A Method for Reengineering and Prioritizing Goal Models

Jan Willem Harmannij and Ella Roubtsova
Faculty of Informatics, Open University of the Netherlands, PO Box 2960 NL-6401 DL Heerlen, The Netherlands
j.harmannij@studie.ou.nl, ella.roubtsova@ou.nl

Keywords: Reverse Engineering, Goal Modelling, Prioritising of Goals.

Abstract: We present a method for reengineering a goal model from an existing software product, based on the ideas of GORE approaches GBRAM and KAOS. We extend the methods with application of specific ranking criteria to the goal models using Hierarchical Cumulative Voting (HCV). The reengineered goal models are used to evaluate and compare requirements for alternative goals based on the ranking criteria, thus providing input for determining the scope of new products or product versions. The method has been tested in two use cases, one of which is described here.

1 INTRODUCTION

The Belgian company Ferranti Computer Systems has developed the MECOMS™ software product, a business support system for energy and utility companies, providing among others, the Meter Data Management (MDM) and a Customer Information System (CIS). With a network of certified partners across the globe, Ferranti offers worldwide capabilities for customized implementations of the software product. These implementations are based on the standard product developed by Ferranti (Ferranti Computer Systems, 2013).

The need to develop the new software features is caused by the market changes and the demands of different stakeholders, but the budget and development capacities of the company are always limited. The MECOMS™ product organization often faces the questions: Which direction of the product development should have the highest priority and why?

Many software supplier businesses find themselves in this situation. They need a method for prioritising the activities of the product development assuming business goals. There is currently no method for reengineering goal models from existing software products, based on documentation and input from domain experts, and employing these goal models for feature selection for a next product or product version. In this paper, we propose such a method.

We started the method development with an observation that the goals from two dimensions make influence one on another: the business goals and the goals of product development. For modelling of goals in both dimensions, Goal-Oriented Requirements Engineering (GORE) (Lapouchnian, 2005) can be used. The goals in the GORE approaches are represented as trees of sub-goals refined to requirements. However, relating of two trees does not make the task of prioritizing the software development activities observable. That is why we decided to apply the GORE methods only for presentations of the software product goals and relate the business goals to criteria for prioritization. These criteria become the attributes of the software product goals. The values of criteria become the measures of priority. The choice of the criteria and the values of the criteria are fulfilled using the Hierarchical Cumulative Voting method (Berander and Jönsson, 2006) that involves participation of stakeholders. The results of prioritizing are presented in the software product goal model to lead the activities within the development team.

Paper layout: In section 2, we review the basis of goal-modelling that can be used for reengineering the goal models. In section 3, we propose a method for reverse engineering and prioritizing the goal models of a growing product. Section 4 presents the results of testing of the proposed method using two case studies taken from the MECOMS™ software product. Section 5 summarises our results and observations and concludes the paper.

DOI: 10.5220/0004775502040209
In *Proceedings of the Third International Symposium on Business Modeling and Software Design (BMSD 2013)*, pages 204-209
ISBN: 978-989-8565-56-3

2 RELATED WORK

We agree with the opinion of (Regev and Wegmann, 2005) that Goal-Oriented Requirements Engineering is an important contribution by the RE research community and provides many benefits.

Firstly, the GBRAM method (Anton, 1997) emphasizes the initial identification and abstraction of goals from various information sources, using keywords and standard questions for eliciting input from domain experts and classifying this information into concepts like goals, constraints, obstacles and requirements. GBRAM structures the goal modelling process into several phases: exploring, identifying, organizing, refining, elaborating and operationalization.

Secondly, by separating the definition of a goal and requirement, GORE gives the fundament for reasoning about the software product in hand.

Thirdly, a goal model can capture variability through the use of alternative goal refinements, separating stable information from volatile one, and making quantitative and qualitative analysis of these alternatives possible (Lapouchnian, 2005).

Goal models can be developed using a simple AND/OR tree structure, linking a goal G with sub-goals G1, G2, …, G*n*. An AND relation requires all sub-goals to be achieved, while an OR relation provides alternative options. Labelling relations with identifiers like "++" ,"+", "-" and "--", partial goal satisfaction can be modelled (Giorgini et al., 2003). There have been several GORE methods defined over the years that provide more goal modelling semantics; the most prominent being KAOS (van Lamsweerde, 2001) and i* (Yu, 1999).

Very little research is available about reengineering goal models from existing software systems. Only (Yu et al., 2005) describe a method for reverse engineering a goal model, but they focus on design recovery for undocumented software, creating goal models from legacy code using state charts and hammock graphs.

