Applying Modular Method Engineering to Validate and Extend the RESCUE Requirements Process

Jolita Ralyté[1], Neil Maiden[2], Colette Rolland[3], and 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. RESCUE had evolved in the light of research findings and client requests. The MMMR approach was applied to model the RESCUE process, 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. Although some robust processes are emerging, such as REVEAL [4], KAOS [16] and RUP [6], we still lack 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 [14] – to model and extend the RESCUE requirements process [10].

Our objectives for this work were two-fold. The RESCUE team wanted to validate and extend the RESCUE process and improve its effectiveness in future requirements engineering projects. The authors of the MAP formalism wanted to test the utility of the map-driven method re-engineering (MMMR) approach [12] for verifying, extending, customising and integrating a full-scale requirements process. RESCUE is a complex and multi-disciplinary process that has been used to specify requirements for several air traffic management systems [9; 10]. In spite of these successes the lack of a formal representation of the process led to concerns about the completeness and effectiveness of RESCUE. Therefore the MMMR approach was applied to achieve three goals. Firstly, it was applied to verify the RESCUE process to discover gaps and inconsistencies in the process. In the MMMR approach this was achieved by

L. Delcambre et al. (Eds.): ER 2005, LNCS 3716, pp. 209 – 224, 2005.

discovering missing and single strategies for achieving process intentions. Secondly it was used to extend RESCUE by adding new strategies based on reported good practice and academic research for scenario-based requirements processes. Thirdly, the maps were used to enable local customization of RESCUE to meet client process needs and constraints.

The remainder of this paper is in 5 parts. Section 2 describes the MMMR approach. Section 3 describes 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. The paper ends with a review of this work, and outlines future work.

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

Our approach for method re-engineering uses the MAP formalism [14]. This section briefly introduces this formalism and describes the Map-driven Modular Method Re-engineering (MMMR) approach.

2.1 The Map Formalism

The MAP formalism 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 an approach, a manner to achieve an intention. Following the Map formalism, several strategies can be provided by the process model to achieve each intention.

Another key element of a map is a triplet $<I_i, I_j, S_{ij}>$ named a *section*. A *section* represents a way to achieve the target intention I_j from the source intention I_i following the strategy $S_{ij.}$ Each section of the map captures the condition to achieve an intention and the specific manner in which the task associated with the target intention can be performed. This manner is called an *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. The directed nature of the graph shows which intentions can follow each other. Two types of progression guidelines, *Intention Selection Guideline (ISG)* and *Strategy Selection Guideline (SSG)*, help to select the next intention and the next section respectively.

The process model represented in the form of a map has a modular structure; each of its IAGs represents a more or less autonomous guideline which can be *simple*, *tactical* or *strategic* with regard to its content, formality, granularity, etc. A *simple guideline* may have an informal content and advise on how to proceed to handle the situation in a narrative form. A *tactical guideline* is a complex guideline, which uses a tree structure to relate its sub-guidelines. This guideline follows the *NATURE* process modelling formalism [5], which proposes two different structures: the *choice* and the *plan*. Each of its sub-guidelines belongs to one of these types of guideline. Finally, the s*trategic guideline* is a complex guideline using the MAP formalism. Therefore, 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.

2.2 The Process Model for Map-Driven Modular Method Re-engineering

We represent the process model of every method as a map with its associated guidelines. As mentioned above, the map structure offers the re-engineered method a high degree of modularity and provides means to evaluate this method, to decompose it into method chunks, to enhance it by adding new strategies to achieve its intentions, etc. As shown in Fig. 1, our MMMR process model is also represented as a map.

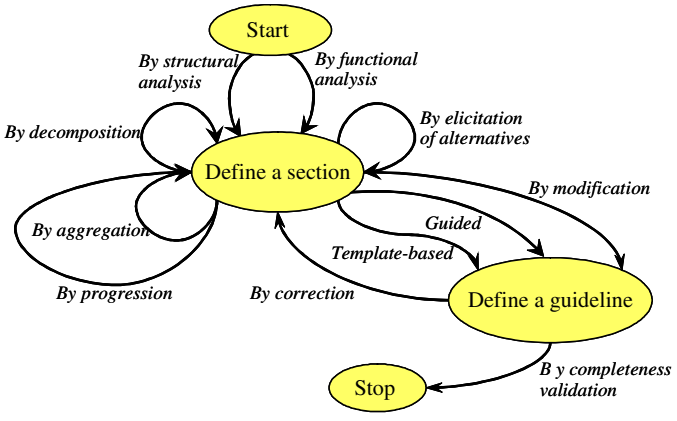

Fig. 1. Process Model for Map-driven Method Re-engineering

According to this map structure, re-engineering the process model of a method requires us first to redefine it in terms of map sections and their guidelines. For this reason, our process seeks to achieve two core intentions: *Define a section* and *Define a guideline* and proposes a set of strategies to satisfy them. For example, there are two strategies *By structural analysis* and *By functional analysis* to achieve the intention *Define a section*. The *Structural analysis* strategy is recommended when the re-engineered method does not provide the method engineer with a formally defined process model but with a simple description of the product to construct. This strategy uses a glossary of generic process intentions to support the discovery of method intentions. On the other hand, the *Functional analysis* strategy should be used if the method has a defined process model that is expressed in the form of steps and recommended actions. This strategy helps to identify the method map sections from these actions, and steps.

