On the Impact of Goals on Long-Living Systems

Arfan Mansoor, Detlef Streitferdt

{arfan.mansoor | detlef.streitferdt}@tu-ilmenau.de

Abstract: Changes in software systems are inevitable. Addressing future changes in the requirements engineering phase influences the resulting architecture, improves its maintainability and can prolong the lifetime of software architectures. In this paper, structures in goal models are identified and discussed. Such structures are parts in requirements models with a high probability for future changes. Designing a system to support changes is a way towards a longer lifetime. Variable requirements model parts can be handled using adjustable parts in the design. Thus, the better current as well as future variation points can be identified in the requirements model, the better changes can be handled by a system and, consequently, the longer the lifecycle of software architectures could be.

1 Introduction

The first important step towards developing a system is requirement engineering. Changing requirements while the system is developed, as well as changes over the lifetime of a system, are part of the daily work in software development [Som00]. Thus, the better changes are addressed or foreseen in the requirements engineering phase of the initial system, the longer the lifetime of a system could be, since the system is able to address many changes "by design". Moving from requirements to the system design, a proactive approach to address changes are product lines [BKPS04]. Here, from the beginning on the system is designed for changes, which are used to derive applications based on the product line.

The lifetime of a software system is mainly influenced by the changes that happen over time, especially changes not within the standard use of the software. Changes of requirements or changes of the hardware platform will most likely disrupt the architecture of the software. To prolong the time before an architecture is not usable any more, as many variabilities as possible need to be identified in the requirements engineering phase. After a thorough analysis, such variabilities may be part of the design and thus, the probability to disrupt the architecture can be reduced.

In this paper, a goal-oriented requirements approach with a goal model is used to address possible changes in the requirements engineering phase. Structural patterns in the goal model are used, which refer to the variable parts of the system identified in an early development phase. With a relation of such a variable goal model to a feature model of a product line, a highly configurable system architecture can be developed.

The approach is based on an extensive analysis of the current and future requirements and goals of the system. Based on this analysis a roadmap containing the different versions of the system over time will be developed. All the different versions are based on the same software architecture, in our case a product line, which supports all versions from the beginning on. Thus, changes, starting in the requirements engineering phase, are built into the system. Addressing possible future requirements in the system design ensures their seamless integration into the system over the complete lifetime. The value of the goal model is its structure for variable goal sets, which can be directly transformed into features as described in this paper.

2 State of the Art

Requirements engineering approaches so far are in the what-how range. The idea of goals has introduced "why" concerns in the early stage of requirement engineering i.e., "why" the system was needed and does the requirement specification captures the needs of stakeholders. The idea of goals also emphasized the understanding of organizational context for a new system [Lap05]. Goal based requirements engineering is concerned with the identification of high level goals to be achieved by the system envisioned, the refinement of such goals, the operationalization of goals into services and constraints and the assignment of responsibilities for the resulting requirement to agents such as human, devices and programs [DvL96]. Moving from goal models to the system design, the goal model structures referring to variable elements are responsible for the future capability of a system to react upon changes. The combination of goal models and product lines is very promising to continuously address variability within system development.

Highly configurable, mass customizable systems, are often based on product lines having a core and several, user selectable variabilities. The core and the variabilities are modeled with features, first described 1990 by Kyo C. Kang and further developed in [KKL+98] and [CE00]. Features describe the product line for a future user, so that the user can choose an own application, based on the features of the family. They should describe an outstanding, distinguishable, user visible aspect, quality or characteristic of a software system or system. Features are arranged as a tree, the root of the tree is the concept node, representing the product line itself. Every feature can be optional or mandatory. By definition, product lines are used for many systems and our assumption is, they have a longer lifetime than single systems. Of course, the best support for a long lifetime is to prepare the product line to handle as many of the future changes as possible. Thus, as more degrees of freedom are built into the product line, more changes can be addressed. Sure enough, the degrees of freedom must be based on a thorough requirements analysis phase.

Using goal models to derive feature models is described in [UHS09]. There, the goal models of several system variants are merged and analyzed for common and variable goals. This commonality/variability analysis leads to common (mandatory) and variable (optional) features. The structure of the goal model is kept stable. In addition to the approach described above, the analysis of goal structures together with the clustering of

goals, as described in this paper, leads to an enhanced feature model, with the support for additional future changes. In this paper, the combination of goal oriented requirements engineering and feature modeling results in an enhanced version of the feature model for a product line, addressing the possible variations of the initial goal model. With the analysis and extension of the variable parts in the goal model we expect the resulting product line to have a longer lifetime compared to a product line without the integration of future changes.

3 Variabilities in Goal Models related to Feature Models

The example in this paper is a cycle computer system. This system will be attached to a bicycle, will process data from various sensors, and will communicate with a standard PC. A cyclist will be supported while riding the bike, for maintenance issues, for tour preparations, or to enhance the safety using the bike. One of the results of the requirements engineering phase is a goal model.

