Paper submitted to ATM2003

Innovative Requirements Engineering Applied to ATM

Neil Maiden, Sara Jones & Mary Flynn*

Centre for HCI Design, City University, London
*Eurocontrol Experimental Centre, Paris

Abstract

RESCUE is an innovative and multi-disciplinary process for specifying requirements for the complex socio-technical systems found in ATM. It supports 4 concurrent process streams – human activity modeling, system modeling, scenario-driven walkthroughs and managing requirements. RESCUE integrates these streams using synchronisation stages to cross check models from each stream, ensure that these models are shared between the 4 streams, and support decision-making about system requirements and designs. It draws on established research in different disciplines to support the 4 streams. RESCUE was applied successfully to the Eurocontrol ASA programme's CORA-2 project to produce an operational requirements document. Lessons learned have led to improvements reported in this paper as the basis for further applications of RESCUE in the ASA programme.

1. Introduction

In spite of recent advances in software engineering we still lack systematic and scalable requirements engineering processes for socio-technical systems such as air traffic management (ATM). The need for such processes is increasing as computer systems for managing air traffic become more complex. One problem is that established requirements techniques have emerged from single disciplines – use cases from software engineering and task analysis from human-computer interaction are two obvious examples. Safety-critical socio-technical systems such as ATM demand rigorous analyses of controller work, software systems that support this controller work, and the complex interactions between the controllers, the air traffic and the software systems. To do this we need new hybrid processes that integrate best-practices from the relevant disciplines. This paper presents one such process, RESCUE, and describes its application to a large and computerised ATM system project.

Academic researchers worked with Eurocontrol's ASA programme to design and implement an innovative process to determine stakeholder requirements for CORA-2 (Conflict Resolution Assistant), a system that will provide computerised assistance to air traffic controllers to resolve potential conflicts between aircraft. CORA-2 is a complex socio-technical system in which controllers and computers depend on each other to bring about the effective functioning of the system. Although our main foci in the CORA-2 process are stakeholder requirements, these requirements could not be reified without understanding candidate socio-technical designs for CORA-2. Our challenge, therefore, was to design and implement a systematic requirement engineering process that took into account both important human factors analysis and creative design ideas.

The remainder of the paper is in 5 sections. The next section reviews related work. Section 3 describes the RESCUE process and its 4 main process streams. Section 4 describes RESCUE's 5 key synchronisation stages that cross check and integrate the 4 process streams. Section 5 reports the application of the first version of RESCUE to Eurocontrol's CORA-2 project, and summarises key lessons learned. The paper ends with plans to further improve the RESCUE process for future ASA programme projects.

2. Related Work

RESCUE's roots in the software and systems engineering, human factors and cognitive creativity disciplines mean that most related work cannot be reported in this paper. Instead we review research applied to ATM in the areas of development methodologies, function allocation, ethnography, requirements engineering, creativity, and integrated system and use case modeling research.

Leveson et al. (2000) describe a safety and human-centred approach to developing ATM tools that integrates human factors and systems engineering work. Although similar in spirit to RESCUE, their approach includes safety hazard analysis and verification that were outside RESCUE's scope, and covers the full development cycle. In contrast RESCUE focuses its support on one systems engineering process – generate system and operational requirements – and provides more prescriptive guidance based on innovative tools and techniques.

Wright et al. (2000) address the relationship between work in systems engineering, human-computer interaction and computer-supported

collaborative working with reference to case studies in aircraft design. We share their concern that traditional systems engineering approaches to function allocation (determining which system-level functions should be carried out by humans, and which by machines) may lead to designs which are less than ideal. This is due, in part, to a lack of understanding about the human component and, in particular, about the context in which it takes place. In RESCUE we seek to address this issue by the integration of human activity modelling into the process as a whole.

Randall et al. (1994) report the use of ethnographic techniques in investigating electronic alternatives to paper flight strips. Ethnographers spent several months observing the work of air traffic controllers with the aim of informing the specification of requirements for a replacement system. Although the RESCUE process leaves room for the application of ethnographic techniques, full-scale ethnographic studies are often beyond the scope of most ASA programme projects.