However, most software products still have requirements documented in some form, and functional experts are often willing to provide additional information. This means, that it is not necessary to rely only on legacy code reengineering the goal models. Existing documentation like requirements specification documents, user manuals, marketing brochures and product roadmaps should be used as a starting and reference point. As the domain experts often have a busy schedule, completely relying on their input is a bad strategy (John, 2006).

Prioritization of goals is not part of any of the existing GORE modelling methods.

The prioritizing is used only at the level of software product requirements. According to (Berander and Andrews, 2005), the prioritizing is often performed using a hierarchy of high- and low-level requirements, which are evaluated using methods like the Analytical Hierarchical Process (AHP) or Hierarchical Cumulative Voting (HCV). In our opinion, prioritising at the level of goals provides more information about the motivation of priorities and helps to explain the management decision.

3 A METHOD FOR REENGINEERING AND PRIORITIZING OF GOAL MODELS

In order to design our method of goal-model reengineering we have combined the useful ideas of goal-oriented approaches, and extended them with the process for goal prioritising.

Our method has two different input sorts. At the stage of initial goal model development, the input is the documentation of the software product and the approval of it by the domain experts. At the stage of prioritising, the input comes from the votes of experts. The method consists of seven phases, as shown in Figure 1.

The first four phases (1-4 in Figure 1) copy the GBRAM method (Anton, 1997), but use documentation as the input.

During the Explore phase (1), all relevant documentation is explored. This provides us with unstructured information, like the terminology used to describe functionality, core concepts, modules, etc.

In the Identify phase (2), we try to find the answers on the WHY-questions and create the first draft goal models. The draft models are created using the information obtained during the Explore phase and input from domain experts in modelling sessions.
Draft goal models are corrected by experts live during these sessions, using the KAOS modelling tool Objectiver™ (Respect-IT, 2007). One of the reasons why we have chosen to use KAOS instead other goal modelling methods is that we can easily separate the parts of goal modelling from other KAOS functionality (responsibility modelling, object modelling, operation modelling) by omitting

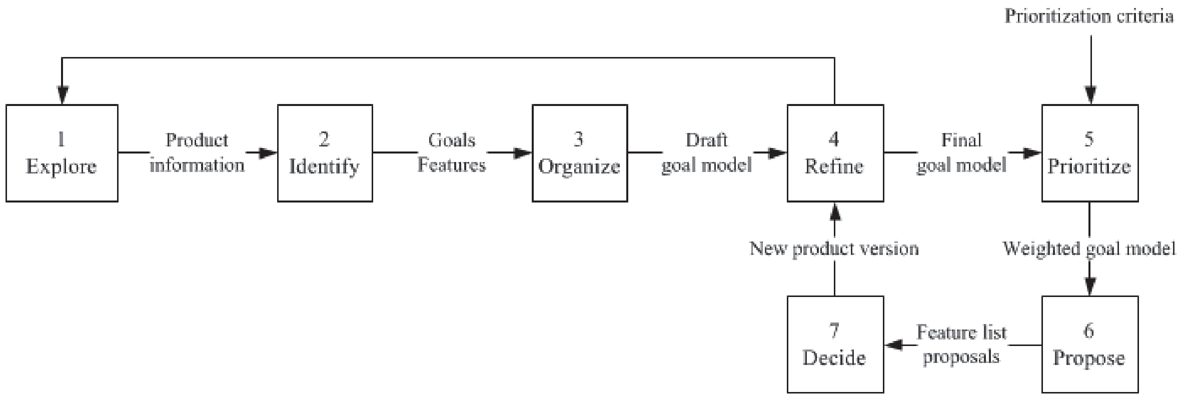

Figure 1: Method for Reengineering and Prioritizing of Goals.

relations with those parts. In other goal–oriented methods, for example in i*, modelling agent relationships and rationales is an essential activity and cannot be separated from the goal model.

An example model is visible in Figure 2. In KAOS, goals, obstacles and features are defined according to the following definitions:

- A goal (a box with thin borders in Figure 2) describes the visual part of an imaginary state of the system which can be achieved by fulfilling all sub-goals (or alternative sub-goals), or in case of the lowest-level goals, when all linked features have been completely implemented;
- An obstacle (a red box, not displayed here) describes an undesirable, but possible system state. Its sub-goals provide solutions to neutralize the obstacle.
- A feature (a thick border box) is a group of related requirements which is implemented in software (van Gurp, 2003). A feature does not describe a system state, but the required functionality for reaching that state. Features are modelled in KAOS as requirements.

The draft goal models are restructured and reorganized during the Organize phase (3). The models are separated into smaller, cross-referenced graphs, checked for completeness and consistency, and documented.

