International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 4, No. 2
May 2015

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 4, No. 2, May 2015

Research Paper

ON ARCHITECTURAL INFLUENCE USING GOALS AND USE CASES

Arfan Mansoor[1]*, Detlef Streitferdt[1] and Franz-Felix Füßl[1]

*Corresponding Author: ***Arfan Mansoor*** ✉ *arfan.mansoor@tu-ilmenau.de*

Goals have introduced the idea of early-stage requirements engineering in which the focus is on high-level goals, i.e., how to model and analyze the stakeholders needs and interests as compared to late-stage requirements engineering which concerns about operationalization of system objectives. Once high level goals are identified, we refine these high level goals to elicit further goals until we arrive at system requirements (functional and non-functional requirements) satisfying or satisficing these higher level goals. On the other hand use cases are used to represent requirements to achieve a goal but they are more suitable for representing functional requirements. We employ use cases and initial goal model to elicit system requirements and constraints between requirements. The example in this paper is a cycle computer system. This system will be attached to a bicycle; it 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. The combination of use cases and goal models will have a positive influence on the architecture of the system in terms of its quality attributes and variability aspects

Keywords: Requirements Engineering, Goals, Use Case, Software Architecture

INTRODUCTION

Requirements engineering must address the contextual goals, functionalities to achieve these goals and constraints restricting how these functions are to be designed and implemented (Van, 2000). These goals, functions, and constraints have to be mapped to precise specifications of software behaviors and their evaluation over time and across software families (Zave, 1997).

Although Yue was the first one who explicitly stated the representation of goals for requirements completeness – the requirements are complete if they are sufficient to establish the goal they are refining (Yue, 1987), but the idea of goal was already recognized as essential component of requirements engineering by Ross and Schoman (Axel, 2001) as they stated "Requirements definition must say why a system

[1] Software Architectures and Product Line Group, Ilmenau University of Technology, Ilmenau, Germany.

is needed, based on current or foreseen conditions, which may be internal operations or an external market. It must say what system features will serve and satisfy this context, and it must say how the system is to be constructed" (Fowler, 1997). UML also mentions the importance of goals as higher-level abstractions "In my work, I focus on user goals first, and then I come up with use cases to satisfy them; by the end of the elaboration period, I expect to have at least one set of system interaction use cases for each user goal I have identified" (Fowler, 1997). From 10th requirements engineering conference the notion of goals has been explicitly stated in requirements engineering: "Requirements Engineering (RE) is the branch of systems engineering concerned with the real-world goals for, functions of, and constraints on software-intensive systems. It is also concerned with how these factors are taken into account during the implementation and maintenance of the system, from software specifications and architectures up to final test cases." The idea of goals also emphasizes the understanding of organizational context for new systems (Van, 2000). On the other hand use cases are used to describe the functionality requirements expressed by the business analyst. They provide a clear and consistent basis to the system analyst, but it is difficult to identify the use cases at the early stage requirements engineering. The solution is to integrate use case modeling with some early stage requirements engineering approach. Goal oriented requirements engineering is one such approach where we start with the identification of high level goals to be achieved by the system envisioned, then refinement and operationalization of these goals into services and constraints and the assignment of responsibilities for the resulting requirements to agents such as human, devices and programs (Darimont, 1996). In this paper, we will describe how the more abstract high level user goals are first refined into concrete level functional requirements and then representing these functional requirements using use cases. Section 2 establishes 'why' context, section 3 explains how goals are used to move from problem space to solution space. Section 4 presents 'Cycle Computer' example, section 5 includes discussion followed by conclusion.