When a section is defined, the method engineer can either define the guidelines associated to this section (to progress to the intention *Define a guideline*) or define new sections (to repeat the intention *Define a section*).

The definition of the section guidelines consists of describing the IAG associated with each section, the ISG associated to a set of sections having the same source intention and different target intentions and the SSG associated to every set of parallel sections. The definition of these guidelines is supported by two strategies: the *Template based* strategy and the *Guided* strategy. The former provides a template for every type of guideline and provides advice to experts whereas the latter helps novices by providing more detailed recommendations.

The definition of new sections based on the existing ones (Fig. 1) may be achieved in four different ways, or manners: *By decomposition* of an existing section into several ones, *By aggregation of* a set of sections into a new one, *By elicitation of alternative* sections to a given one, i.e. having an alternative strategy or an alternative source or target intention, and *By progression* strategy which helps to define a new section allowing to progress in the method map from the existing one.

Modifications of the sections (decomposition, aggregation) imply the revision of the associated guidelines if already defined. The *Modification* strategy guides the method engineer to accomplish these transformations. In a similar manner, the process of guidelines definition may imply the transformation of existing sections. For example, the decomposition of an intention achievement guideline could lead to decomposition of the corresponding section. Such transformations can be accomplished following the *Correction* strategy.

The method re-engineering process ends with the *Completeness validation* strategy. This strategy helps to verify if all of the guidelines associated to the map sections have been defined. Due to space limitation we cannot present all of these guidelines. However some of them will be further explained in section 4 when used to re-engineer the RESCUE approach that we introduce in the next section.

3 Introduction to RESCUE

The RESCUE (Requirements Engineering with Scenarios for User-Centred Engineering) process [9] supports a concurrent engineering process in which different modelling and analysis processes take place in parallel. The concurrent processes are structured into 4 streams shown in Fig. 2. Each stream has a unique and specific purpose in the specification of a socio-technical system:

- Human activity modelling provides an understanding of how people work, in order to baseline possible changes to it [17];
- System goal modelling enables the team to model the future system boundaries, actor dependencies and most important system goals [18];
- 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 [15];
- 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 [13].

Sub-processes during these 4 streams (shown in bubbles in Fig. 2) are co-ordinated using 5 synchronisation stages that provide the project team with different perspectives with which to analyse system boundaries, goals and scenarios. These 4 streams are supplemented with 2 additional processes. Acquiring requirements from stakeholders is guided using ACRE [7], a framework for selecting the right acquisition techniques in different situations.

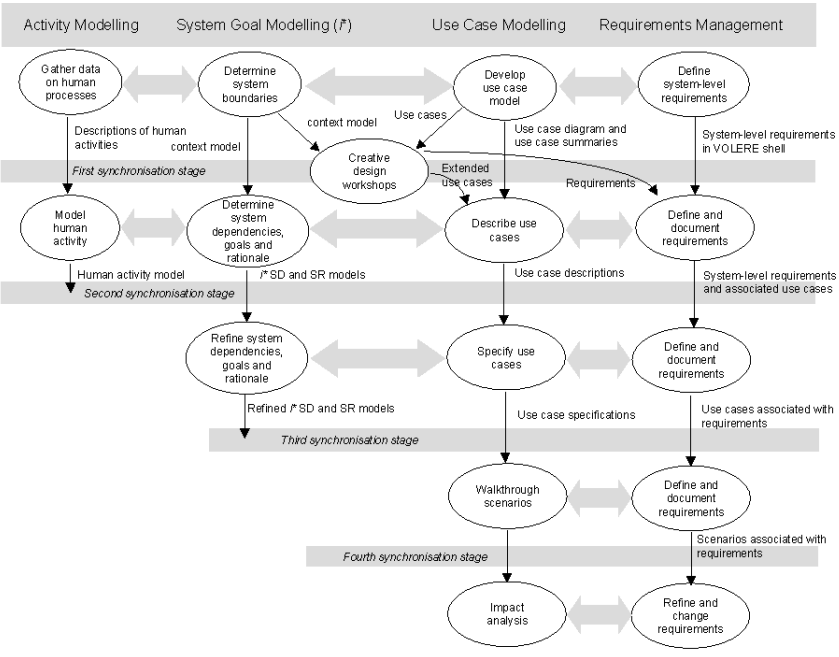

Fig. 2. The RESCUE process structure

Creativity workshops normally take place after the first synchronization stage, to discover and surface requirements and design ideas that are essential for *i** system modelling and use case specification during stage 2. Stage 1 inputs to the workshops include the system context model from the system goal modelling stream and use case diagrams from the use case modelling stream, both shown in Fig. 2.