During the Refine phase (4), the organized models are presented to the domain experts and, with their input, finalized. The alternative goal refinement options are added for the next prioritising steps of our method.

The next steps of the method extend the steps of the GBRAM method. We have added these steps and an iterative cycle of their application in order to support the choice or prioritizing criteria and collecting values of those criteria.

After the Refine phase (4), the goal model of the system with alternatives of its further development is ready for the step of prioritising (5). Using a custom plugin for the Objectiver™ tool, the criteria can be added as attributes to the structures presenting goals.

First of all, the right criteria should be determined. Possible criteria include financial value, cost, urgency, readiness, and several others (Ruhe, 2011). It is highly improbable that these criteria are all equally important for the software supplier. Selecting which criteria should be used for evaluating the product features is a management decision. The input from the experts is used at this stage.

When the prioritization criteria have been determined, the feature alternatives are evaluated according to these criteria. This evaluation is the subject of a workshop session with stakeholders from the software supplier company, such as sales representatives, product managers and service managers. These stakeholders define the score of each alternative goal refinement according to the chosen prioritization criteria. We use the Hierarchical Cumulative Voting technique (Berander and Jönsson, 2006), because of its hierarchical nature that is easily combined with goal models, and its suitability for rapidly evaluating a large number of alternatives and multiple stakeholders.

The evaluation results on each criterion are normalized to a scale of 0 to 100 and can be noted as custom goal attributes in the goal models.

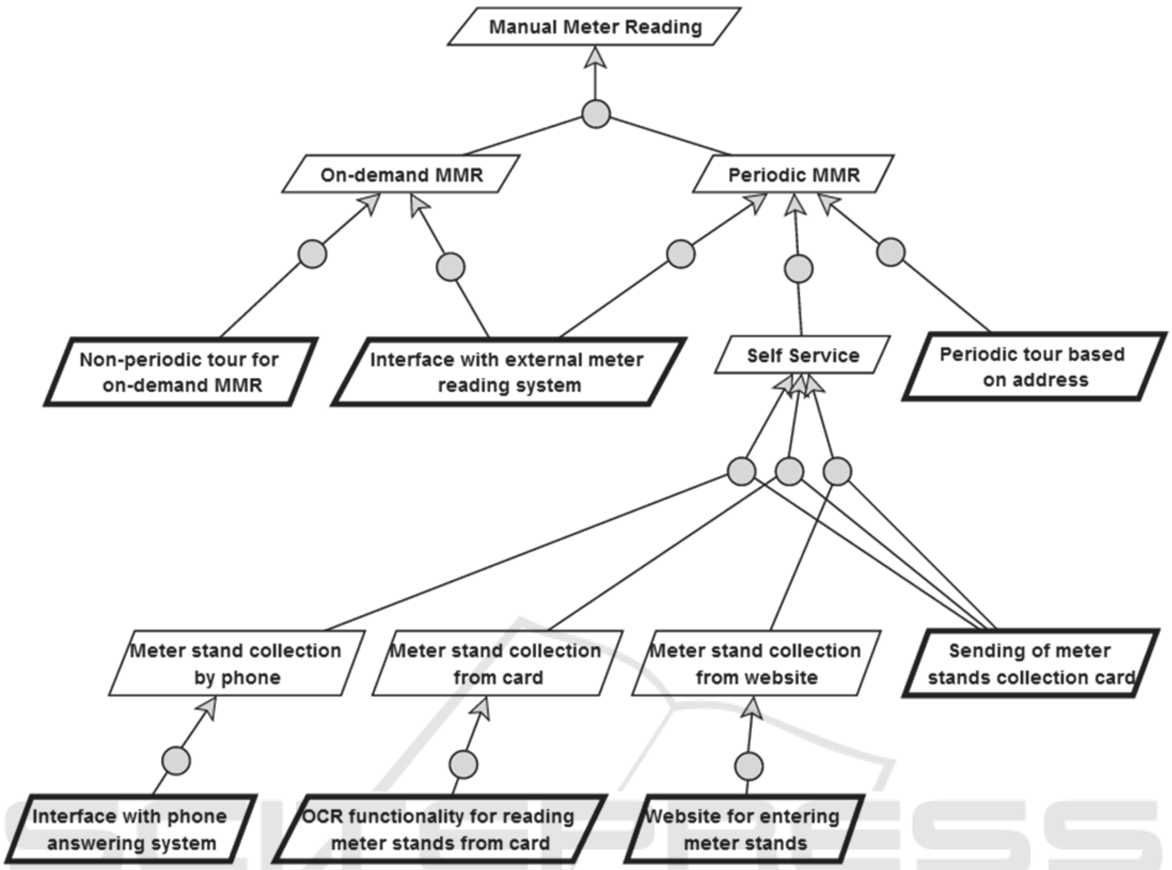
Manual Meter Reading
On-demand MMR
Periodic MMR
Non-periodic tour for on-demand MMR
Interface with external meter reading system
Self Service
Periodic tour based on address
Meter stand collection by phone
Meter stand collection from card
Meter stand collection from website
Sending of meter stands collection card
Interface with phone answering system
OCR functionality for reading meter stands from card
Website for entering meter stands

