Map-driven Modular Method Re-engineering: Improving the RESCUE Requirements Process

Jolita Ralyté[1], Neil Maiden[2], Colette Rolland[3], Rébecca Deneckère[3]

[1]CUI, University of Geneva, Rue de Général Dufour, 24, CH-1211 Genève 4, Switzerland
ralyte@cui.unige.ch
[2]Centre for HCI Design, City University, Northampton Square, London EC1V OHB, UK
N.A.M.Maiden@city.ac.uk
[3]CRI, University of Paris 1 – Sorbonne, 90 Rue de Tolbiac, 75013 Paris, France
{rolland, denecker}@univ-paris1.fr

Abstract. Configuring and applying complex requirements processes in organisations remains a challenging problem. This paper reports the application of the Map-driven Modular Method Re-engineering approach (MMMR) to a research-based requirements process called RESCUE in order to identify omissions and weaknesses, and to reason about improvements to RESCUE that are currently being implemented. Results have implications for both the scalability and effectiveness of the MMMR approach and for innovative requirements processes such as RESCUE.

1 Introduction

Establishing the requirements for software-based socio-technical systems remains a challenge for many organisations. One reason for this is the increasing complexity of the processes needed to establish such requirements effectively. As a consequence, we need tried-and-tested techniques for manipulating and adapting these requirements processes so that they meet the needs and constraints of client organisations. This paper reports the results of a collaboration between method engineering and requirements engineering researchers to apply one formalism – the MAP formalism [7] – to model and extend the RESCUE requirements process [4].

Our objectives for this work were two-fold: (1) to validate and extend the RESCUE process and improve its effectiveness in future requirements engineering projects, and (2) to test the utility of the map-driven method re-engineering (MMMR) approach [5] for verifying, extending, customising and integrating a full-scale requirements process. The MMMR approach was applied to discover gaps and inconsistencies in the RESCUE process, to extend RESCUE by adding new strategies based on reported good practice and academic research and to enable local customization of RESCUE to meet client process needs.

In the next section we briefly describe the MMMR approach. Section 3 introduces the RESCUE process. Section 4 describes how we re-engineered RESCUE using MMMR. Section 5 reports how RESCUE was extended using this re-engineering work. Section 6 concludes the paper.

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)

2 Map-driven Modular Method Re-engineering (MMMR)

Our approach for modular method re-engineering uses the MAP formalism [7] which provides a process representation system based on a non-deterministic ordering of *intentions* and *strategies*. An *intention* I_i is a goal to be achieved by the performance of an activity whereas a *strategy* S_{ij} is a manner to achieve an intention. Several strategies can be provided by the process model to achieve each intention. Each triplet $<I_i, I_j, S_{ij}>$ in a map is named a *section* and represents a way to achieve the target intention I_j from the source intention I_i following the strategy S_{ij}. This way is captured in the *Intention Achievement Guideline (IAG)*. The arrangement of the sections in a map forms a labelled directed graph with intentions as nodes and strategies as edges. Two types of progression guidelines, *Intention Selection Guideline (ISG)* and *Strategy Selection Guideline (SSG)*, help to select the next intention and the next strategy respectively. The map allows to represent methods in different levels of abstraction. An IAG associated to one map section can also be represented by a map at a lower level of abstraction.