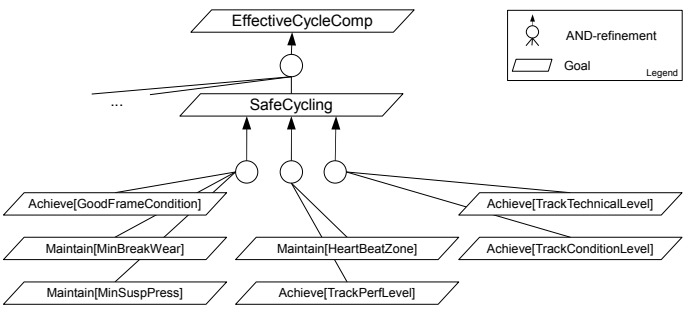

Figure 1: Cycle Computer Goal Model (partly)

As drawn in Figure 1 and part of the overall goal model of the cycle computer, the safety goal has three alternative clusters of subgoals, which are based on the idea of three ways to achieve safety while riding the bicycle. The selected goals per cluster are the result of a goal workshop and may be subject to change in the future development.

The first alternative cluster relates to the technical safety of the bike. To achieve a good frame condition (Achieve[GoodFrameCondition]), any torsion of the frame is permanently measured and analyzed to calculate the current wear level. Thus, the probability of a cracked frame can be derived and used to inform the cyclist of upcoming hazards. Especially in the downhill domain, it is important to know whether the frame would most probably survive the next race. For an enhanced safety information, the wear levels of the brake pads need to be maintained at a minimum level (Maintain[MinBreakWear]) to ensure the proper function. Finally, the suspension pressure also needs to be maintained above a minimum level (Maintain[MinSuspPress]) to prevent serious damage on the bike and possible accidents.

The second alternative of safety is related to cyclists with cardiovascular diseases. Two goals need to be satisfied for a safe bicycle tour. First, the cyclist will be informed whether his current heartbeat rate is inside the heartbeat zone (Maintain[HeartBeatZone]) discussed with the doctor. Thus, this zone needs to be maintained throughout the complete cycle tour, to ensure a healthy trip for the cyclist. In addition, for any planned trip, its performance level (Achieve[TrackPerfLevel]) needs to fit to the cyclists capabilities. Such a selection of trips ensures to stay inside individual heartbeat zones.

The third group of goals refers to the safety of a chosen track. For each part of the track the technical level (Achieve[TrackTechnicalLevel]) should be known, or in other words, what driving experience is needed to handle the track safely. For the complete track, the technical level should fit to the cyclists capabilities. Finally, the condition of the track (Achieve[TrackConditionLevel]) should fit to the cyclists fitness level. Single trails, gravel paths, or paved roads result in different performance requirements on the cyclist, even though the length and gradients of a trip may be the same.

The identification of possible changes at the requirements level is important to address changes properly in the design phase. In the goal model, the relation of goals to subgoals, with subgoals being arranged in clusters, is a very promising pattern for future changes. This structural pattern may be present as alternatives, shown in Figure 1, or as standard goal-subgoal relation. For the cycle computer example the identification of the clusters was based on the domain knowledge and on the different types of cyclists identified in the use case analysis.

With the goal model described above, a first level of variability is achieved by the three alternative clusters of the goal model. Adding new alternatives to satisfy the safety goal is an easy extension of the goal model. The structure of the goal model will not suffer. Adding new goals to a given alternative would also keep the structure stable. The relation of the goal model to a feature model can answer the question for the extensibility of the design. Feature levels in feature models can be used to identify architectural layers and feature subtrees can refer to components. Such components are parameterized (with different binding times) to meet different requirements and realize the derivation of products out of the product line.