Scenarios walkthroughs to discover more complete requirements take place during stage 4. Scenarios are generated and walked through using ART-SCENE, a web-based scenario environment that was designed using one cognitive principle often exploited during prototyping – that people recognise items, for example scenario events generated by ART-SCENE, better than they recall them from memory [2]. For each generated normal event and alternative course the facilitator guides stakeholders to recognize, discover and document requirements.

Work and deliverables from RESCUE's 4 streams are coordinated at 5 key synchronisation points at the end of the 5 stages shown in Fig. 2, implemented as one or more workshops with deliverables to be signed off by stakeholder representatives:

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.

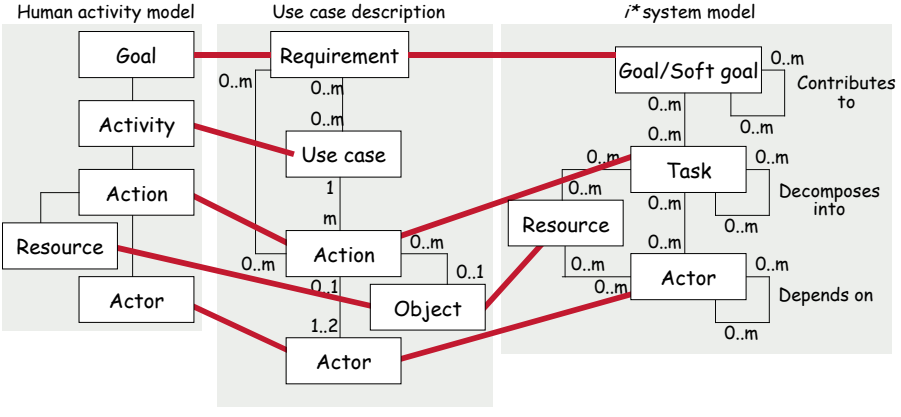

Fig. 3. The RESCUE concept meta-model as a UML class diagram showing mappings between constructs in the 3 model types

The synchronisation checks applied at these 5 points are designed using a RESCUE meta-model of human activity, use case and *i** modelling concepts constructed specifically to design the synchronisation checks. It is shown in simplified form in Fig. 3 – the darker horizontal lines define the baseline concept mappings across the different models used in RESCUE. In simple terms, the meta-model maps actor goals in human activity models to requirements in use case descriptions and *i** goals and soft goals. Likewise, human activities map to use cases, and human actions to use case actions that involve human actors in use cases and tasks undertaken by human actors in *i** models. Human activity resources map to *i** resources and objects manipulated in use case actions, and actors in all 3 types of model are mapped. The complete meta-model is more refined. Types and attributes are applied to constrain possible mappings, for example use case descriptions and *i** models describe system actors, however only human actors in these models can be mapped to actors in human activity models.

RESCUE was originally developed to support the scenario-driven specification of requirements using ART-SCENE [11]. Streams such as use case modelling were developed to provide direct inputs into ART-SCENE's scenario generation tool, and other streams such as activity modelling and goal modelled were added to improve the completeness and correctness of the use case specifications. Other changes were made in response to client requests as the process was rolled out on different projects. At no time was RESCUE re-engineered systematically to improve its completeness, to enable it to be customized to meet the needs of different clients or to support effective integration with other processes with the RUP. Therefore, in the summer of 2004, a collaborative exercise to model and re-engineer RESCUE using MMMR was undertaken.

4 Re-engineering RESCUE

The complexity of RESCUE meant that its process should be represented at different levels of abstraction. The re-engineering activity 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.

4.1 Defining First Level RESCUE Map

As RESCUE is process-oriented, we apply the *Functional analysis* strategy (Fig. 1) to re-engineer its process into a map. This strategy recommends to identify first the main method intentions and the strategies proposed by the method to satisfy these intentions, and finally, to order these intentions and strategies in the map.