ESTABLISHING THE CONTEXT

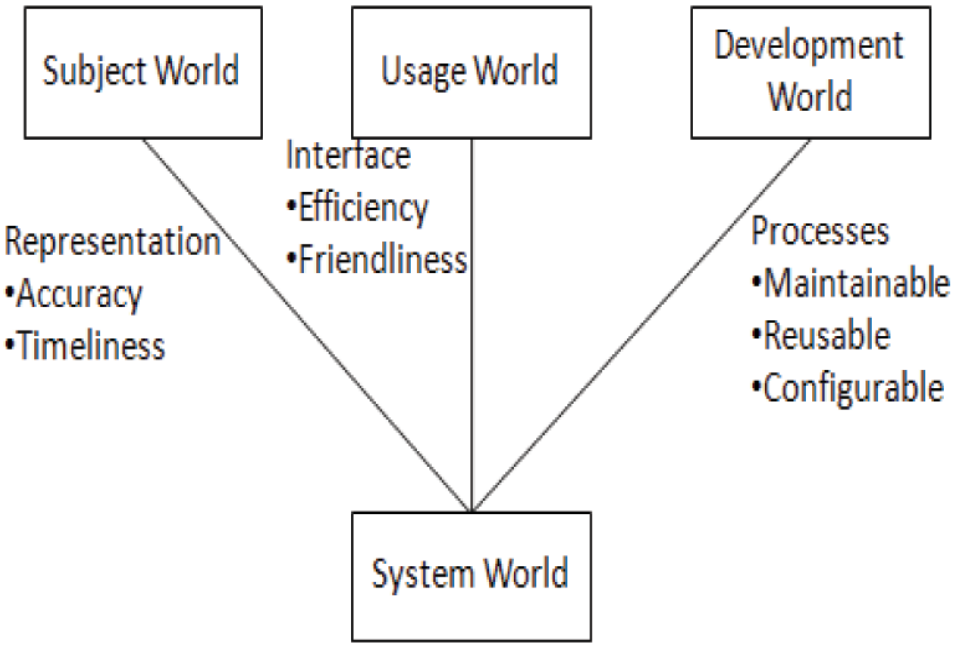
System development is considered successful if the system meets its intended purpose and for this we need to have a thorough understanding of the system and user behavior, the underlying technology and how the elements are going to interact with each other. Pohl (1994) defined information system modeling in four worlds as shown in Figure 1 (1) subject world; (2) usage world; (3) system world; and (4) development world (Reference to figure1). Subject world is the most general one and represent the domain of the system about which it has to provide information. Domain requirements are facts of nature and reflect domain laws imposed by subject world. The usage world contains the direct and indirect users of the system and is described by an organizational structure. The usage world expresses how systems are used to do work, i.e., the tasks, procedures, interactions, etc., performed by agents. The high level business goals, quality factors such as response time, user-friendliness, and functionality with respect to system are defined in usage world. Therefore usage world generates user-defined requirements, which arise from people in the

organization and reflect their goals, intentions and wishes. Subject world relationships to the usage world are governed by legal concerns such as privacy and ownership. The system world is the world of system specifications in which the requirements arising from the subject world (domain requirements) and usage world (user requirements) must be addressed (Matthias, 1994). The relationship between the usage world and system world is named 'intentional relationship' (Colette, 2005). The relationship between subject world and system is described by quality of information factors such as accuracy and timeliness and is called 'representation relationship'. In goal oriented requirements engineering the focus is on 'why', i.e., at requirements engineering level we need to justify the reasons why a system is needed. This is addressed by the relationship between usage world and system world (intentional relationship) which provides the rational for building the system. This relationship concerns the system purpose and relates the system to the goals and objectives of the organization (Colette, 2005). Development world deals with the mapping of the specifications of intentional relationships and representation relationships to the design and implementation and this mapping may enforce new technical requirements, e.g., resource constraints.

PROBLEM SPACE TO SOLUTION SPACE THROUGH GOALS

The problem space is mainly focused on customer needs and problems while in the solution space the focus is on developing products, system architectures, standards and legacy systems (Lehto, 2005).Requirements related to the problem space are considered to

Figure 1: The Four Worlds of Software Development (Matthias, 1994)



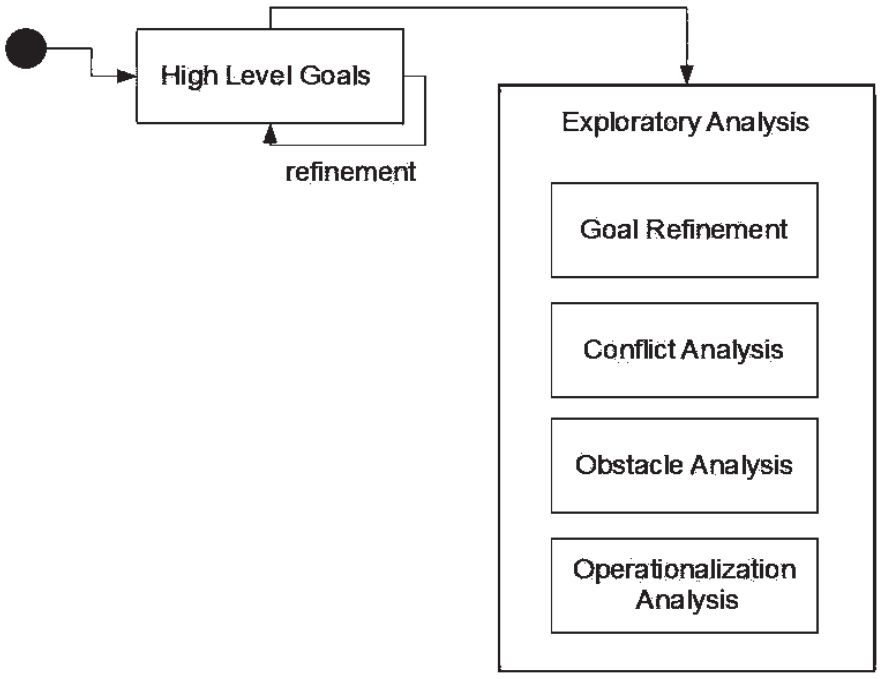
be external requirements (related to customer/ user) and requirements related to the solution space are considered as internal requirements (related to solution and technical stakeholders). For a requirements engineer it is important to distinguish that requirements belong to the problem space or to the solution space. A better quality system is intended to meet its user requirements. RE needs to explore the objectives of different stakeholders and the activities carried out by them to meet these objectives in order to derive purposeful system requirements. Goal-driven approaches aim at meeting these objectives (Colette, 2005).