There is also increasing interest in creativity research in safety-critical systems. In the ATM domain Marti & Moderino (2002) report creative design undertaken by musicians and artists to rethink controller work practices through the use of portable devices. The work was both exploratory and unusual because stakeholders did not have direct input into the creative design process. This contrasts with RESCUE's user-centred approach that integrates stakeholder involvement in creativity workshops into the wider systems engineering process.

Beyond ATM applications, researchers have integrated the *i** goal modeling approach implemented in RESCUE with use case approaches, similar to RESCUE. Santander & Castro (2002) present guidelines for automatically deriving use case models from *i** system models, and Liu & Yu (2001) integrate goal modeling with the GRL with use case maps to refine scenarios into architectural designs with goal-based rationale. Our work in RESCUE is similar to the latter work but exploits *i** models to scope use case models and inform scenario walkthroughs rather than derive architectures per se.

3. The RESCUE Process

The RESCUE (Requirements Engineering with Scenarios for User-Centred Engineering) process was developed by multi-disciplinary researchers at the Centre from HCI Design at City University in London. It integrates applied research and best-practice in:

- Human activity modeling (Vicente 1999)
- Scenario-based requirements engineering using the CREWS-SAVRE approach (Sutcliffe et al. 1998);
- System modeling with *i** (Chung et al. 2000),
- Requirements management based on the VOLERE requirement shell (Robertson & Robertson 1999);
- Requirements acquisition using the ACRE framework (Maiden & Rugg 1996);
- Innovative techniques for creative requirements engineering (Maiden & Gizikis 2001).

RESCUE supports a concurrent engineering process in which different modeling and analysis processes take place in parallel. Furthermore the use of creativity workshops encourages requirements and design ideas to be discovered and elaborated together, so that requirements inform high-level design selection, and candidate acceptable designs restrict and constrain requirements to those that are viable. The concurrent processes are structured into 4 streams shown in Figure 1.

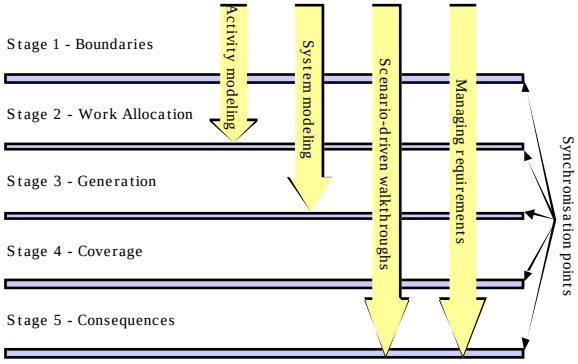

Figure 1. The RESCUE process structure – activity modeling ends at stage 2, system modeling at stage 3, and scenario-driven walkthroughs and modeling requirements at stage 5.

Each stream has a unique and specific purpose in the specification of a socio-technical system:

- Human activity modeling provides an understanding of how people work, in order to baseline possible changes to it;
- System modeling enables the team to model the future system boundaries, actor dependencies and most important system goals;
- Scenario-driven walkthroughs enable the team to communicate more effectively with stakeholders and acquire complete, precise and testable requirements from them;
- Managing requirements 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.

These 4 streams are supplemented with 2 additional processes. Acquiring requirements from stakeholders is guided using ACRE (Maiden & Rugg 1996), a framework for selecting the right acquisition techniques in different situations. Creativity workshops (Maiden & Gizikis 2001) are ran at the beginning of the process to discover and surface requirements and design ideas that are essential for

system modeling and use case authoring later in the RESCUE streams.

Sub-processes during these 4 streams are co-ordinated using 5 synchronisation stages that provide the project team with different perspectives with which to analyse system boundaries, goals and scenarios.

The next sections describe each of the 4 streams in more detail.

3.1. Human Activity Modeling

In this RESCUE stream the project team develops an understanding of the current socio-technical system to inform specification of a future system. Activity modeling focuses on the human users of the technical system, in line with the principle of human-centred automation (ICAO 1994). To do this the project team must first understand the controllers' current work – its individual cognitive and non-cognitive components and social and co-operative elements - to specify the technical systems that can better support that work. Introducing artefacts, tools or procedures into the work domain changes the way in which people work and process information. It also brings about changes in the cooperative, and possibly organisational structures that are related to the new system. RESCUE guides the project team to generate requirements, design visions and automation boundaries for a new system that reflects an understanding of how new artefacts might potentially change existing work practices.