Defining RESCUE map sections. The guideline associated to the *Functional analysis* strategy recommends analysing the process steps to identify the key product parts that are target products of these steps, and to couple them with some of the generic intentions provided in our method base glossary representing the objective of each step. Therefore, each method step is defined by one or more intentions. As shown in Fig. 2, RESCUE is divided into five main stages. Based on these stages the core intentions of the RESCUE map were identified as follows:

- The objective of the first RESCUE stage is to identify the boundaries of the system under consideration and to approve them. We called this intention *Agree on System Boundaries*. The RESCUE approach uses different models, such as human activity model, context model and use case model, to achieve this objective. As a consequence, we named the strategy that achieves this intention the *Multi-perspective modelling* strategy.
- The second RESCUE stage is called work allocation. It is intended to deliver use case specifications for each actor of the system. We called the corresponding intention *Specify Use Cases*. The achievement of this stage is mainly based on organisation of creativity workshops. Therefore we named the strategy *Creativity workshop driven*.
- The third RESCUE stage results in the automatic scenario generation from the use cases using ART-SCENE. Therefore, the name of the intention is *Generate Scenarios* and the corresponding strategy is called *With ART-SCENE.*
- During the fourth RESCUE stage the stakeholders are invited to walk through the generated scenarios to discover and express requirements so that they are testable. As a result, the main intention of this stage is *Specify Requirements* and the strategy is called *With Scenario Walkthrough*.
- Finally, the fifth RESCUE stage deals with requirement validation by analysing the impact of scenario execution and requirements correction, and new requirements acquisition and specification if necessary. Consequently, we could define two main intentions: (1) *Validate Requirements*, which can be achieved by following the *Impact Scenario Analysis* strategy and (2) *Specify Requirements*, which is achieved by following the *Feedback* strategy.
- The RESCUE process ends by delivering the complete set of requirements specifications. We called this strategy the *Delivery* strategy.

The next step recommended by the guideline consists in ordering the identified intentions and strategies in a process map. For every intention and one associated strategy we have to identify the pre-conditions that should be satisfied in order to reach the intention following this strategy. That is, we need to identify the product necessary to achieve this intention (the required input product) and then to identify which intention produces this product. For example, the achievement of the intention *Specify Use Cases* using the *Creativity workshop driven* strategy requires as input products the models that are obtained during the first process stage, that is by achieving the intention *Agree on System Boundaries*. The intention *Specify Requirements* requires as input product scenarios that are obtained by achieving the intention *Generate Scenarios*. Furthermore, the intention *Specify Requirements* was identified twice (in stages four and five), but it is evident that we put this intention in the map only once. In a similar manner we arranged the identified intentions and strategies in the map and we obtained the first version of the RESCUE map shown in Fig. 4.

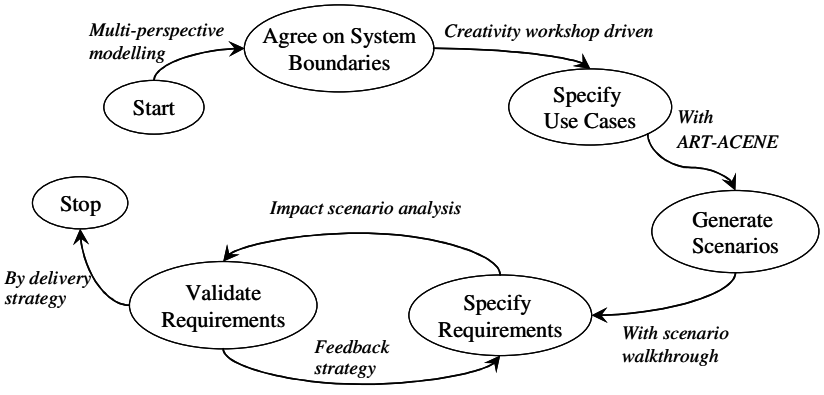

Fig. 4. First version of the RESCUE map

Following Fig. 1, the next step is to refine the obtained map sections by applying different strategies or to define different guidelines associated to this map. Let us refine the map first.

Refining the RESCUE map. Each intention in the RESCUE map should be modelled at the same level of abstraction. The intentions described in the first level RESCUE map represent the main products that are obtained by applying RESCUE. 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, we merged the sections <*Specify Use Cases, Generate Scenarios, With ART-SCENE*> and <*Generate Scenarios, Specify Requirements, With scenario walkthrough*> by applying the *Aggregation* strategy (Fig. 1) to obtain a new section <*Specify Use Cases, Specify Complete Requirements, With generated scenario walkthrough*>.