GORE concerns are classified into major categories, i.e., goal analysis and goal evolution. Goal analysis is the process of exploring gathered documents, ranging from information about the organization, (i.e., enterprise goals) to system specific information for the purpose of identifying, organizing and classifying goals (Anton, 1996). Goal evolution concerns how goals are changed from when they were identified to when they are operationalized. A preliminary analysis of current systems is important source for goal identification. The main sources for identifying goals are to look for intentional keywords provided in documents like interviews, transcripts, mission statements, policy statements (Van, 2000), etc. A common approach is to find deficiencies that can be formulated, negate these deficiencies to produce first list of goals (Axel, 2001). Once high level goals are identified we use goal refinement, goal abstraction, and goal elaboration processes to identify further goals. Goal analysis deals with identification, organization and classification of goals (Axel, 2001). After the initial identification of goals, the new goals are identified by asking 'How' and 'Why' questions. 'How' questions are used to identify more concrete goals from high level goals so that these goals can be easily operationalized and implemented. AND/OR refinement links are used to link these goals. Subgoals may also need to be refined further until we can get assignable subgoals. 'Why' questions are used to elicit abstract goals which are refinements of some parent goal and somehow we have missed that parent goals in the initial goal identification process. Goals may also be elicited using scenarios. In fact there is bidirectional relationship between goals and scenarios. When a goal is identified a scenario can be authored for it and when a scenario is authored it can be analyzed to yield goals (Van, 2002). Scenarios are used to elaborate goals by asking and listing different activities. Scenarios are comprised of actions or behaviors which may be mapped to goals (Anton, 1998). GORE helps us to find more robust requirements through obstacle analysis. Obstacles provide a mechanism for anticipation of exceptional cases. We introduce some obstacle(s) for a goal and then look for way to resolve these obstacles. There are a number of approaches for obstacle resolution like obstacle elimination, obstacle prevention, goal substitution, agent substitution, goal de-idealization, obstacle mitigation, goal restoration (Van, 1998 and 2000). Some of these techniques like goal substitution, goal restoration, obstacle prevention and obstacle mitigation help us to find or define some new goals (Van, 2000) e.g., a goal restoration strategy consists of adding new goals to make obstacles disappear. In addition, numbers of refinement techniques are proposed (Letier, 2001; Dardenne, 1993): agent driven decomposition, case driven decomposition, time driven decomposition. For time driven decomposition formal refinement patterns are

proposed and they are useful for number of reasons, e.g., they allow formal reasoning to be hidden from requirements engineers, they help in detecting incomplete refinements and they allow choices underlying the refinements to be made explicit (Darimont, 1996).

During the goal identification and goal refinement process, the objects concerned by these goals are also identified. The identification and characterization of objects from goals ensure that only those objects are identified which are relevant to a goal (Dardenne, 1993). The identified objects and attributes are defined by relating them to real-world quantities they belong. Next the Identification of agents is made by the actions they are capable of performing on objects (Dardenne, 1993). Basic preconditions and postcondition for actions are also specified. Agents are active system components which have choices of behaviors to ensure goals they are assigned to. The identification of agents and the assignment of agents to actions also help in determining terminating condition for goal refinement (IETIER, 2001); goal refinement stops when a responsibility can be assigned to a single agent. Terminal goals assigned to agents in the software-to-be are known as requirements while goals assigned to the agents in the environment of the software-to-be are known as assumption. Leaf goals assigned to agents need to be operationalized. Constraints are introduced for them which are formal assertions to objects and actions available to agents. Constraints provide information about the requirements that must be met for a goal completion and they also provide insight into issues when goal priorities change (Anton, 1996). Constraints may be divided into two classes, i.e., Hard constraints and Soft constraints. Hard constraints may never be violated while soft constraints may be temporarily