The stream consists of two sub-processes – gathering data about and modeling the human activity.

During the first sub-process, data about all components of the activity model are gathered and recorded, initially in a relatively unstructured way. Techniques to gather this data include:

- Observation of current system use;
- Informal scenario walkthroughs, using scenarios that describe how the current system is used;
- Interviews with representative human users;
- Analysis of verbal protocols, or recordings of users talking through scenarios or tasks;
- Ethnographic techniques.

A combination of different techniques is usually needed to elicit different kinds of data. Data gathered using one technique may be needed to confirm inferences made on the basis of data from another technique.

The project team must keep an open mind and not pre-judge the relevance of particular information. The inherent non-determinism of complex and open socio-technical systems such as those used in ATM means that almost any data may be relevant in some way. In particular the team is encouraged to record data about what are called the 'non-prescribed' elements of the current system. Non-prescribed elements are those elements – goals, actors, actions etc – which the human users of the system have developed to 'finish the design', often to work around difficulties with the current system or increase its flexibility, thus also increasing redundancy and hence safety. They will, by definition, not be described in manuals or codes of practice, and may often not be made explicit, even by those who use them. It is the team's job to infer, then confirm, the existence of such non-prescribed system components through observation, often of less obvious actions, such as non-direct communication like watching and listening.

This stream also focuses on describing different, equally valid ways by which to achieve the same system goals, and how actors behave differently on different occasions. Both inter- and intra-controller differences in performance may be due to different trade-off criteria, attitudes to risk, trust in a controller's own knowledge, trust in other controllers, etc. The team must attempt to understand and record the way in which different factors appear to constrain and influence the behavior of human system users.

In the second sub-process the project team create different activity models corresponding to major types of activity in the current system. An activity model is a repository of information about various aspects of the current system. One key aspect of an activity model is goals - the desired states of the system. Goals may be: (i) high-level functional goals relating to the system as a whole, or local goals relating to particular tasks; (ii) individual goals, relating to single actors, or collective goals, relating to teams of actors; (iii) prescribed goals or non-prescribed goals. Other aspects to describe in a model include:

- Human actors - people involved in system;
- Resources – means that are available to actors to achieve their goals;
- Resource management strategies – how actors achieve their goals with the resources available;
- Constraints - environmental properties which affect decisions;
- Actions - undertaken by actors to solve problems or achieve goals; and
- Contextual features – situational factors that influence decision-making.

Data gathered in the first sub-process is recorded under each of these headings as natural language statements. No attempt is made at this stage to impose a more formal modeling paradigm on the activity model, as enforcing a more detailed conceptual framework would constrain the usefulness of the data.

The resulting activity models provide direct inputs to the 3 other RESCUE streams. The models provide important sources of data for *i** modeling in the system modeling and use case authoring in scenario-driven walkthrough streams, and data for fit criteria for system requirements. Furthermore the team obtain a better understanding of the work and application domains, which is essential for effective requirements acquisition.

3.2. System Modeling

In this RESCUE stream the project team models the future system's actors (humans and otherwise), dependencies between these actors and how these actors achieve their goals, in order to explore the boundaries, architecture and most important goals of the socio-technical system. RESCUE adopts the established *i** approach (Chung et al. 2000) but extends it to model complex technical and social systems, establish different types of system boundaries, and derive requirements. *i** is an approach originally developed to model information systems composed of heterogeneous actors with different, often-competing goals that nonetheless depend on each other to undertake their tasks and achieve these goals – like the complex socio-technical systems found in ATM.

The system modeling stream requires 3 analyses to produce 3 models. The first is a context diagram, similar to the REVEAL process (Praxis 2001) but extended to show different candidate boundaries for:

- The technological systems, expressed in terms of software and hardware actors;
- The redesigned work system, expressed primarily in terms of human actors;
- Other hardware and software systems directly influenced by the redesign of the new system;
- Systems that interact with the new system but are not influenced by its redesign.

To help determine these system boundaries we also adopt the notion of different types of adjacent actors (Robertson & Robertson 1999). The result is an extended context model with typed actors that provides a starting point for *i** system modeling. A simple context model for the CORA-2 system is shown in Figure 2.

