Improving Decision Making in Software Product Lines Product Plan Management ⋆

Pablo Trinidad, David Benavides, and Antonio Ruiz-Cortés

Dpto. de Lenguajes y Sistemas Informáticos
University of Seville
Av. de la Reina Mercedes S/N, 41012 Seville, Spain
{trinidad,benavides, aruiz}@tdg.lsi.us.es

Abstract The increasing demand on developing Software Product Lines (SPL) has given a lot of interest to software engineering researchers in improving and also replacing, current methods and techniques applied to clasical sofware systems development.
In this paper, we introduce our first ideas within our proposal on improving the decision making while SPL Product Plan Managing. The key points in our proposal are originality and viability. We do not know any other proposal dealing with the same problem so far, and our first impressions guide us to predict a high viability.

Keywords: Production Plan, Feature Model, Software Product Lines, Cost Estimation, Time Estimation.

1 Introduction and background

Software reuse is one of the main goals software engineers treat to achieve. Many approaches have been proposed during last decades following this purpose: libraries, objects and classes, components and nowadays Software Product Lines (SPL). Previous approaches allowed to reuse in a higher or lower percentage depending on the effort of software engineers, becoming software reuse an ad-hoc methodology. SPL systematizes software reusing by building products from existing assets, rather than developing products one by one from scratch. SPL consist of a set of common and product specific features so that products are built from one or more common features and their specific features. Thus, software reuse comes from common features, also named core-assets, shared by concrete products. In [4], it may be observed that the appliance of an SPL approach in real projects, has improved the reusability and therefore, final product liability and time-to-market among many other attributes.

In SPL development, three main activities are distinguished [1]: core-asset development or domain engineering, product development or application engineering and management. The products are built from core-assets, but also core-assets may be built from existing products. The synchronization between these two activities is arranged by management.

⋆ This work was partially funded by the Spanish Ministry of Science and Technology under grant TIC2003-02737-C02-01 (WEBMADE)

Figure 1. Essential Activities in Software Product Lines Practice

One of the main activities in SPL development is production planning, which describes how concrete products are built from core-assets in documents named Production Plans (PP). There should be one product line PP and one PP for each concrete product, which would be a particular case of product line PP. Core-asset developers would be the responsible of developing the product line PP, and would communicate the production strategy to product developers. Product developers would be in charge of going into detail and infer one concrete product PP from the product line PP.

A PP may be made up of one or more of these issues:

- Core-asset development organization.
- Core-asset testing and maintenance.
- Dependencies among core-assets and concrete products.
- Change controlling and core-assets configuration.
- Cost and development time core-assets estimations.
- Core-asset development resources.

For each concrete product, product developers consider the same issues, taking as reference the SPL Production Plan.

Considering previous issues, product plan should be built from products information. Software engineers can build many possible PP, but decisions should be made to build the best one. Examples of decisions may be: which assets should core-asset developers build? And what about product developers? Which are the products supplying a quality level? Which is the product supplying some features that costs less than an amount? How long would it take to develop it? How many product developers will be needed in case of building some specific products?

Software engineers should obtain information to make decisions. Where do they take this information from? It may be extracted from SPL definition following traditional methods, where each concrete product must be analysed in particular. This method is not scalable because considering a huge set of assets forming an SPL, the

number of potential products increases exponentially. A method to model SPL compactly would ease its representation. If extracting information from this model is feasible, it may be used to make decisions about PP building and managing.

We consider that feature modelling has these desirable qualities. Feature models (FM)[9,2,7,8,6], are used to compactly define SPL in terms of features and relations amongst them. A feature is a distinctive characteristic of a product, and it may refer to a requirement, component or even a piece of code. FM allow to represent the SPL variability points, using notations such as the one proposed by Czarnecki [2]. But, what kind of information can we obtain from FM? In [3], a method to automatically extract functional and extra-functional information from FM is proposed. Questions may be asked to the model to obtain information. But, what kind of questions may be asked to the model to obtain information to assist decision making? Does FM reasoning answer all the questions software engineers ask?

In this paper, we deal with to automatically obtaining useful information from FM. This information may assist decision making in SPL Production Plan. As far as we know, the originality of this work is its key point, since the only proposal [5] we have found about SPL Product Planning has no automatic support. In the other hand, our first implementations of our ideas have given us a good feedback that helps us to predict a high viability.

The reminder of this paper is structured as follows: First, we present Feature Models and how to extract information from them, in Section 2; In Section 3 answers to some questions that may assist decision making are given.

2 Feature Modeling in Software Product Lines

Feature Modeling is a technique which allows to model and organize the differences and commonalities among products in a SPL. The result is a compact representation of all the potential products within a SPL. To achieve this, product features and relations among them are determined. A feature is a distinctive characteristic of a product. Depending on the development stage, it may refer to a requirement (if products are requirement documents), a component in an architecture (if products are component architectures) or even to pieces of code (if products are binary code). There are many notations to specify feature models. We found Czarnecki's notation [2] as the most comprehensive and flexible one, and also one of the most cited.