Figure 2: Goal Exploratory Analysis



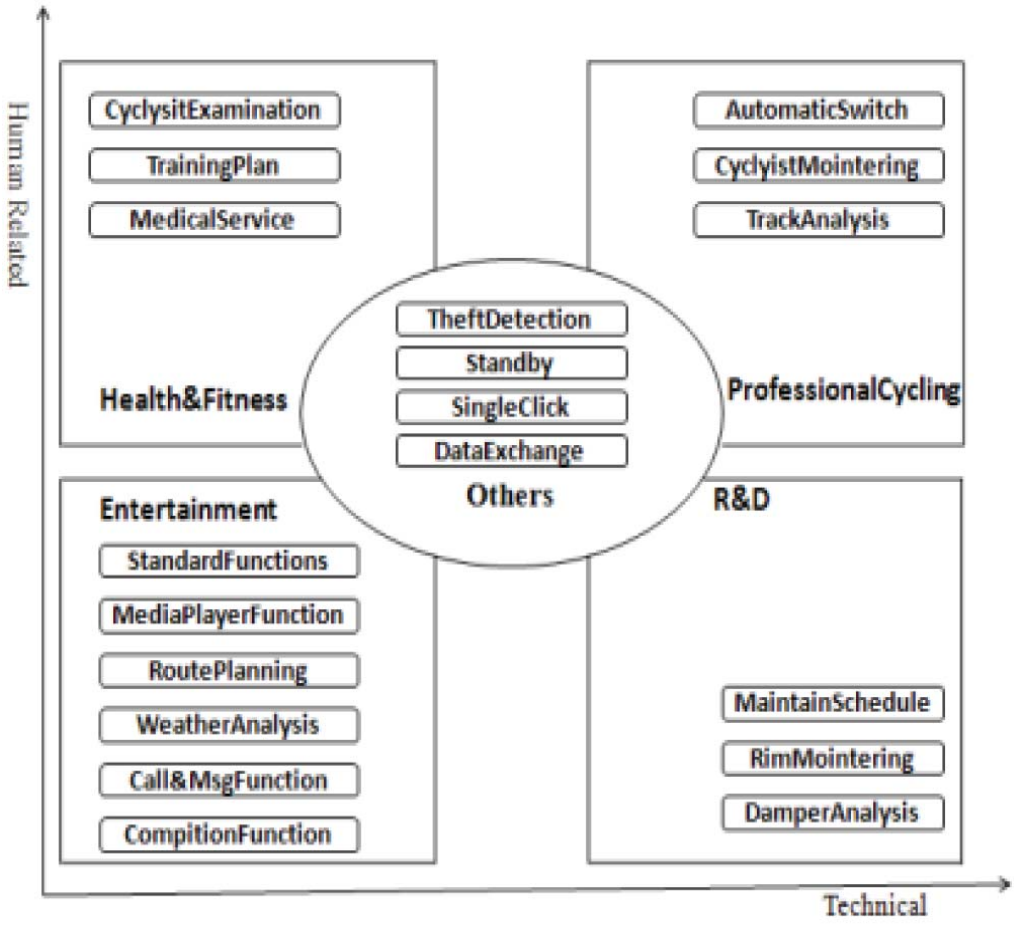
violated. Hard constraints may be safety and time critical constraints, e.g., 'no planes on same portion of air corridor' is a hard constraint. Since soft constraints may be temporarily violated therefore every soft constraint must have one restoration action with it. Starting from problem space (goals and domain objects), operationalization helps to bridges the gap towards solution space (operations representing agent behaviors and objects interacting with those operations). Figure 2 depicts the GORE steps to move from problem space to solution space.

CYCLE COMPUTER EXAMPLE

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. After the brainstorming sessions and group discussions the following major goals were identified to achieve high level goal 'EffectiveCycleComp': Achieve [Entertainmentgoal], Achieve [Health and Fitness goal],

First two goals are more concerned about human related activities while the last two goals address the technical concerns. Each of these

Figure 3: Goals in Human-Technology Portfolio



goals is refined into number of subgoals. Figure 3 shows these high level goals along with their subgoals in Human-Technology portfolio.

To keep things simple we just take one goal and refine it into subgoals to reach at requirement level assignable to agents.According to a March 2010 study published by the IFAK Institute for Market and Social Research, 50% of all respondents are regular bicycle. The proportion of respondents who never goes for entertainment (leisure) bike is just 25% and ratio of professional cyclists is 23% while this ratio is negligible for the R&D related activities. For this reason, the numbers of human-related goals in figure 3 are more than the number of goals with reference to technical goals.

Now we will take one subgoal from Entertainment goal, i.e., Call&Msg Function and refine it into further subgoals until we reach at requirements assignable to agents. The bold line parallelograms in Figure 5 represent requirements.