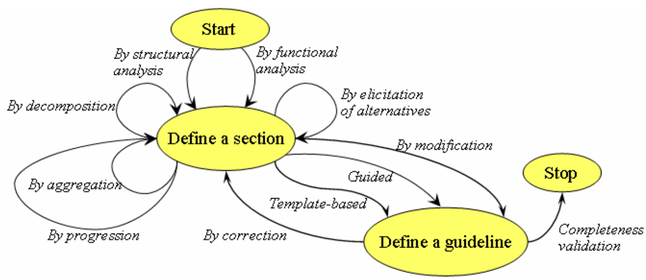

Fig. 1. Process Model for Map-driven Method Re-engineering

Fig. 1 depicts the process model of the MMMR as a map. This process model helps to represent every method as a map with its associated guidelines. According to the map structure, re-engineering the process model of a method consists in redefining it in terms of map sections and their guidelines. For this reason, our process (Fig. 1) seeks to achieve two core intentions: *Define a section* and *Define a guideline* and proposes a set of strategies to satisfy these two intentions. For example, we can identify map sections *By structural analysis* of the method product model elements or *By functional analysis* of its process model steps and activities. The definition of new sections based on the existing ones may be achieved *By decomposition* of an existing section into several ones, *By aggregation of* a set of sections, *By elicitation of alternative* sections to a given one, and *By progression* strategy which helps to define a new section allowing to progress in the map from the existing one. The *Template based* strategy provides a template for every type of guideline (IAG, ISG and SSG) while the *Guided* strategy helps novices with more detailed recommendations for guidelines definition. The *Modification* strategy helps to revise the existing sections while the *Correction* strategy allows to transform the existing sections due to some guideline modification. The *Completeness validation* ends the re-engineering process by verifying if all the guidelines have been defined.

3 Introduction to RESCUE

The RESCUE (Requirements Engineering with Scenarios for User-Centred Engineering) process [3] supports a concurrent engineering process in which different modelling and analysis processes take place in parallel. The concurrent processes are structured into 4 streams: (1) *Human activity modelling* provides an understanding of how people work, in order to baseline possible changes to it [9]; (2) *System goal modelling* enables the team to model the future system boundaries, actor dependencies and most important system goals [10]; (3) *Use case modelling* and scenario-driven walkthroughs enable the team to communicate more effectively with stakeholders and acquire complete, precise and testable requirements from them [8]; (4) *Requirements management* enables the team to handle the outcomes of the other 3 streams effectively as well as impose quality checks on all aspects of the requirements document [6]. Work and deliverables from RESCUE's 4 streams are coordinated at 5 key synchronisation points at the end of the 5 stages:

1. The *boundaries* point, where the team establishes first-cut system boundaries and undertakes creative thinking to investigate these boundaries;
2. The *work allocation* point, where the team allocates functions between actors according to boundaries, and describe interaction and dependencies between these actors;
3. The *generation* point, where required actor goals, tasks and resources are elaborated and modelled, and scenarios are generated;
4. The *coverage* point, where stakeholders have walked through scenarios to discover and express all requirements so that they are testable;
5. The *consequences* point, where stakeholders undertake walkthroughs of the scenarios and system models to explore impacts of implementing the system as specified on its environment.

The synchronisation checks applied at these 5 points are designed using a RESCUE meta-model [4] of human activity, use case and *i** modelling concepts constructed specifically to design the synchronisation checks.

4 Re-engineering RESCUE

The re-engineering of RESCUE started by defining the map at the higher level of abstraction, then by detailing the IAG associated with each of its sections as lower level maps. As RESCUE is process-oriented, the *Functional analysis* strategy was applied to identify the sections of its map. This identification was based on the analysis of the five main RESCUE stages and their objectives. Therefore, five main intentions were identified: *Agree on System Boundaries, Specify Use Cases*, *Generate Scenarios*, *Specify Requirements* and *Validate Requirements*. Besides, the strategies allowing to achieve these intentions was named and ordered. For example, the *Multi-perspective modelling* strategy proposes different models, such as human activity model, context model and use case model, to achieve the intention *Agree on System Boundaries*. The *Creativity workshop driven* strategy proposes to organise creativity workshops in order to *Specify Use Cases*. The intention *Generate Scenarios* is realised *with ART-SCENE* tool which generates scenarios automatically from the previously specified use cases. The *Scenario Walkthrough* strategy is proposed to *Specify Requirements*. The *Impact Scenario Analysis* strategy is used to *Validate Requirements* by analysing the impact of scenario execution and requirements correction, while the Feedback strategy is used for new requirements acquisition and specification if necessary. The RESCUE process ends by delivering the complete set of requirements specifications. We called this strategy the *Delivery* strategy.