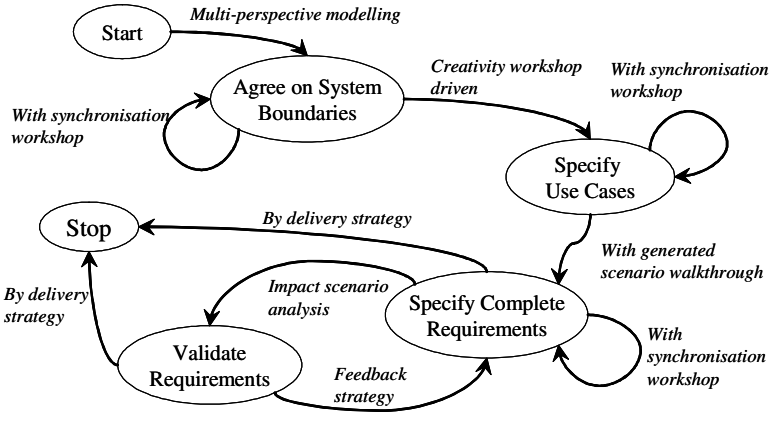

Fig. 5. The RESCUE map

The *Progression* strategy (Fig. 1) allowed us to add new sections to the RESCUE map. Each RESCUE stage ends by synchronising the results of that stage. To describe these synchronisations we added three new sections to the RESCUE map: (1) <*Agree on System Boundaries, Agree on System Boundaries, With synchronisation workshop*>, (2) <*Specify Use Cases, Specify Use Cases, With synchronisation workshop*> and (3) < *Specify Complete Requirements, Specify Complete Requirements, With synchronisation workshop*>.

In a similar manner we added a new section <*Specify Complete Requirements, Stop, By delivery strategy* > that ends the RESCUE process after achieving the intention *Specify Complete Requirements*. Fig. 5 depicts the obtained first-level RESCUE map.

4.2 Defining Second Level Maps

Each IAG associated to the RESCUE map can also be defined as a map. Therefore, we re-applied the map definition process as defined in our MMMR (Fig. 1). Because of the lack of space, we do not describe how all of the second level maps were developed. Fig. 6 illustrates the map representing the IAG associated to the section <*Start, Agree on System Boundaries, Multi-perspective modelling strategy*> of the RESCUE map (Fig. 5). According to this map, the requirements engineer has to work with four artefacts – the human activity, context and use case models, and the system requirements documentation, to define system boundaries. The RESCUE approach provides several different ways to achieve the four corresponding intentions. For example, there are two strategies, *With light weight ethnography techniques* and *With DFD techniques*, to *Model Human Activity*. The *Cross checking* strategies allow to validate the correctness and coherence of the obtained models.

Fig. 7 shows another example of the second level map, 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.

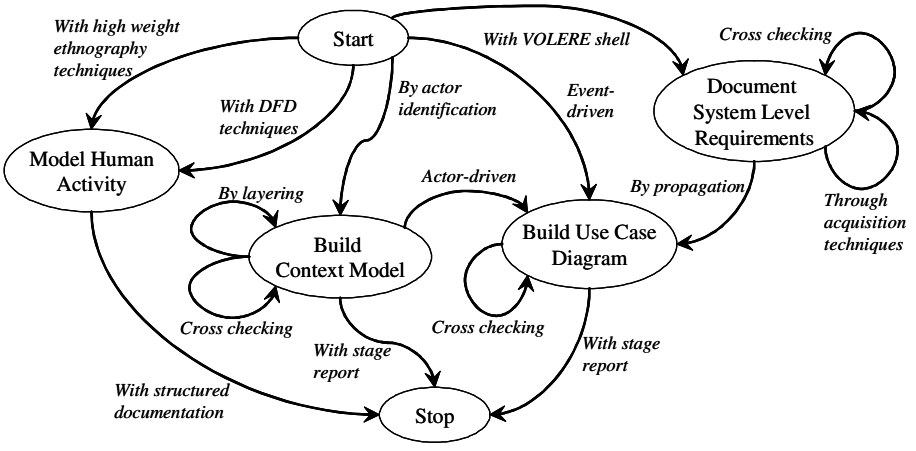

Fig. 6. Second level RESCUE map: the IAG associated to the section <*Start, Agree on System Boundaries, Multi-perspective modelling strategy*>

Whilst the first level map represents a more or less linear process, the second level maps are richer and often provide several strategies to achieve each intention. The progression guidelines are important in the second level maps. Given that, each map describes multiple manners, or ways, to achieve an intention, it needs to provide as much guidance as possible for selecting the right intention for each situation in RESCUE. An SSG provides this guidance for each set of parallel sections, whilst an ISG has to help the selection of the next intention to attain.

The definition of selection arguments plays an important role in strategies selection. In order to better define strategy selection arguments we propose a set of predefined attributes such as time, amount of resources, required domain knowledge, user involvement, difficulty of management, etc., that are specialised according to the nature of the strategies to compare. These attributes allow us to evaluate different aspects of the corresponding strategies and to compare them. Table 1 illustrates the comparison of four strategies allowing to attain the intention *Discover Requirements* from the intention *Produce Agreed Scenario* (Fig. 7).