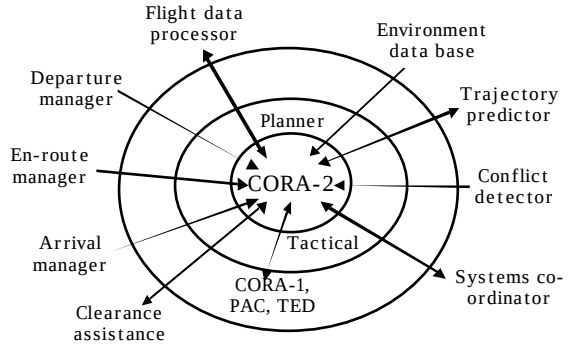

Figure 2. A Simple Context Model for CORA-2

The second model is the *i** Strategic Dependency (SD) model, which describes a network of dependency relationships among actors identified in the context model. The opportunities available to these actors can be explored by matching the depender who is the actor who "wants" and the dependee who has the "ability". Since the dependee's abilities can match the depender's requests, the system-wide strategic model is developed. A depender can depend upon a dependee to achieve a goal, undertake a task, obtain or use a resource, and achieve a soft goal in a particular way. Figure 3 shows part of the SD model for the CORA-2 system.

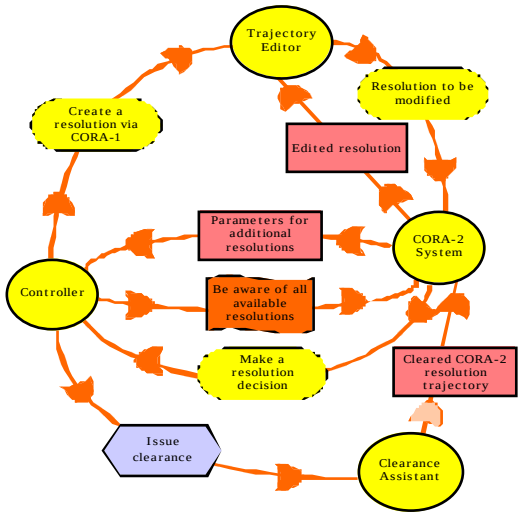

Figure 3. Part of the SD Model for CORA-2. Actors, shown as circles, are linked using dependencies of different types. The half-circles denote the direction of the dependency as the letter "D" is read. For example, the CORA-2 system depends on the controller for parameters to have the resource parameters for additional resolutions.

RESCUE provokes the team to ask important questions about systems boundaries by re-expressing them in terms of the goal dependencies between actors on either side of a boundary. Actors with goals that the team will seek to test for compliance are, by definition, part of the new system. Such re-expression also leads to more effective requirements specification by referring to named actors that will be tested for compliance (e.g. "The *controller* using CORA-2 shall be aware of all resolutions", and "The *clearance assistant* shall provide cleared CORA-2 resolution trajectories to CORA-2"). It also suggests a first-cut architecture and functional allocation for the socio-technical system by defining which actors undertake which tasks.

The second type of *i** model is the Strategic Rationale (SR) model, which provides an intentional description of how each actor achieves its goals and soft goals. An element is included in the SR model only if it is considered important enough to affect the achievement of some goal. The SR model includes the SD model, so it describes which actors may be able to

accomplish something by themselves, or by depending on other actors.

The SR model includes goals, tasks, resources and soft goals linked by dependency links from the SD model, task decomposition links, means-end links, and the contributes-to-soft goal links (Chung et al. 2000). These simple link-types offer considerable power of expression when modelling how actors both conflict with and depend on each other to achieve their goals.

We have developed REDEPEND to support *i** modeling within RESCUE. REDEPEND is a graphical modeling tool developed as a plug-in to Microsoft Visio 2000 that enables the team to construct and analyse *i** SD and SR models.

This stream provides key inputs to the managing requirements and scenario-driven walkthroughs. Goals and soft goals in *i** SR models become requirements in the managing requirements stream. Context and *i** models define the system boundaries essential for use case modeling and authoring. The *i** SR models define goal and task structures that suggest skeletal use case descriptions to refine the scenario-driven walkthroughs stream.