Figure 2: Part of the MDM goal model, showing "Manual Meter Reading" functionality.

The goal model is now ready to be used for the proposition of the variations that are possible for the next product instantiation or product version. Using the custom prioritization attributes from the goal model and an analyzing tool like the KAOS query language OQL, the alternative scope options can be selected and compared on the basis of the values of the prioritization criteria. This results in a document with feature list proposals and their score according to the criteria that were used for the prioritization of the goals and features.

During the final Decision phase, one of the proposals is selected by a Change Advisory Board. The model can now be refined again (back to phase 4), and a new iteration can be started.

4 APPLICATION OF THE PROPOSED METHOD

The MECOMS™ system was used in two case studies to test our method, "Meter Data Management" (MDM) and "Customer Information System" (CIS).

We describe here the MDM case, for which a partial goal model is shown in Figure 2:

1. Explore. By analyzing the documentation of the Meter Data Management (MDM) functionality, we found the concepts 'Manual Meter Reading' (MMR), 'Periodic MMR', 'On-demand MMR' and 'Self Service'. To identify goals, the documentation is analyzed by asking, "What goal(s) does this statement/fragment exemplify?" and "What goal(s) does this statement block or obstructs?"
2. Identify. The participants named additional goals and product features, which were modelled using the Objectiver™ tool on a large screen. For example, using the question "How do we provide Self Service functionality in the product?" we discovered the sub-goals "Meter stand collection by phone," "Meter stand collection from card" and "Meter stand collection from website." We then proceeded to ask the "How?" questions for the three sub-goals to formulate the features at the bottom of the model.

3. Organize. The model was then restructured into several smaller parts, and analyzed for missing goals and obstructions. These were identified by asking, "Are there alternative ways for achieving this goal?" and "Could this goal somehow be obstructed?" Also, goals that were not yet refined into features were listed. In the MDM case, the model was split into smaller parts for different functional areas. This resulted in 7 interconnected partial goal models.
4. Refine. In a second session with the domain experts, the missing concepts were recognized and added from the documentation. For example, we modelled a feature "Sending of meter stands collection card" as a prerequisite for all "Self Service" sub-goals.
5. Prioritize. Goals and features that were not yet (completely) realized, were prioritized using the Hierarchical Cumulative Voting method (Berander and Jönsson 2006). The results of the prioritization phase are presented in Table 1.
6. Propose. From the results of the prioritization session, a selection of features was proposed for inclusion in a future MECOMS™ release. This proposition was supported by information about the goals that could be achieved by implementing these features, and the HCV scores that were assigned by the stakeholders.
7. Decide. A decision about the scope of the next MECOMS™ release by the Change Advisory Board has not yet been made. This means the case study has not been fully completed yet. However, this last phase is only an endorsement by the CAB of our proposal, and does not add more information than what we already have.

The finalized MDM goal model consists of 37 goals, 4 obstacles and 29 features, modelled in 7 partial (interconnected) goal models. Figure 2 shows a fragment of the goal model, concerning Manual Meter Reading (MMR) functionality. Periodic meter readings can be read into the system by an interface with an external meter reading system, periodic and non-periodic tours, or through one of the several self-service methods. The self-service goal-refinement alternatives are the meter stand collection by phone, mail-in card, or company website. In order to achieve these goals, we need respectively an automated phone-answering system, OCR (Optical Character Recognition) software, or a self-service website. For all three alternatives, a prerequisite is that first a notification is sent to the customers: this is modelled as the feature "Sending of meter stands collection card".

These goal refinements provide alternative solutions for achieving the high-level "Periodic MMR" goal.

Table 1: Score of the MDM features as a result of the Hierarchical Cumulative Voting method.

Feature	Score
Interface with the phone answering system	6,70
OCR functionality for reading meter stands from card	4,54
Sending of meter stands collection card	20,96
Website for entering meter stands	19,66
Periodic tour based on the address	29,19
Non-periodic tour for on demand MMR	18,96

Some observations

The models created for the MECOMS™ products were much larger than anticipated. Still they only cover mostly high-level system goals, and all requirements have been grouped into features in order to keep the models to a reasonable size.