Table 1. Comparison of four strategies to achieve the intention *Discover Requirements* from the intention *Produce Agreed Scenario*

Strategy selection attributes	Distributed workshop	Individual walkthrough	Facilitated workshop	Mobile walkthrough
Elapsed time to discover requirements/per scenario	5	0.5	0.5	0.3
Amount of analyst resource needed	0	0	2	1
Level of domain knowledge required	High	High	Low	Medium
Level of user involvement needed	High	High	High	Low
Level of management commitment needed	Low	Medium	High	Low
Capability to handle complex systems	Medium	Low	High	Medium
Capability to handle innovative systems	Medium	Low	High	Low
Capability to handle unstable requirements	High	High	High	High
Requirements discovery rate/hour	<8	<8	8-10	8-10
Number of VOLERE attributes discovered	5	5	5	5

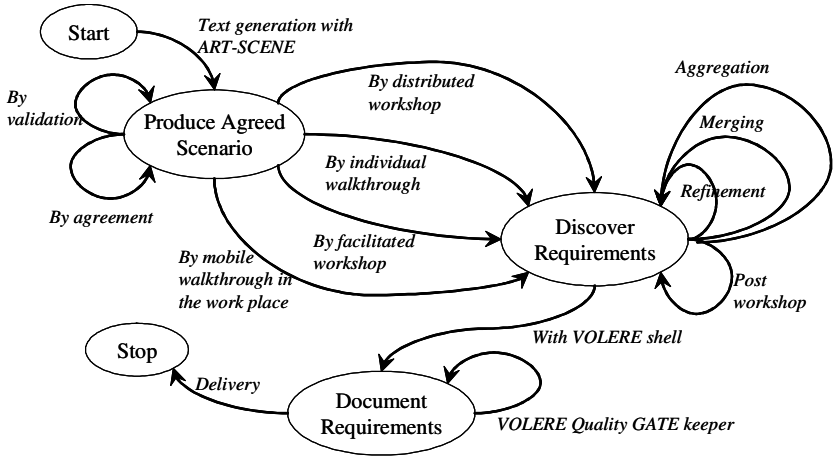

Fig. 7. IAG associated to the section <*Specify Use Cases, Specify Complete Requirements, With generated scenario walkthrough*>

5 Validation and Extension of RESCUE

In order to validate and extend the RESCUE process we explored all its second level maps, including some not shown in this paper. In particular we sought to overcome the weakness of single strategy intentions of RESCUE map and create and develop new strategies to achieve intentions that only have one strategy in the current version

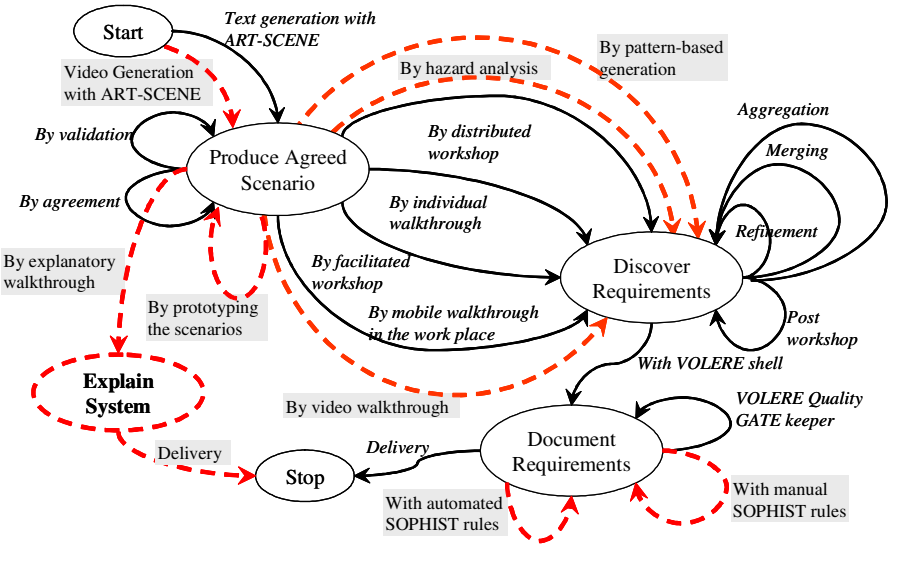

Fig. 8. The guideline IAG <*(Use case specifications), Specify Complete Requirements with generated scenario walkthrough* > enhanced with new sections

of RESCUE. For example, we considered possible new strategies to enhance the complete requirements specification process captured in the map of Fig. 7.

Fig. 8 shows the new process map for requirements specification with new sections and strategies shown in dashed edges. By systematically reviewing and walking through the process map, we were able to consider each intention in turn, and brainstorm new strategies for each intention. As a consequence, 9 new strategies and one new intention were identified and modelled. The new intention, *Explain System*, was generated and 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. Hence the new intention and an associated new strategy, *By explanatory walkthrough*, added to the process map. An explanatory walkthrough exploits the narrative structure of a scenario to describe and explain the future system's behaviour, and other types of requirement linked to that behaviour. It is an important component of a requirements review or read through activity.