3.3. Scenario-driven Walkthroughs

In this RESCUE stream the team writes use cases then generates and walks through rich scenarios to discover and acquire stakeholder requirements that are complete, precise and testable. We have applied results from the EU-funded CREWS project (Sutcliffe et al. 1998) to extend the Rational Unified Process with increased method guidance for use case authoring, innovative software tools for scenario generation and walkthroughs, and rich traceability to link and contextualise requirements in scenarios. There are 5 sub-processes.

The first sub-process is use case modeling (Jacobson et al. 2000) that we have extended to model and investigate different system boundaries identified in the context model. The outcome is a use case model with use cases and short descriptions that are inputs into use case authoring.

In the second sub-process the team writes detailed use case descriptions using the structured templates derived from use case best-practice (e.g. Cockburn 2000) shown in Figure 4. Authoring is guided using use case style and content guidelines from the CREWS-ECRITOIRE method (Ben Achour et al. 1999), temporal semantics expressed as action-ordering rules, and an extensive lexicon of ATM nouns and verbs elicited from controllers. To write each description the team draw on outputs from the other streams – activity models, *i** strategic rationale models, stakeholder requirements, and innovative design ideas from the creativity workshops. Once each use case description is complete and agreed with the relevant stakeholders, the team produce a use case specification from it, and parameterise it to generate scenarios automatically from each description.

	Name of Use Case
Use Case ID	Unique ID for Use Case
Author	Name of author
Date	Date Use Case was written
Source	Source of Use Case
Actors	Actors involved in Use Case (from the Use Case Model)
Problem statement (now)	Description of current problem
Precis	Informal scenario description
Functional Requirements	Requirement that the use case **DECOMPOSES**
Non-functional Requirements	Requirements that impose **CONSTRAINS** on the required behaviour in the use case
Added Value	Benefit of Use Case above and beyond the original scenario from the original system
Justification	Why is the Use Case needed?
Triggering event	Event or events that can trigger the Use Case
Preconditions	Necessary conditions for the Use Case to occur
Assumptions	Explicit statement of any assumptions made in writing the Use Case
Successful end states	Successful outcome(s) of the Use Case
Unsuccessful end states	Unsuccessful outcome(s) of the Use Case
Normal Course	1. Action 1 System requirement **ENABLES** Action 1 System requirement **CONSTRAINS** Action 1
	2. Action 2 System requirements **ENABLES** to Action 2 System requirement **CONSTRAINS** Action 2
Variations	1. If [condition] then [variation statement] (related to Action 1)
Alternatives	1. If [condition] then [alternative course statement] (related to Action 1)

Figure 4. The RESCUE use case template

The next sub-process automatically generates scenarios using the CREWS-SAVRE software tool (Sutcliffe et al. 1998). We use CREWS-SAVRE to improve productivity by generating scenarios automatically and providing richer scenarios with wider coverage than most human authors can produce in the same time. CREWS-SAVRE generates one or more scenarios from each use case specification. Firstly the generation algorithm generates possible different normal course scenarios from the action ordering rules and generation parameters in the use case specification. Each different possible ordering of normal course events is a different scenario. Secondly, the algorithm generates different possible alternative courses for each normal course event. It again uses the use case parameters to query a database of candidate abnormal behaviours and states, and applies 17 basic rules to generate alternative course 'what-if' questions for each normal course event.

The queried database is in 3 parts. The first contains 35 classes of abnormal behaviour and state derived from scenario semantics, for example events not occurring, actions not completing and objects failing. The second part contains over 50 classes of abnormal behaviour and state linked to simple types of actions (e.g. cognitive, physical, communication, etc), agents (human, machine, etc) and environmental factors (noise, lighting, etc) defined in CREWS-SAVRE. If an action involves a human-type agent, then alternative

courses are generated from classes of human error drawn from cognitive science and human factors research (e.g. *what if the controller has insufficient knowledge to undertake the action?*). If an action involves a machine-type agent, then alternative courses are generated from classes of machine failure (e.g. *What if the power to the CORA-2 system fails?*). If the communication action involves at least one human-type agent and one machine-type agent, then alternative courses are generated from interaction failures in human-computer interaction (e.g. *What if the controller does not perceive CORA-2 system changes?*). If the action involves two machine-type agents, then alternative courses are generated from machine networking failures (e.g. *What if the connection between the CORA-2 and CORA-1 systems fail?*). Likewise, if it involves two or more human-type agents, then alternative courses are generated from models of communication failure (e.g. *What if the tactical controller misinterprets the gesture from the planner controller?*). Other classes define exceptions about information-models and work-domains.

