can be provided. The second type of agility relates to a third-party providing product functionality as opposed to the first type of agility which relates to the original development organization responding to innovative ideas during product line operation. The ability for large amounts of additional features to be added to a product after it leaves the original producing company provides flexibility with respect to market. The open system to which large additions may be made by others allows agility with respect to the market but in many cases adds complexity.

The strategic advantage of a software product line may become a liability if these forms of agility aren't recognized, accommodated, and appropriately managed.

5 SUMMARY

Software product lines continues to be a means of achieving the strategic objectives of an organization whose mission is to produce software-intensive products. Researchers continue to refine product line practice. New techniques for creating and using feature models are being created. Techniques for justifying and planning product line efforts are maturing as we devise more complete models. We are beginning to consider how to optimize product line practice by asking questions about the agility of otherwise successful product line organizations. All of these advances better position the product line organization to achieve its strategic goals.

SPLC is the forum in which software product lines researchers and practitioners explore ideas and report results. The growing community is a vibrant group that is making substantial progress in a wide range of areas. I have only reviewed a few of the ideas presented this year. In particular I have not discussed most of the industrial reports and panels nor have I described the various tutorials and research workshops. SPLC provides a venue for many types of interaction with, and presentation by, industrial and academic participants.



I invite you to sample these areas when our community meets again in Baltimore in August 2006. See www.splc.net for details.

REFERENCES

[Batory 05] Don Batory, Feature Models, Grammars, and Propositional Formulas, Software Product Line Conference, 9th International Conference, LNCS 3714, 2005.

[Böckle 05] Günter Böckle, Innovation Management for Product Line Engineering Organizations, Software Product Line Conference, 9th International Conference, LNCS 3714, 2005.

[Bosch 05] Jan Bosch, Software Product Families in Nokia, Software Product Line Conference, 9th International Conference, LNCS 3714, 2005.

[Clements 05] Paul Clements, A Competition of Software Product Line Economic Models, Software Product Line Conference, 9th International Conference, LNCS 3714, 2005.

About the author

Dr. John D. McGregor is an associate professor of computer science at Clemson University and a partner in Luminary Software, a software engineering consulting firm. His research interests include software product lines and component-base software engineering. His latest book is *A Practical Guide to Testing Object-Oriented Software* (Addison-Wesley 2001). Contact him at johnmc@lumsoft.com.