The remainder of this section is two parts. The first outlines new strategies added to the *Complete Requirements Specification* map as a result of the process modelling exercise. The second describes strategy selection arguments in tabular form to demonstrate how to select between these new strategies.

5.1 New Strategies for Specifying Complete Requirements

Two new strategies, *Video Generation with ART-SCENE* and *Discover Requirements, by Video Walkthrough* were designed to produce the agreed scenario and discover requirements. Currently ART-SCENE scenarios are text-based. A text-based use case specification is input to ART-SCENE to generate an interactive and structured scenario that describes normal and alternative course events in text form. However, recent extensions to ART-SCENE to support multi-media representation of scenarios [19] have revealed new opportunities for video-based scenario walkthroughs. Initial trials reveal that multi-media scenario representations provide more cues from which stakeholders can discover and document requirements [19]. Therefore, the use case specification will be extended with a video sequence that describes the normal behaviour of actors to achieve their goals, and use case normal course events are linked to episodes, such as *an air traffic controller communicating with a pilot*, in the digital video. The ART-SCENE algorithm will still be used to generate alternative courses for each normal course event that are now linked directly to one or more digital video episodes, thus producing an agreed scenario in a video form. To discover requirements, an enhanced version of ART-SCENE will enable stakeholders to control and play a digital video of the normal course behaviour. Then, during the playing of the video, stakeholders are prompted with alternative course questions in text form, such as what if the pilot misunderstands the air traffic controller, in response to which they can document new requirements using existing ART-SCENE functions. We hypothesise that richer scenario representations will lead to more complete requirements discovery.

The brainstorming session also surfaced 3 other strategies for discovering requirements. One strategy, *By pattern-based generation*, recalled earlier research

undertaken by the RESCUE team that is not currently implemented in ART-SCENE. Alexander's original ideas of a pattern [1] focus on the interactions between the physical form of the built environment and how this form inhibits or facilitates various sorts of individual and social behaviour in it. The emphasis is on the characteristics of the environment that might facilitate or inhibit action. A pattern captures the essentials of a 'good design' that maximises characteristics that facilitate desirable actions over those that inhibit these actions. Applied to socio-technical system design with ART-SCENE, a pattern must capture the essential elements of the software system (expressed as functional and non-functional requirements), and how the form of this system facilitates and inhibits desirable individual or social behaviour (expressed using the scenario). It captures good designs that have been shown to facilitate desirable behaviour expressed in the scenario [8].

Based on the discovery of this strategy, we will extend ART-SCENE with 2 types of pattern that guide the discovery and documentation of system requirements. Firstly, we will develop and implement patterns that describe classes of solutions, expressed as generic requirement statements, to classes of abnormal behaviour and state in the environment, expressed as alternative courses that instantiate these classes. Implementation of these patterns in ART-SCENE is tractable because scenario alternative courses are generated automatically using classes of abnormal behaviour and state. During a scenario walkthrough, the pattern is applied to recommend generic requirements statements that describe what a system shall do avoid or mitigate against the effects of a selected alternative course [15]. For example, *an expected event not occurring* can be handled by the system in different manners – *by re-requesting the event*, *by undertaking some default action*, or *by assuming that the event has taken place*.

Secondly, we will develop and implement socio-technical system design patterns that link sequences of events and actions that describe desirable future system use in the environment, expressed as scenario normal courses, to system requirements facilitate the desirable and inhibit undesirable behaviour. Consider the *collect-first-objective-last* pattern reported in [8]. A person who interacts with a system using a personal item should not leave the personal item behind. One design to achieve this is to make the user reclaim the item before achieving their goal. This design can be found in ATMs, metro barriers and secure access systems, and can be specified computationally as a pattern to match in a scenario normal course. Again, we hypothesise that implementing these 2 strategies for *By pattern-based generation* will lead to the discovery of more complete and correct requirements.

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. ART-SCENE's automatic generation of scenario alternative courses can also identify potential hazards associated with a specified 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 with which to document requirements. Goetz & Rupp [3] report 25 authoring

rules from psychotherapy that assist in the analysis and quality assurance of requirements expressed in text form. Examples of these rules include *(6) Clarify the modal operators of imperative (e.g. the use of should, shall, must etc)* and *(12) Question nouns without references (e.g. reference to all users, or just certain user groups or individuals)*. In RESCUE we can supplement its use of the VOLERE requirements shell [13] with manual and automatic application of these 25 rules. The manual strategy is now implemented through engineer training and guidelines in ART-SCENE that advice on how to describe textual requirements. The automatic strategy will be implemented using a new tool that will parse and invite re-writes of the entered requirements specification to check each requirement against each of the 25 requirements authoring rules. Again, we hypothesise that these 2 strategies will result in more correct and consistent documentation of requirements.