Use cases are used to document the behavior by capturing the interaction between system and external agents but they do not give an adequate understanding of the interaction that they are intended to describe (Anton, 2000). They provide an overview of restricted aspects of operation model derived from goal model (Axel, 2009). In contrast, in goal model high level goals are refined till they reach at requirements level. Requirements are assigned to agents, objects are identified and operations are operationalized. Interactions

Figure 4: Goals in Human-Technology Portfolio

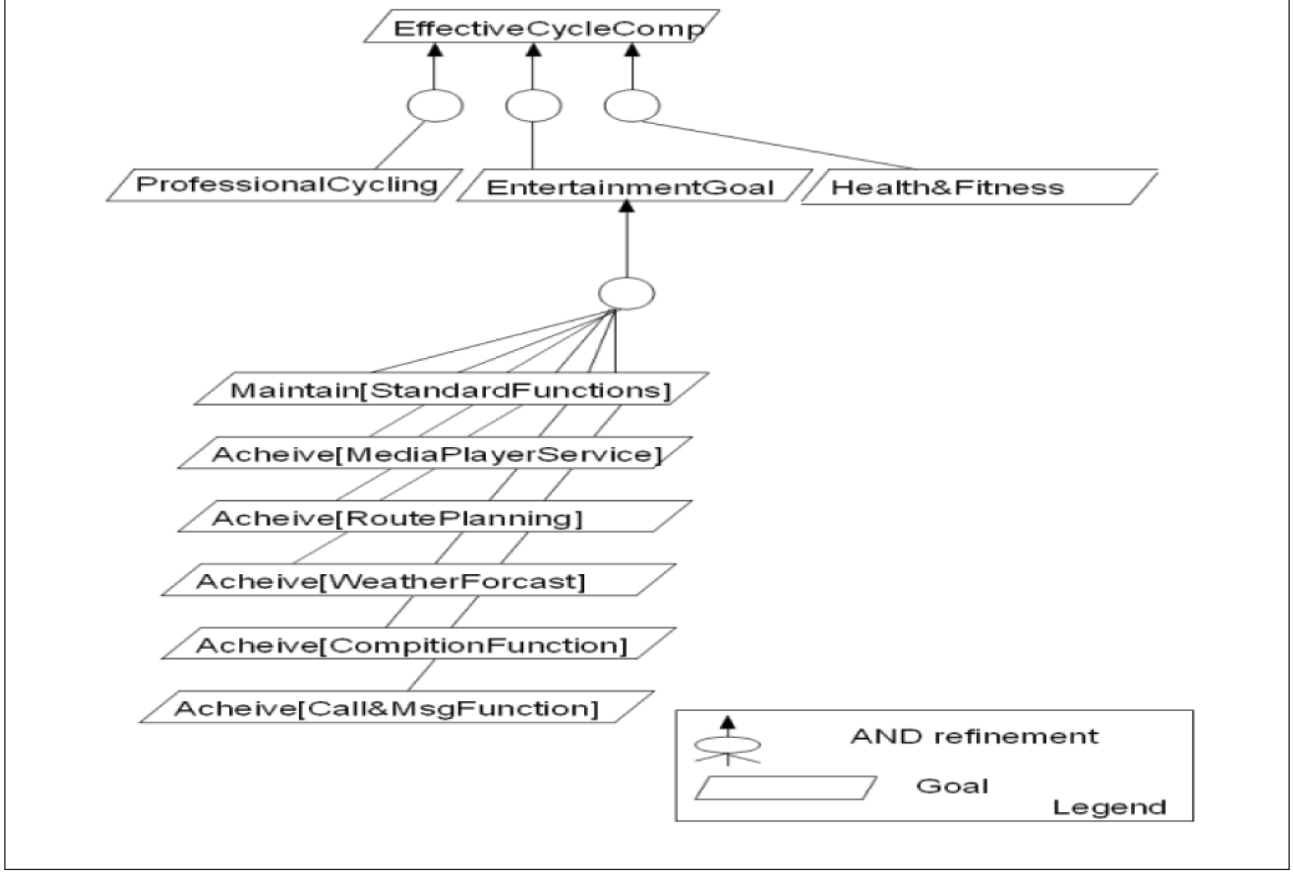


Figure 5: Call&Msgsubgoal Refined To Requirements Level

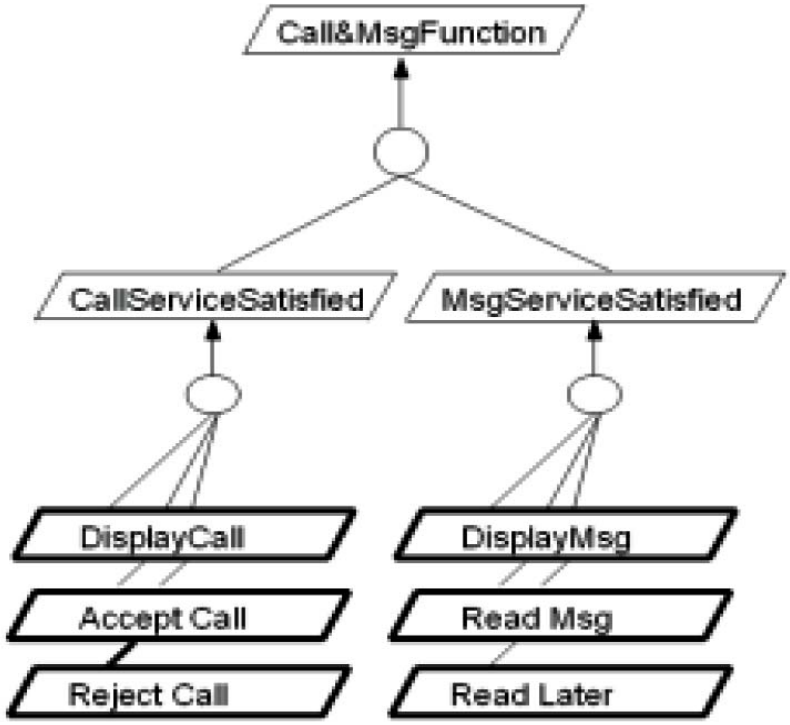


Figure 6: Use Case Model For Call&Msgfunction

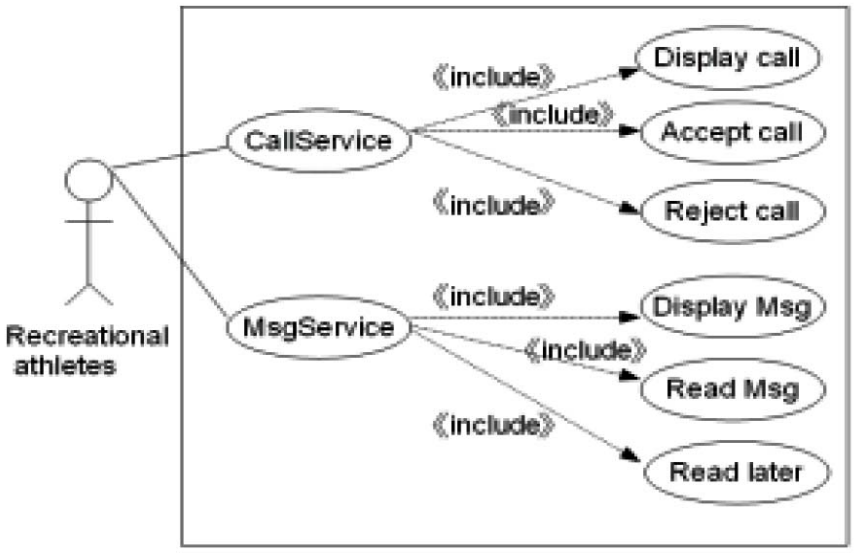
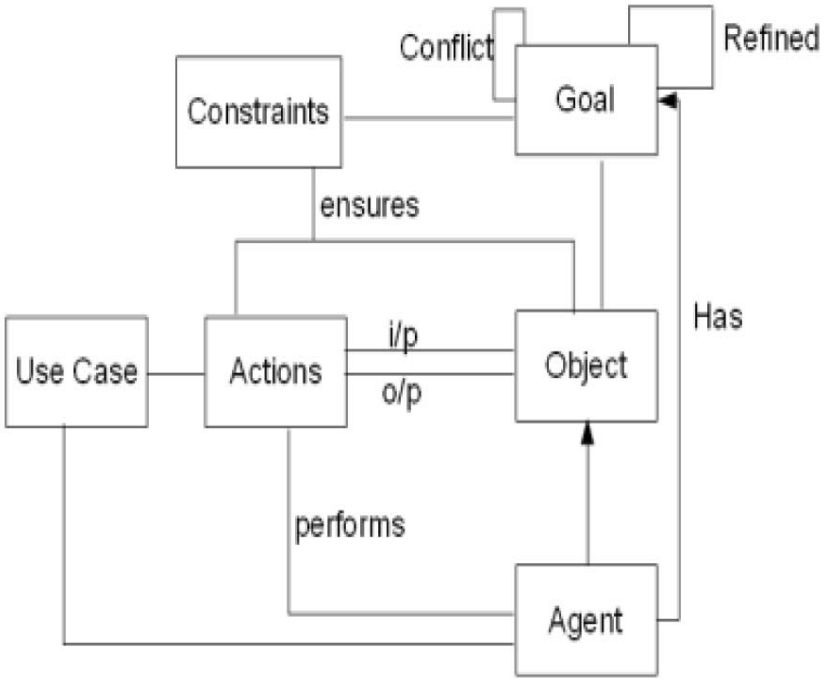


