Product Line Requirements based on Goals, Features and Use cases

Bruno González-Baixauli, Miguel A. Laguna, Yania Crespo

Department of Computer Science, University of Valladolid,
Campus M. Delibes, 47011 Valladolid, Spain
{bbaixauli, mlaguna, yania}@infor.uva.es

Abstract. Traditional PL requirements approaches present several problems in requirements analysis, mainly in variants analysis and selection. The main handicap is dealing with non-functional requirements. These problems can be solved with the introduction of the goal/softgoal paradigm. This paradigm introduces intentionality ("whys") and allows relating functional and non-functional requirements, the basis of the variant analysis. This proposal improves the PL requirements introducing the goal/softgoal paradigm and relating it with well-known techniques as feature and use case modeling.

1 Introduction

Software reuse has been a very promising discipline for many years, but the results have not been as good as expected. Recently, Product lines (PL) appear as the more successful approach in the reuse field, as they combine coarse-grained components, i.e. software architectures and software components, with a top-down systematic approach, where the software components are thought a priori and integrated in a high-level structure [6].

Concerning requirements, the more used techniques are related with features. These techniques are focused in commonality and variability to find the components common to the entire PL. However, we think these techniques are too focused on PL architecture definition (too much design-oriented), therefore complex to apply by non-domain experts or without a more requirements-oriented technique.

Recently, requirements engineering (RE) has taken growing importance inside software engineering. One of its most important approach is the goal-oriented RE. It proposes to model explicitly the intentionality of the system (the "whys"). The intentionality has been widely recognized as an important point to the system, but usually it is not modeled. The main advantages of the goal-oriented approach are that can be used to study alternatives in software requirements (it uses AND/OR models that models the different alternatives) and that can relate functional and non-functional requirements (NFR) easily.

Then, there are two important characteristics of goals that can be useful to the PL approach: first, they express the intentionality of the system, for that reason they give a very natural way to take decisions. We can take decisions from "what we want"

versus “what the system does”. Second, they can model alternatives in system requirements, what can be easily mapped to variants in PL.

However, this concept must be linked to PL, mainly with PL architectures. An easy way to do this is to relate goals with traditional features. Here, lot of work have been done relating goals and use-cases/scenarios and use-cases/scenarios with features. Consequently, use-cases/scenarios can be a good joining point.

In this context, we propose an approach to PL requirements where the concept of goal is a guide for the selection of variants.

2 Using Goals, Features and Use Cases for Requirement Variability

PL Requirements define the developable products in the PL and their features. There are two specific characteristics different from traditional requirements: the main requirements of every PL product should be determined, even the requirements of the non-developed products (must be forecasted); and it is fundamental to know their commonality and variability, and the dependencies between them.

To represent this kind of information, the requirements are usually structured in definition hierarchies [7] or feature models as in FORM (Feature-Oriented Reuse Method) [4]. Thus, each PL requirement is a prominent and distinctive concept or characteristic that is visible to various stakeholders, or feature. These features are organized by a graphical AND/OR hierarchy diagram, and set whether they are mandatory, optional, or alternative. The main problems with this technique are that a large domain experience is needed, and it is more oriented to the architecture definition than to client presentation or requirements definition. Another problem is that it does not deal with NFR, despite a subtype of capability features is NFR.

Requirement elicitation can also be based on use case analysis (usually a more familiar technique). The basis for the use case support of the variability is the *extend* mechanism [3]. Although this technique is more requirements-oriented, the *extend* mechanism is not enough to represent complex variability. In addition, there is not a mechanism to deal with NFR.

The common characteristic of these two techniques is their solution-space orientation. It is widely recognized that an intentional viewpoint that responses the “whys” is necessary. In order to address this issue, large work has been done in the field of goal-oriented RE. A goal is an objective the system under consideration should achieve [8]. There are two types of goals: (hard) goals and *softgoals*: goals satisfaction can be established through verification techniques [9], but satisfaction of *softgoals* cannot be established in a clear-cut sense (usually used to model non-functional characteristics of the system) [8]. The dependences between goals and *softgoals* can be established. The NFR framework defines these correlations [1].

It is necessary to gather these three viewpoints: intentional (goals), operational (use cases) and functional (features). To achieve this, the use case and scenarios are the natural joining point. There are already several works that relate goals with use cases or scenarios in a single system environment [5] and use cases with features in PL approaches [2]. The idea is to use the first techniques where goals help to construct use cases, and use cases assist in goals discovery. Then, by means the use cases, it is

possible to find features in an easy way with the later techniques. The utility of features is that they are closer to the architecture definition.

Traceability links goals with use cases and use case with features. Subsequently, the goals (intentionality) are related to features, and these to PL architecture and assets. In addition, a direct relationship between goals and features can be found, the goal-oriented task concept (the means for attaining the goals) can be easily mapped to the more implementation kind features. Following this reasoning, we are separating the features in two concepts: the goals that model the capabilities features (general functionality and operations and non-functional requirements), and the tasks, that model all the other features types (operating environment, domain technology and implementation technique). In Fig. 1 the relations between the different PL requirement engineering models are outlined.