5.2 Strategy Selection

Adding new strategies enriches RESCUE but also makes it more difficult to implement. Additional selection guidelines are needed to combine and/or select between strategies to achieve one intention. To guide selection we have developed new strategy comparison tables that define the predicted cost and benefit of adopting one strategy over another according to strategy selection attributes. Table 2 compares 3 of the defined strategies for achieving the intention *Discover Requirements* from the intention *Product Agreed Scenario*.

Table 2. Comparison of three new strategies to achieve the intention *Discover Requirements* from the intention *Produce Agreed Scenario*

Strategy selection attributes	Pattern-based generation	Hazard analysis	Video walkthrough
Elapsed time to discover requirements/per scenario	0	0.5	0.5
Amount of analyst resource needed	0 (min)	2	0
Level of domain knowledge required	Low	High	Medium
Level of user involvement needed	None	Low	High
Level of management commitment needed	Low	Medium	High
Capability to handle complex systems	Low	Medium	Medium
Capability to handle innovative systems	Low	Low	Low
Capability to handle dependencies on other system (or inter-system dependencies)	Low	Low	N/A
Capability to handle unstable requirements	Low	N/A	High
Requirements discovery rate/hour	Unknown	N/A	<12
Number of VOLERE attributes discovered/requirement	2	N/A	5
Use case action specification rate/hour	High		Low

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. Moreover, thanks to the MAP formalism the RESCUE process was transformed into a modular method: each RESCUE map section represents a more or less autonomous process module. These modules can be combined in different manners and reused in the construction of situation-specific requirements engineering processes in order to meet the needs of client organisations.

The next stage of our collaboration will model RUP's requirement processes [6] as a basis for integrating RESCUE into RUP. Once RUP process maps have been developed, we will merge intentions shared by RUP and RESCUE, add RESCUE intentions to RUP process maps, and introduce RESCUE strategies for achieving these shared intentions.

References

1. Alexander, C. (1979), 'The Timeless Way of Building', New York: Oxford University Press.
2. Baddeley, A.D. (1990), 'Human memory: Theory and practice', Lawrence Erlbaum Associates, Hove.
3. Goetz, R. & Rupp, C. (2003), 'Psychotherapy for Systems Requirements', Proceedings 2nd IEEE International Conference on Cognitive Informatics, IEEE CS Press, p. 75-80.
4. Hammond, J., Rawlings, R. & Hall, A. (2001), 'Will It Work?', Proceedings 5th IEEE International Symposium Requirements Engineering, IEEE CS Press, p. 102-109.
5. Jarke, M., Rolland, C., Sutcliffe, A. & Domges, R. (1999), 'The NATURE requirements Engineering', Shaker Verlag, Aachen.
6. Leffingwell, D. & Widrig, D. (2000), 'Managing Software Requirements: A Unified Approach', Addison-Wesley-Longman.
7. Maiden, N.A.M. & Rugg, G. (1996), 'ACRE: Selecting Methods For Requirements Acquisition', Software Engineering Journal 11(3), p. 183-192.
8. Maiden, N.A.M., Cisse, M., Perez, H. & Manuel, D. (1998), 'CREWS Validation Frames: Patterns for Validating System Requirements'', Proceedings REFSQ98 Workshop.
9. 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.
10. 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.
11. Mavin, A. & Maiden, N.A.M. (2003), 'Determining Socio-Technical Systems Requirements: Experiences with Generating and Walking Through Scenarios'', Proceedings 11th International Conference on Requirements Engineering, IEEE CS Press, p. 213-222.

12. Ralyté, J. & Rolland, C. (2001), 'An Approach for Method Re-engineering'. Proceedings of the 20th International Conference on Conceptual Modeling (ER2001), LNCS 2224, Springer, p. 471-484.
13. Robertson, S. & Robertson, J. (1999), 'Mastering the Requirements Process', Addison-Wesley-Longman.
14. Rolland, C., Prakash, N. & Benjamen, A. (1999), 'A multi-model view of process modelling'. Requirements Engineering Journal, p. 169-187.
15. 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.
16. van Lamsweerde, A. (2004), 'Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice', Proceedings 12th IEEE International Conference on Requirements Engineering, IEEE CS Press, p. 4-7.
17. Vicente, K. (1999), 'Cognitive work analysis', Lawrence Erlbaum Associates.
18. Yu, E. & Mylopoulos, J.M. (1994), 'Understanding "Why" in Software Process Modelling, Analysis and Design', Proceedings, 16th International Conference on Software Engineering, IEEE CS Press, p. 159-168.
19. Zachos, K. & Maiden, N.A.M. (2004), 'ART-SCENE: Enhancing Scenario Walkthroughs with Multi-Media Scenarios', Proceedings 12th IEEE International Conference on Requirements Engineering, IEEE CS Press, p. 360-361.