Figure 7: A Goal, Use Case Relation

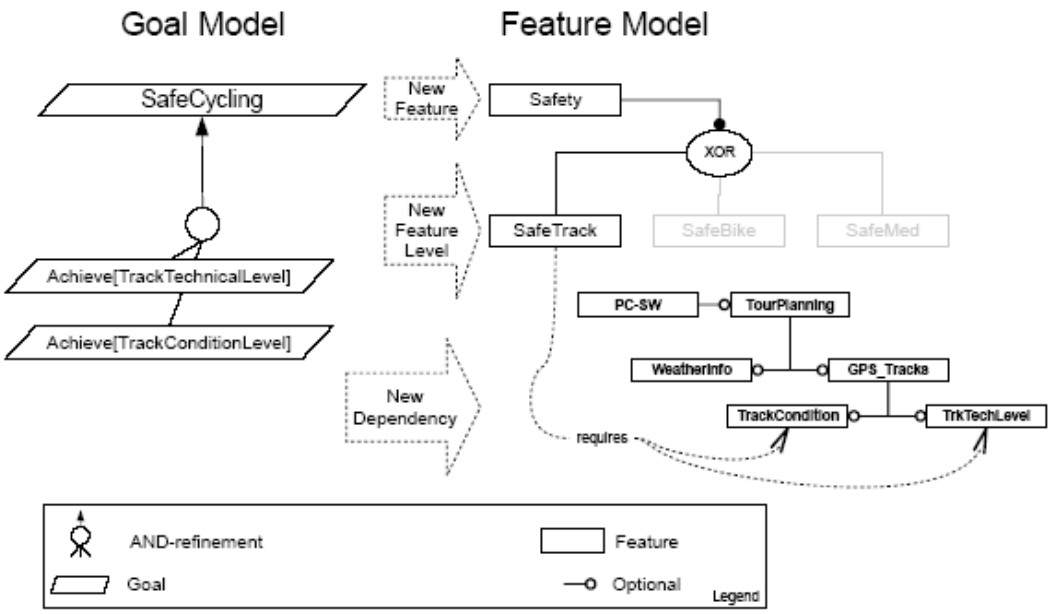


between agents and assigned responsibilities are well captured in goal models. The variants and dependency relationships between goal and subgoals in goal models are described,in use cases, by using ‹‹extends›› and ‹‹include›› stereotypes respectively. For actor in use case which is an agent in goal model, the rectangle contains all the operations that the actor/agent needs to perform. Use cases derived from the goal model in Figure 5 are shown in Figure 6. Figure 7 represents relations between goals and use case.

DISCUSSION

Goal oriented approaches allow us to decompose non-functional requirements in a similar way as functional requirements. Software architectures are generally influenced by these non-functional requirements. A non-functional requirements, also called softgoal in goal oriented approaches, is refined until they are mapped to some functional requirements where they might impact a functional goal in positive or in negative way. In goal models many of the high level design decisions are already made at requirements level that influence software architecture. For example, in goal models we have to select the best option from many alternatives at multiple places e.g., in goal refinement many refinements are possible, in conflict analysis we have to choose among conflicting resolution options, during obstacle analysis different obstacle resolution techniques are available, during operationalization of a goal different operationalization options are available, similarly in responsibility assignment different assignments are possible, etc. In all these steps we have to decide about the best option according to our needs (Axel, 2009). All these

Figure 8: Relation of Goals to Features (Arfan, 2011)



alternatives require some kind of trade off to be decided at the requirements level and that will have influence on the architecture. In Figure 4 we can have two options either to select only one option early in the analysis, e.g., only entertainment goal or to support all there alternative options and let the stakeholders select the best option resulting in customizable solutions. In both cases we will have different architectures. In (Axel, 2003) a complete process of extracting abstract dataflow architecture from software specifications is presented. First from high level goals software requirements are established (in terms of goal models, object model, agent model, and operationalization model), software specifications are derived from these software requirements and as a final step dataflow architecture is developed.

The use of goals at requirements level helps to address the variability. By relating goal models to feature models leads to additional components and addresses variability inside the component as another level for possible future changes. In the goal model, the relation of goals to subgoals with subgoals being arranged in clusters is a very promising pattern for future changes. Thus, a system based on a goal oriented requirements approach and a partly derived feature model can handle more changes. 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. Consider the figure 8 above, 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. 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 (three alternative goals in goal model) below the safety feature. The alternative clusters of the goal model are resolved as XOR-relation in the feature 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 8 are resolved by two "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.