Each intention in the RESCUE map should be modelled at the same level of abstraction. However, the scenarios produced by achieving the intention *Generate Scenarios* are only used as a means to specify requirements in a specification that is the main achievement from RESCUE. Therefore, the *Aggregation* strategy (Fig. 1) was applied and the intentions *Generate Scenarios* and *Specify Requirements* were merged into a new one *Specify Complete Requirements*.

As each RESCUE stage ends by synchronising the results of that stage, we added three new sections allowing to reiterate the execution of the intentions *Agree on System Boundaries*, *Specify Use Cases*, and *Specify Complete Requirements*, following the *With synchronisation workshop* strategy. Fig. 2 depicts the obtained first level RESCUE map.

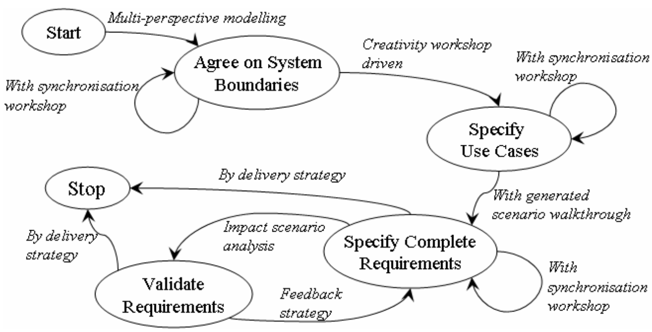

Fig. 2. The first level RESCUE map

Each IAG associated to the RESCUE map can also be defined as a map following the MMMR approach (Fig. 1). Fig. 3 illustrates the map (in solid lines) representing the IAG associated to the RESCUE map section <*Specify Use Cases, Specify Complete Requirements, With generated scenario walkthrough*>. The RESCUE team generates scenarios and walking through them using ART-SCENE to discover requirements by using different walkthrough techniques. Requirements are documented using the VOLERE shell.

5 Validation and Extension of RESCUE

In order to validate and extend the RESCUE process we explored all its second level maps. First of all we analysed the intentions having only one strategy in the current version of RESCUE and considered other possible ways to achieve these intentions.

For example, we considered possible new strategies to enhance the complete requirements specification process captured in the map depicted in solid lines in Fig. 3. As a consequence, 9 new strategies and one new intention were identified and modelled. They are shown in Fig. 3 in dashed lines.

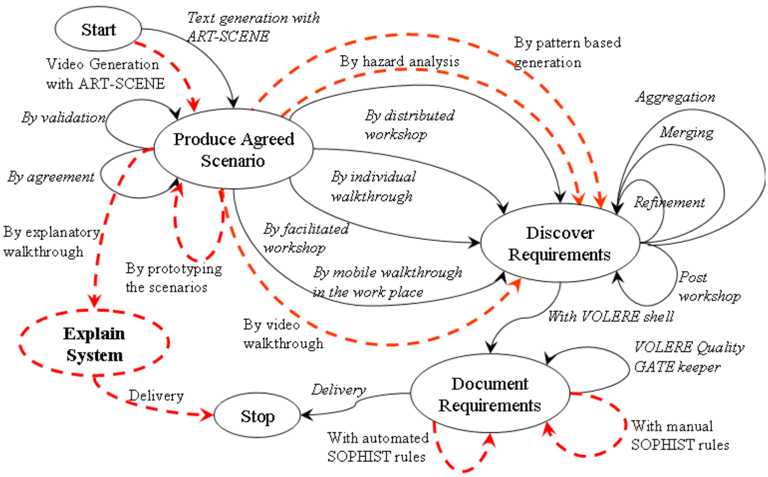

Fig. 3. Solid lines represent the IAG associated to the section <*Specify Use Cases, Specify Complete Requirements, With generated scenario walkthrough*>; dashed lines represent the added new sections.

The new intention, *Explain System*, was added to the map in response to questions about how the process ended. Not all instances of the process result in documented requirements. Scenarios can also be used as effective communication and explanation devices for a new system, independent of their use to discover requirements.