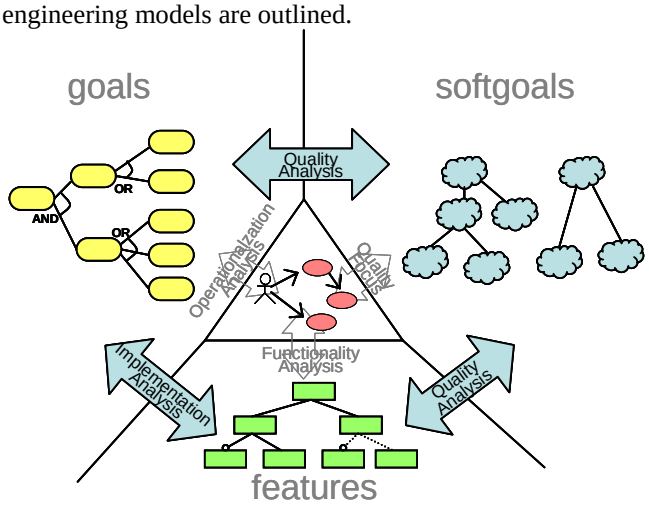

Fig. 1. Relationships between models. Goals are operationalized into use cases, whose variants are focused to different quality factors. Features implement the use cases actions with different contributions to *softgoals*. Also direct relationships (without use cases) are possible.

This approach also helps in the PL variants selection problem. In general, only the functional or operational point of view guides the selection of variants. We think that a more rigorous mechanism of selection is required. In this sense, goal approach allows to relate functional requirements with non-functional ones and to conduct formal analysis [1]. Therefore, goal analysis techniques can be used, and the selection is done from a more natural viewpoint (intentionality).

The idea is that, given a set of requirements variants, encoded in our model, this technique allows to select the better solution (set of features and related assets) according to a basic functional selection (*hardgoals*) and a given quality criteria (*softgoals*). Here, hardgoals limit the variability space and softgoals give the criteria to get the best variant from the limited variability space.

Nowadays, we are focused in the modeling aspect of the approach. We are extending UML to allow modeling goals and features with two new models. In addition, relationships between models are important too: the goals cover use cases, the use cases are described by features, and features operationalize goals.

3 Conclusions and future work

The main contribution of this approach is the definition of a general model of requirement variability that incorporates goals as guide for the variants selection. This approach makes easier and more complete the PL requirement analysis introducing an intentional viewpoint and relating three well-known techniques. In addition, the variability analysis is improved by means of considering non-functional requirements with goal analysis. Consequently, the product architects have a rationale for the selection of those characteristics that had better support the requirements (functional and non-functional) of a new product line member.

We are working on a tool that allows the correlation of goals, use cases and features models. This tool will focus on variant selection from the goals, selecting the desired functionality with the (hard) goals and the preferred quality properties by prioritizing *softgoals*. In this way, we can hide the features, but their relationships and constraints are used to get the variants. These relationships and constraints will be used to find relationships on goals, comparing the possible variants from goal selections. Therefore, it is possible to know if one goal requires other (the goal is only achieved when the other), two goals are mutual-exclusive (there is no variants that achieve both goals), one hints to the other or they mutually hinders.

References

1. Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers 2000.
2. Griss, M., Favaro, J., and d' Alessandro, M. Integrating feature modeling with the RSEB. In Proceedings of the Fifth International Conference on Software Reuse, pages 76--85. IEEE Computer Society Press, 1998.
3. Jacobson I., Griss M., and Jonsson P.: Software Reuse. Architecture, Process and Organization for Business Success. ACM Press. Addison Wesley Longman (1997)
4. Kang, K. C., Kim, S., Lee, J., and Kim, K.: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering, 5:143-168 (1998)
5. Kavakli, E., Loucopoulos, P., and Filippidou, D. Using Scenarios to Systematically Support Goal-Directed Elaboration for Information System Requirements. In Proceedings of IEEE Symposium and Workshop on ECBS'96, Germany:, 1996. pp. 308-314
6. Knauber, P., and Succi, G.: Perspectives on Software Product Lines. ACM Software Engineering Notes, 26(2):29-33 (2001)
7. Kuusela, J., and Savolainen, J.: Requirements Engineering for Product Families. In Proceedings of 22nd International Conference on Software Engineering – ICSE 2000. Pages 60-68. ACM Press (2000)
8. Mylopoulos, J., Chung, L., Yu, E. and Nixon, B.: Representing and Using Non-functional Requirements: A Process-Oriented Approach, IEEE Trans. on Software Eng, 18(6), June 1992 pp:483-497.
9. van Lamsweerde, A. *"Goal-Oriented Requirements Engineering: A Guided Tour"*, Proceedings of the 5 IEEE Int. Symp. on Requirements Engineering, 2001, pp:249-262