We also found that goal refinements are often a complex combination of AND and OR relations, and sometimes a XOR-like goal-refinement relation was needed but was unavailable or would heavily complicate the model. This resulted in workarounds and, in a few cases, omissions in the models.

We found that there are no suitable concepts for modelling maintenance goals in the current version of the KAOS modelling tool Objectiver™ (Respect-IT, 2007). We modelled these goals as regular (achievement) goals.

The criteria that were used to prioritize the goals and features in the MDM and CIS case studies were determined by the stakeholders. What we didn't expect, was that they opted to use different criteria for goals and for features. The reason for this is that features can be evaluated on a much more detailed level than goals. In our case, the goals were only subjected to the criterion "Ability to sell", while the features were evaluated on six different criteria.

5 CONCLUSIONS

As with any production process, software development requires support for project planning, monitoring the current state of the business product and the next steps of its development at the strategic level. The goal models of a product are instruments supporting the product assessment and planning the steps of its evolution.

However, the goal models of a real product are usually large. In order to be observable the goal models need to have a mechanism of labelling the goals with priorities and filtering the goals with high priority.

In this paper, we have presented a method for building the goal models labelled with criteria measuring priority. Our method includes an expert approach to selecting the labelling criteria related to business goals and collecting the values of the criteria attributed to each goal and associated features.

There are different methods for selecting labelling criteria. Therefore, one of the directions for future research would be goal prioritization based on conflicting criteria. Future work could also focus on the tool support for building the goal models labelled with priorities and filtering the goals having the highest priority. The procedures of assessing the values of the criteria attributed to each goal and filtering could be used periodically and become a valuable mechanism for management of software product-development teams.

ACKNOWLEDGEMENTS

We would like to express our gratitude to Prof. Dr. Rob Kusters at the Open University of the Netherlands for his comments, suggestions and support.

We thank Dr. Robert Darimont, CEO of Respect-IT S.A., for providing us with a research license and customized plugin for the Objectiver™ modelling tool.

REFERENCES

Anton, A. 1997. Goal Identification and Refinement in the Specification of Software-Based Information Systems. Ph.D. thesis, Department of Computer Science, Georgia Institute of Technology.

Berander, P., and Andrews, A., 2005. Requirements Prioritization. In *Engineering and Managing Software Requirements*, 69-94: Springer.

Berander, P. and P. Jönsson, 2006. Hierarchical cumulative voting (HCV) – prioritization of requirements in hierarchies. *International Journal of Software Engineering and Knowledge Engineering, 16(06): 819-849.*

Ferranti Computer Systems. MECOMS™ Product Overview. http://www.mecoms.com/product (accessed March 13, 2013).

Giorgini, P., Mylopoulos, J., Nicchiarelli, E. and Sebastiani, R., 2003. Reasoning with goal models. *Conceptual Modeling—ER 2002*, 167-181.

John, Isabel, 2006. Capturing Product Line Information from Legacy User Documentation. In *Software Product Lines*, edited by Timo Käköla and JuanCarlos Duenas, 127-159: Springer Berlin Heidelberg.

Lapouchnian, A., 2005. Goal-Oriented Requirements Engineering: An Overview of the Current Research.

Regev, G. and A. Wegmann, 2005. Where Do Goals Come From: The Underlying Principles of Goal-Oriented Requirements Engineering. *Proceedings of the 13th IEEE International Conference on Requirements Engineering,* 253-362: IEEE Computer Society.

Respect-IT, 2007. *A Kaos Tutorial*: Objectiver.

Ruhe, Günther, 2011. *Product Release Planning: Methods, Tools and Applications*: Auerbach Publications.

Gurp van, J., 2003. On the Design & Preservation of Software Systems. PhD dissertation, Computer Science Department, University of Groningen, Groningen.

van Lamsweerde, A., 2001. Goal-Oriented Requirements Engineering: A Guided Tour. In *IEEE International Conference on Requirements Engineering*, 249. Toronto, Canada.

Yu, E., 1999. Strategic Modelling for Enterprise Integration. In *Proceedings of the 14th World Congress of the International Federation of Automatic Control, July 5-9, 1999*, 5-9. Beijijng, China.

Yu, Y., Y. Wang, J. Mylopoulos, S. Liaskos, A. Lapouchnian, and J.C.S. do Prado Leite, 2005. Reverse Engineering Goal Models from Legacy Code. *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, 363-372: IEEE Computer Society.