Two new strategies, *Video Generation with ART-SCENE* and *By Video Walkthrough* were designed to produce the agreed scenario and discover requirements by using the recent extensions to ART-SCENE to support multi-media representation of scenarios and video-based scenario walkthroughs [11]. The *Pattern-based generation* strategy for discovering requirements recalled earlier research undertaken by the RESCUE team that is not currently implemented in ART-SCENE. Two types of patterns [2] that guide the discovery and documentation of system requirements will extend the ART-SCENE. Another discovered strategy for discovering requirements from an agreed scenario was *By hazard analysis*. In simple terms, hazard analysis applies simple techniques, such as checklists, to discover hazards associated with a new system. To implement a full hazard analysis strategy within ART-SCENE we will extend its model of abnormal behaviour and state to include a more complete set of hazard classes, then introduce generation settings that will allow a requirements engineer to generate scenarios that are tailored for more rigorous hazard analysis. Finally, we introduced two new strategies based on techniques from the SOPHIST group to document requirements. 25 authoring rules from psychotherapy that assist in the analysis and quality assurance of requirements expressed in text form are reported in [1].

We hypothesise that all these new strategies will result in more correct and consistent discovery and documentation of requirements.

6 Conclusion

This paper reports a research-driven investigation of the MMMR approach to re-engineer the RESCUE requirements process. Findings were relevant for RESCUE and MMMR. Development of the process models revealed important omissions and single strategy intentions in RESCUE that we resolved by adding new intentions and strategies to the process models. This led us to re-investigate existing literature about scenario-driven requirements processes, and to undertake cost-benefit analyses of RESCUE strategies that we will investigate through future RESCUE rollouts. Existing process representations of RESCUE did not afford such analysis. The MMMR process maps also gave the authors confidence that changes to RESCUE were consistent with the existing process. The result was an agenda of improvements to RESCUE and its software tools that we are currently implementing.

The paper also demonstrates the effectiveness of MMMR for modelling large-scale requirements processes. Modelling intentions and strategies, rather than processes and artefacts was tractable and cost-effective whilst still allowing the discovery of missing or weak elements of the process.

References

[1] Goetz, R. & Rupp, C. (2003), 'Psychotherapy for Systems Requirements', Proceedings 2nd IEEE International Conference on Cognitive Informatics, IEEE CS Press, p. 75-80.

[2] Maiden, N.A.M., Cisse, M., Perez, H. & Manuel, D. (1998), 'CREWS Validation Frames: Patterns for Validating System Requirements'', Proceedings REFSQ98 Workshop.

[3] Maiden, N.A.M., Jones, S.V. & Flynn M. (2003), 'Innovative Requirements Engineering Applied to ATM', Proceedings ATM (Air Traffic Management), Budapest, June 23-27.

[4] Maiden, N.A.M., Jones, S.V., Manning, S., Greenwood, J. & Renou, L. (2004), 'Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study', Proceedings CAISE'04, Springer-Verlag LNCS 3084, p. 368-383.

[5] Ralyté, J. & Rolland, C. (2001), 'An Approach for Method Re-engineering'. Proceedings ER'2001, LNCS 2224, Springer, p. 471-484.

[6] Robertson, S. & Robertson, J. (1999), 'Mastering the Requirements Process', Addison-Wesley-Longman.

[7] Rolland, C., Prakash, N. & Benjamen, A. (1999), A multi-model view of process modelling. Requirements Engineering Journal, p. 169-187.

[8] Sutcliffe, A.G., Maiden, N.A.M., Minocha, S. & Manuel, D. (1998), 'Supporting Scenario-Based Requirements Engineering', IEEE Transactions on Software Engineering, 24(12), p. 1072-1088.

[9] Vicente, K. (1999), 'Cognitive work analysis', Lawrence Erlbaum Associates.

[10] Yu, E. & Mylopoulos, J.M. (1994), 'Understanding "Why" in Software Process Modelling, Analysis and Design', Proceedings ICSE'94, IEEE CS Press, p. 159-168.

[11] Zachos, K. & Maiden, N.A.M. (2004), 'ART-SCENE: Enhancing Scenario Walkthroughs with Multi-Media Scenarios', Proceedings RE'2004, IEEE CS Press, p. 360-361.