Czarnecki's notation distinguishes four kinds of relations: mandatory, optional, alternative and or-relation. These are relations between a parent feature and one child feature (in mandatory and optional cases) or among a parent feature and one or more child features (in alternative and or-relation cases).

Figure 2 depicts a feature model representing a Home Integration System (HIS) SPL using Czarnecki's notation. This example is partially inspired by [8]. It represents a product line which controls and manages a collection of devices to maintain security and safety of a building or a house. Optional features such as appliances control and flood control may be included in more advanced products.

Then, The first objective of compactly representing the feature model is achieved. But is it possible to ask questions to the model and extract information? In [3], a method

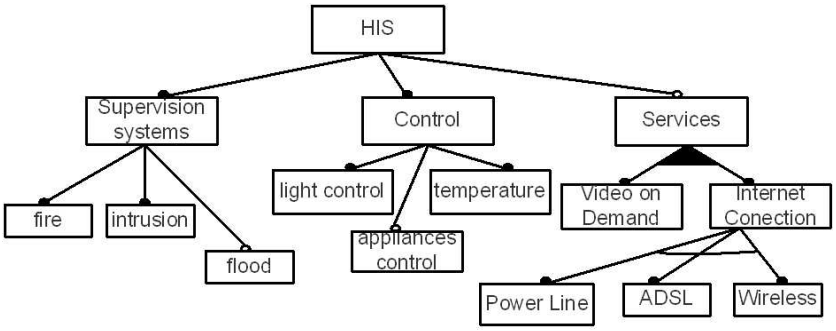

Figure 2. Home-Integrated System Feature Model

to reason on a SPL based on Constraint Programming is proposed. The next step would consist of trying to extract useful information from FM. But feature models only allow to model features that deal with characteristics related to the functionality offered by a SPL (functional features). However, the remainder characteristics, also called extra-functional features cannot be modeled in Czarnecki's notation.

[3] deals with this problem and introduces a method to model functional and extra-functional features extending Czarnecki's notation, as seen in Figure 3. Moreover, an holistic reasoning on SPL is also possible using both features at the same time.

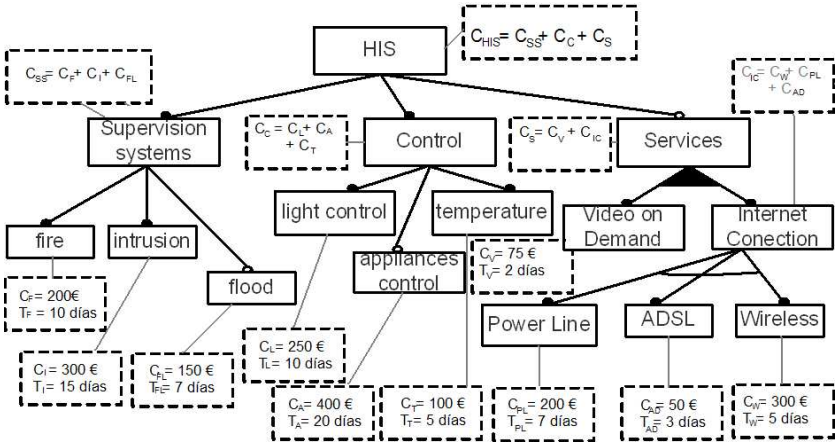

Figure 3. Extended Feature Model considering extra-functional features

Next section, we are dealing with answering some typical questions that may assist decision making on PP managing and building.

3 Our proposal

Taking in account the reasoning and information extracting capability on FM, the next step would be obtaining useful information to assist decision making on PP managing. In particular, we deal with the decisions presented in Section 1.

3.1 Decision Assistance with core-asset developers

The first issue we are considering in PP building is which assets would core-asset developers build. In order to assist decision making on this issue, we should ask the model which are the core features in feature model. The question to be asked to the model is: Which is the set of mandatory features in the model? Applying this question to the example in Figure 2 the result would be:

$$CoreFeatures = \{Fire, Intrusion, LightControl, Temperature\}$$

Thus, core-asset developers should build this set of core-assets.

3.2 Decision Assistance with product developers

Next question to be considered is which assets should product developers build. Let us remind that product developers are in charge of building products integrating core-assets. Then, product developers need to know which are the core-assets to be integrated and which new features they should build.

Feature model represents all the possible products that can be build within the SPL definition. The assets product developers build, would depend on the products to be built. Therefore, we should know which are the features each product is formed by. Two questions should be asked in order to know the product developers work in a concrete product development: Which are the core-assets to be used in products development? Which are the features that composes products supplying some functional and/or extra-functional features?

First question is already answered in the previous subsection. Second question can be obtained from feature model applying a filter [3]. Filter acts as a limitation for potential products of the model. Then product developers should integrate core features and build those features in the product that are not part of the set of core-assets.

3.3 Decision Assistance with extra-functional features

Many times, the extra-functional features in a product are a key issue. Products may differ not only in its functional features nor in its extra-functional features. For example, two different products may exist in the FM depicted in Figure 3 with the same functional features. This would happen if bandwidth of ADSL can be 128 *Kbps* and 256 *Kbps*. Then, reasoning on extra-functional features is important in decision making assistance. As introduced before, reasoning on FM is feasible using extra-functional features. Thus, this kind of questions may be asked to the FM.