The third part of the database includes 138 sub-classes of abnormal behaviour and state specific to the ATM domain. This ATM domain knowledge was again elicited from controllers, modeled using the UML, and extended with rules to generate alternative courses from the ATM lexicon used to define normal course actions in the use case description. A simple example from the model, describing abnormal aircraft behaviour, is shown in Figure 5. Possible generated alternative courses include *What if the aircraft shows unexpectedly low performance?* and *What if the aircraft is a high-performance military aircraft?* for normal course events that describe the movement of aircraft.

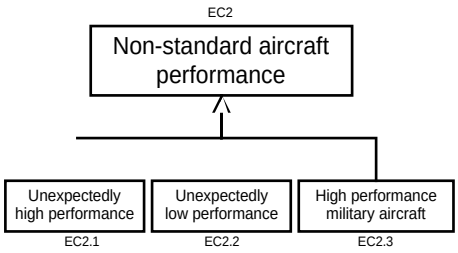

Figure 5. Part of the ATM abnormal behaviour model, showing sub-classes of abnormal aircraft behaviour

The fourth sub-process, pivotal to RESCUE, is to walk through each generated scenario with stakeholders using bespoke software tool support. Each scenario is delivered for walkthrough in two forms – either as an interactive MicroSoft Excel spreadsheet that can downloaded by the session facilitator – or through the web-based Scenario Presenter tool shown in Figure 6.

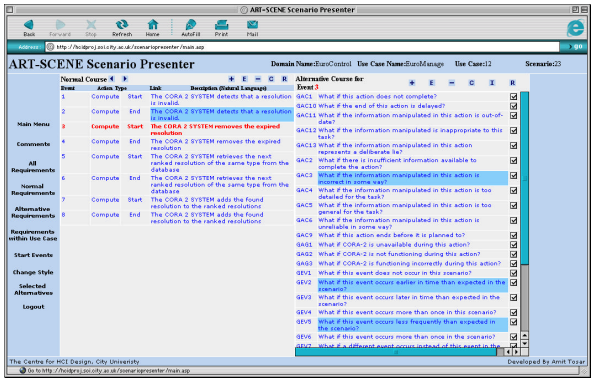

Figure 6. The Scenario Presenter Tool, available from www.soi.city.ac.uk/artscene

Each scenario is in 4 parts. The left-side menu provides different functions for viewing the scenario and comments and requirements generated from it. The top-line button offers walkthrough functions (e.g. next or previous event, add/edit/delete event) and functions to add, edit or delete comments and requirements. The left-hand main section describes the normal course event sequence for the scenario. Each event describes the start or end of an action, thus enabling a scenario to describe concurrent actions in this text-list form. The right-hand main section describes generated alternative courses for each normal course event, presented in the form of 'what-if' questions. Different alternative courses are presented for different normal course events. Facilitators walk through the scenario with stakeholders, guided by the Scenario Presenter, to consider each normal course event and each alternative course linked to that normal course event in turn. For each event, the facilitator guides stakeholders to consider whether:

1. This event might occur?
2. This event is relevant to the future system?
3. Does the future system, as currently specified in the requirement document, handle the event?

If no requirements are specified to handle a relevant event then omissions have been discovered, and one more new requirements can be written. A human scribe uses the tool to document all requirements and comments arising from the walk through directly into the Scenario Presenter.

The final sub-process extends these scenarios in order to undertake simple impact analyses on the specified system by investigating how the system, as specified, will impact key environmental factors such as job security, actor roles and responsibilities, and access to information.

The main outcome of this stream is a set of more complete requirements that can be traced to the originating scenario, and hence use cases, and specified in context to remove ambiguity and make them more testable.

3.4. Managing Requirements

In this fourth RESCUE stream the project team documents, manages and analyses requirements generated from the other 3 streams – automation and process requirements emerging from human activity modeling, system actor goals and soft goals from *i** system modeling, and requirements arising from scenario walkthroughs with the Scenario Presenter.