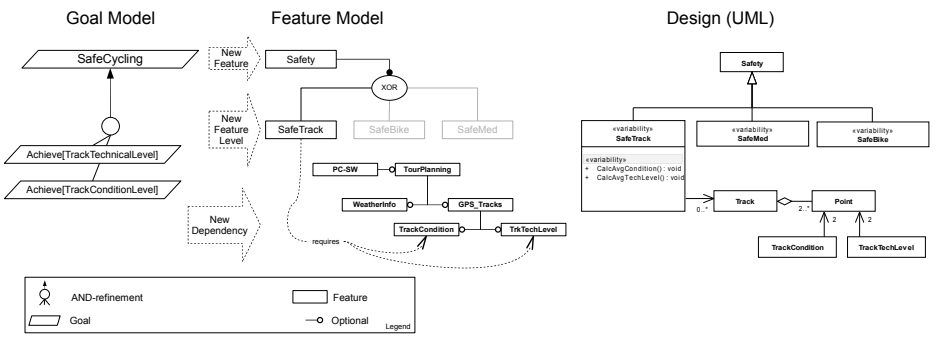

Figure 2: Relation of Goals to Features

For the cycle computer, the proposal for the relation of goals and goal structures is shown in Figure 2. The high level goal of reaching safe cycling trips results directly in a newly introduced feature "Safety". This feature is the root feature of a new feature subtree resulting in a component of the product line, the internal design of the component is shown on the right side of Figure 2. On the next level, each alternative in the goal model is represented by a corresponding feature. Here, three new features, "SafeTrack", "SafeBike", and "SafeMed" are organized below the safety feature. The alternative clusters of the goal model are resolved as XOR-relation in the feature model, to conform to the initial goal model. In case the alternatives are not mutually exclusive, the feature "Safety" would have three optional subfeatures. The subgoals on the left side of Figure 2 are resolved by two new "requires"-dependencies. They refer to the corresponding features in the feature tree of the cycle computer to enable cycling tracks with the needed information about the technical level and the track condition. Finally, as drawn on the right side of Figure 2, the feature model is realized by the UML class model. Here, the realization of the variabilities in the feature model is achieved by generalizing the "Safety"-class with the concrete safety feature. The stereotype <<variability>> together with the generalization is used to realize the XOR-relation of the feature model. Based on the "requires"-relationship, the features "TrackCondition" and "TrkTechLevel" are selected and present in the UML class model.

The identification of variable structures in the goal model points to model parts, where changes can be made without causing structural damage to the overall model. Such parts need to get more emphasis by additional future workshops to identify as many future goals as possible. The result is an enhanced goal model, which can be used as roadmap of the intended system, reaching in the far future. Long living architectural structures should be designed for changes. Feature models can handle the core and any variable parts of a product line. A component-oriented approach to design a product line is common. Thus, changes on the component level of product lines are expected. Relating goal models to feature models leads to additional components and addresses variability inside the component as another level for possible future changes. Thus, a system based on a goal oriented requirements approach and a partly derived feature model, as discussed above, can handle more changes. With this, we expect a longer lifetime for systems based on such an architecture. The benefit of the approach lies in the support of the product line design, namely the feature model. Based on the goal model, the feature model will be enhanced with new structures, which introduce additional levels of variability. As discussed, the level of variability, especially variability in relation to future goals, results in a design supporting the roadmap of the product with its future changes without breaking the software architecture. Automated transformations from the goal model to the feature model are the subject of current research efforts. They will reduce the development time.

4 Summary and Outlook

In this position paper an approach was presented to use the goal model of goal-oriented requirements engineering and identify structural parts in the goal model, which have a high

probability for changes (goals are either added, changed, or removed). These structures are goal-subgoal relations with subgoals that can be clustered in logical groups, all satisfying the upper goal with a different strategy. These structures have been mapped to a feature model, by introducing new features and "requires"-relations. The new features refer to a new system component parameterizable according to the initial goals. The thorough analysis of future requirements and goals will add the variability for future changes to the goal model. The feature model is enhanced to support the future variabilities without breaking the software architecture. Finally, automated transformations from goal models to feature models, which are subject of current research efforts, can reduce the development time. This approach is promising for long living systems, since the resulting product line architecture will address more changeable goals of the initial requirements engineering phase. The formalization of the approach, its full integration into goal oriented requirements engineering and the validation by additional examples are future research topics.

References

[BKPS04] Günter Böckle, Peter Knauber, Klaus Pohl, and Klaus Schmid. *Software Produktlinien*. dpunkt.verlag, 2004.

[CE00] Krzysztof Czarnecki and Ulrich W. Eisenecker, editors. *Generative and Component-Based Software Engineering, First International Symposium, GCSE'99, Erfurt, Germany, September 28-30, 1999, Revised Papers*, volume 1799 of *Lecture Notes in Computer Science*. Springer, 2000.

[DvL96] Robert Darimont and Axel van Lamsweerde. Formal Refinement Patterns for Goal-Driven Requirements Elaboration. In *SIGSOFT FSE*, pages 179–190, 1996.

[KKL+98] Kyo Chul Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. *Ann. Software Eng.*, 5:143–168, 1998.

[Lap05] A. Lapouchnian. Goal-Oriented Requirement Engineering. An Overview of the Current Research. *Technical Report, University of Toronto*, 2005.

[Som00] Ian Sommerville. *Software Engineering 6th Edition*. Addison Wesley, 2000.

[UHS09] Kohei Uno, Shinpei Hayashi, and Motoshi Saeki. Constructing Feature Models Using Goal-Oriented Analysis. In Byoungju Choi, editor, *QSIC*, pages 412–417. IEEE Computer Society, 2009.