CONCLUSION

Goals have introduced the idea of early stage requirements engineering. They provide a systematic approach to move from problem space to solution space. Goals are reduced/ refined until we reach at requirements assignable to agents. From them we can drive use cases which are used to document the behavior of the system.The assignment of the agents to requirements and their relationships among each other helps to define architectural constraints and from the agent model and operationalization model abstract data flow architecture can be derived. In goal approaches the non-functional requirements are refined the same way as functional requirements, until these non-functional requirements are mapped to some functional requirements. These non-functional requirements have a strong influence on the software architecture. They help to make decisions regarding tradeoff between various alternatives present throughout goal analysis. Each alternative might result in different software architecture.

The goals aid to identify possible changes at the requirements level. 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 or as standard goal-subgoal relation. Combing goal models with feature models assist to handle the variability in software architectures and facilitate in defining more stable software architectures. By relating goal models to feature models as shown in figure 8 leads to additional components and addresses variability inside the component as another level for possible future changes. A system based on a goal oriented requirements approach and a partly derived feature model can handle more changes and thus resulting in more stable software architectures.

REFERENCES

1. Anton A I (1996), "Goal Based Requirements Analysis," Int. Conf. on Requirements Engineering (ICRE 96), Colorado Springs, Colorado, USA, pp. 136-144, April.
2. Anton A I and Potts C (1998), "The Use of Goals to Surface Requirements for Evolving Systems", Proc. ICSE-98: 20thIntrnational Conference on Software Enginering, Kyoto, April.
3. Antón A I, Dempster J H and Siege D F (2000), "Deriving Goals from a Use Case Based Requirements Specification for an Electronic Commerce System" Proc, REFSO'.
4. Axel Van Lamsweerde (2001), "Goal-Oriented Requirements Engineering: A Guided Tour", Fifth IEEE International Symposium on Requirements Engineering (RE'01), Toronto, 2001. re, p. 0249

5. Axel van Lamsweerde (2003), *From System Goals to Software Architecture*, SFM, pp. 25-43.

6. Axel van Lamsweerde (2009), "Requirements Engineering - From System Goals to UML Models to Software Specifications", Wiley, isbn 978-0-470-01270-3.

7. Axel van Lamsweerde (2009), *Reasoning About Alternative Requirements Options*, Conceptual Modeling: Foundations and Applications, pp. 380-397.

8. Arfan Mansoor and Detlef Streitferdt (2011), "On the Impact of Goals on Long-Living Systems", *Software Engineering (Workshops)*, pp. 133-138.

9. Colette Rolland and Camille Salinesi (2005), "Modeling Goals and Reasoning with them", *Engineering and Managing Software Requirements*, Springer Berlin Heidelberg.

10. Dardenne A, van Lamsweerde A and Fickas S (1993), "Goal-Directed Requirements Acquisition", *Science of Computer Programming,* Vol. 20, pp. 3-50.

11. Darimont R and van Lamsweerde A (1996), "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", Proc. FSE'4 - Fourth ACM SIGSOFT Symp. on the Foundations of Software Engineering, San Francisco, October, pp. 179-190.

12. Letier E (2001), "Reasoning about Agents in Goal-Oriented Requirements Engineering", Ph. D. Thesis, University of Louvain, May.

13. Lehto A Jari and M Pentti (2005), "Decision-Based Requirement Engineering Process", Workshop on Collaborative (embedded) Systems Development, 6th International Conference on Product Focused Software Process Improvement, Profes.

14. Matthias Jarke and Klaus Pohl (1994), *Establishing Visions in Context: Towards a Model of Requirements Processes,* EMISA Forum (EMISA), Vol. 4, No. 2, pp. 49-62.

14. van Lamsweerde A and Letier E (1998), "Integrating Obstacles in Goal-Driven Requirements Engineering", Proc. ICSE-98: 20th Intrnational Conference on Software Enginering, Kyoto, April .

15. Van Lamsweerde A (2000), "Requirements Engineering in the Year 00: A Research Perspective", Invited Keynote Paper, Proc. ICSE'2000: 22nd International Conference on Software Engineering, ACM Press, pp. 5-19.

16. van Lamsweerde A and Letier E (2000), "Handling Obstacles in Goal-Oriented Requirements Engineering", *IEEE Transactions on Software Engineering, Special Issue on Exception Handling,* Vol. 26 No. 10, pp. 978-1005.

17. van Lamsweerde A and Letier E (2002), "From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering", Radical Innovations of Software and Systems Engineering in the Future, 9th International Workshop, RISSEF 2002, Venice, Italy, October 7-11, pp. 325-340

18. Yue K (1987), "What Does It Mean to Say that a Specification is Complete?", Proc. IWSSD-4, Fourth International Workshop on Software Specification and Design, Monterey.

19. Zave P (1997), "Classification of Research Efforts in Requirements Engineering", *ACM Computing Surveys,* Vol. 29, No. 4, pp.315-321.

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com