Each requirement is documented using the VOLERE shell (Robertson & Robertson 1999), a requirement-attribute structure that guides the team to make each requirement testable according to its type. Use cases and scenarios are essential to making requirements testable. Each new requirement is specified either for the whole system, one or more use cases of that system, or one or more actions in a use case. This RESCUE requirement structure links requirements to and places them in use cases and use case actions "in context", thus making it much easier to a write a measurable fit criterion for each requirement.

The use case-driven requirement structure carries over into the operational requirement document itself, to improve both the readability of the document and the understandability of each requirement statement. The document was divided into a series of use case descriptions using the RESCUE use case template, with requirement statements inserted into normal and alternative course descriptions next to the relevant use case and use case actions, as shown in Figure 7.

1. The CORA-2 system ranks the resolutions.
Rank-Res1: The CORA-2 system shall rank resolutions and their cost values.
Rank-Res2: The CORA-2 system shall rank resolutions based on the cost-value calculated for each resolution.
2. The CORA-2 system retains the ranked resolutions
RankRes5: The CORA-2 system shall retain a record of the ranked resolutions.

Figure 7. Fragment of the CORA-2 Requirements Document, showing 2 use case actions and 3 linked requirements

RESCUE requirements are documented using Rational's Requisite Pro. Outputs from other streams, such as use case, context and *i** models, are also included in the document. We propose workarounds with Requisite Pro to support the complex use case and requirements structures within RESCUE and provide the essential rich traceability for each requirement. Where relevant, one requirement can be linked to one alternative course of one normal course event of one scenario generated from one use case description in the document using the automatic tracing feature in the Scenario Presenter tool.

Finally the team applies the VOLERE Quality Gateway (Robertson & Robertson 1999) to all requirements to be entered into the document. One member of the team is allocated to play the Gatekeeper role, asking a number of questions of each requirement to ensure that only complete and correct requirements enter the document. Questions seek to establish whether the requirement is testable, viable, solution-independent, and other value to stakeholders.

4. RESCUE's 5 Synchronisation Stages

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

- The **boundaries** point, where the team establishes first-cut system boundaries and undertakes creative thinking to investigate these boundaries;
- The **work allocation** point, where the team allocate functions between actors according to boundaries, and describe interaction and dependencies between these actors;
- The **generation** point, where required actor goals, tasks and resources are elaborated and modeled, and scenarios are generated;
- The **coverage** point, where stakeholders have walked through scenarios discover and express all requirements so that they are testable;
- 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.

Each synchronisation stage is described in more detail.

4.1. The Boundaries Synchronisation Point

Key inputs to this first synchronisation point are data about human activities, the context and use case models and use case summaries, system-level requirements and new design ideas from creativity workshops. RESCUE defines numerous guidelines for cross checking models similar to those reported in Santander & Castro (2002), for example:

- Every adjacent actor in the context model should be a use case actor in the use case model;
- Every major human activity should correspond to one or more use cases in the use case model.

The culmination of the synchronisation point is a workshop in which major stakeholders agree the system boundaries, reject requirements that are not viable, agree viable design ideas that satisfy requirements, and make some first-cut design choices regarding work practices, the extent of automation, and interaction modes.

4.2. The Work Allocation Synchronisation Point

This synchronisation point cross checks the human activity models, first-cut *i** SD and SR models, use case descriptions and emerging stakeholder requirements, again using simple-to-apply guidelines, for example:

- Each task undertaken by a socio-technical system actor modeled in the *i** SD and SR models should correspond to one or more actions in a use case description;
- Each *i** goal and soft goal should be described as a requirement using the VOLERE shell.

The purpose is to agree the allocation of work between the actors in the socio-technical system as expressed in the *i** models and use case descriptions. To achieve this the modeling should be incremental and involve key stakeholder representatives throughout. Again a workshop is held to walk through and sign off each use case description and its related *i** models.