3.4 Decision Assistance with Cost

In order to know products cost, we should be able to estimate it *a priori* for each feature. These may be estimated using any of the existing methodologies to estimate cost and time. Let us consider we are able to obtain such estimations precisely. Cost and development time can be treated as extra-functional features, therefore they may be modelled within the FM, as you can see in Figure 3. This way, may questions be asked about cost?

To be able to ask questions about cost, the concept of a product development cost should first be defined. For the example depicted in Figure 3, cost would be considered the sum of each feature cost plus a constant value K_c. Any other expression may be accepted, since it were supported by this model.

Consider we would like to know which are the products supplying some features that cost less than an amount. Two filters should exist in order to obtain such an information. One filter that limits the functional features in resultant products; and another filter that considers a maximum cost. For example, taking in account that the necessary functional features in a product, are *Fire, Intrusion, Light Control, Temperature, Wireless y Appliances control* considering a maximum cost of 1650€, the potential products would be:

- Product 1: {Fire, Intrusion, Wireless, Light Control, Temperature, Appliances,1550€}
- Product 2: {Fire, Intrusion, Video, Wireless, Light Control, Temperature, Appliances, 1625€}

In conclusion, reasoning on cost is feasible in feature models, whenever development cost concept is defined.

3.5 Decision Assistance with Development Time and resources

Task assignment to resources and scheduling are two key activities on PP managing. Decisions like the how many product developers will be needed in case of building some specific product, or how long a product development would take considering a concrete number of resources are very difficult to made. Assistance in these issues are more than desirable. In order to ask questions to the FM about time and resources, these concepts should be defined at first.

In case of being only one resource at our disposal for the whole SPL development, a concrete product development time would consist of the sum of each product feature development time. However, this is an infrequent case. Usually, several resources are available. These may work at the same time and independently. Moreover, dependencies among features can exist, since a feature development could depend on another to be available. Therefore, to be able to reason on development time, we should express at first all this information relating available resources, each feature development time and their dependencies in FM.

In [3], reasoning on FM is based on Constraint Programming, representing FM as a Constraint Satisfaction Problem (CSP). W.Nuijten's thesis [10], exposes a representation of development time and resources assignment information, as a *Time and Resource*

Constrained Scheduling Problem(TRCSP), a particular case of CSP. Thus, it is possible to represent this information as a CSP, and consequently, it is possible to automatically reason on FM.

Once the information may be modelled, questions about development time and resources may be asked to FM.

3.6 Final considerations

Many other questions may be considered apart from the ones considered in this section. As seen before, the only limit we have found in Benavides' model is the ability of representing within the FM, the information to be extracted.

4 Conclusion and Further Work

In this paper we introduce our first applications of feature modelling to product planning. The ability of reasoning on feature models, based on Constraint Programming is very promising. Constraint Programming is widely supported by commercial products. Our first implementations, (check them on *http://www.tdg-seville.info/topics/spl*) solve some problems exposed in section 3 in fair times. The extension of our work is twofold: First, keep on finding new decisions to make in production planning and improving the ones considered in this paper; Secondly, check our ideas in an industrial environment, devoloping a tool which supports to assemble the previous ideas and the visual definition of SPL via feature modelling.

References

1. P. C. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison-Wesley, August 2001.
2. K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Techniques, and Applications*. Addison–Wesley, May 2000.
3. D. Benavides, A. Ruiz–Cortes, and Pablo Trinidad. Coping with automatic reasoning on software product lines. In *Second Groningen Workshop on Software Variability Management*, December 2004.
4. G. Chastek, P.Donohoe and J. D. McGregor. A study of product production in software product lines. IEEE/ANSI Standard CMU/SEI-2004-TN-012, Carnegie Mellon Software Engineering Institute, March 2004.
5. J. Lee, K.C. Kang, S. Kim. A feature-based approach to product line production planning. In *To appear in the Proc. of the Third Software Product Line Conference (SPLC2004)*, September 2004.
6. J. van Gurp, J. Bosch, and M. Svahnberg. On the notion of variability in software product lines. In *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01)*, pages 45–54. IEEE Computer Society, 2001.
7. K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature–oriented domain analaysis (foda) feasibility study. IEEE/ANSI Standard CMU/SEI-90-TR-21, Carnegie Mellon Software Engineering Institute, November 1990.
8. K.C. Kang, J. Lee, and P. Donohoe. Feature–oriented product line engineering. *IEEE Software*, 19(4):58–65, July/August 2002.

9. M. Griss, J. Favaro, and M. d'Alessandro. Integrating feature modeling with the rseb. In *Journal of Computing and Information Technology*, volume 10, pages 76–85, 2002.
10. W. Nuijten. *Time and Resource Constrained Scheduling : a Constraint Satisfaction Approach*. Phd thesis, Technische Universiteit Eindhoven, 1994.