4.3. The Generation Synchronisation Point

At the third synchronisation point the team analyses the final *i** SR models to validate actor goals, soft goals, tasks and resources, trade-offs between the satisfaction of different soft goals, and dependencies between actors to achieve goals and soft goals. CREWS-SAVRE is also applied to generate scenarios that are tailored to examine critical soft goal trade-offs. Again each *i** goal and soft goal is checked to ensure that it is described as a requirement using the VOLERE shell.

4.4. The Coverage Synchronisation Point

The main input to this synchronisation point is a set of testable stakeholder requirements arising from the scenario walkthroughs. These walkthroughs can often generate duplicate and inconsistent requirements from different stakeholder groups. The team inspects the requirement document use case by use case to identify and remove duplicates and inconsistencies. It does this by cross checking key requirements with the *i** SR models that indicate possible conflicts using negative contributes-to soft goal links. Workshops are held to sign off each use case in the requirement document in turn.

4.5. The Consequences Synchronisation Point

The team checks that the consequences of all discovered impacts lead to acceptable changes to use cases and requirements in the requirements document. The team end the process by signing off the document ready for formal reviews external to the process.

5. The CORA-2 Case Study

An earlier version of the RESCUE process was applied to develop operational requirements for Eurocontrol's CORA-2 system. The CORA-2 project team consisted of one manager, 2 requirement engineers, 2 air traffic controllers who acted as domain experts, 1 human factors expert and 1 technical expert. The process was applied in earnest in Summer 2001 and delivered a reviewed and accepted document in May 2002.

The process was delivered to the project team as process documents, tutorials, software tools such as REDEPEND and the Scenario Presenter, and a Requisite Pro requirements document template. Process deliverables were periodically read by the RESCUE process authors to check for the correct application of the process and its techniques and tools.

The team produced an operational requirement document for the CORA-2 system with approximately 400 requirements structured using 22 use cases. The document also contained use case and *i** system models, and some completed VOLERE requirement attributes. A separate 75-page document of design ideas generated from 3 creativity workshops was also produced. Requirements were identified from focus group acquisition sessions, stakeholder interviews, *i** modeling and scenario walkthroughs. Key lessons learned from this application included:

- Ensure that project team members are conversant with all 4 process streams and techniques before the project starts, so that they understand how the streams synchronise together;
- Stakeholder involvement throughout the process is critical to its success. Good requirements practice (e.g. writing testable requirements) necessitates access to domain knowledge that is often only available from stakeholders;
- Scenario walkthroughs using our bespoke tools led to quicker acquisition of more requirements that were more precisely specified than did the use of focus acquisition sessions, interviews or system modeling;
- The *i** modeling technique is an effective technique for exploring the CORA-2 system boundaries by defining all of the other systems and human actors that CORA-2 depends on;
- Obtain agreement on and sign off all deliverables (requirements, use cases, models) throughout the process, otherwise extensive and costly reworking is needed and analysis work is undermined.

These and other lessons have led to substantial improvements grounded in empirical evidence to produce the new RESCUE version presented in this paper.

6. Future Work

At the time of writing we are using the revised RESCUE process presented in this paper to specify requirements for Eurocontrol's future D-MAN Departure Management system, in conjunction with NATS, the UK's National Air Traffic Service. The RESCUE process stream-and-stage structure has provided a simple but effective basis for planning the project. Much of the stakeholder involvement and team effort will be in sub-processes leading to the first 2 stages, to determine boundaries and work allocation for tasks that, thus far, have been largely manual. We look forward to reporting this work in the near future.

7. Acknowledgements

We wish to thank Eurocontrol's Experimental Centre and ASA programme and all of the participants for their valuable efforts in the workshops reported in this paper.

8. References

Ben Achour C., Rolland C., Maiden N.A.M. & Souveyet C., 1999, 'Natural Language Studies on Use Case Authoring', Proceedings 4th IEEE Symposium on Requirements Engineering, IEEE Computer Society Press, 36-43.

Chung L., Nixon B., Yu E. and Mylopoulos J., 2000, 'Non-Functional Requirements in Software Engineering', Kluwer Academic Publishers.

Cockburn A., 2000, 'Writing Effective Use Cases', Addison-Wesley Pearson Education.

Leveson N, de Villepin M., Srinivasan J., Daouk M., Neogi N., Bachelder E, Bellingham J., Pilon N. & Flynn G., 2001, 'A Safety and Human-Centred Approach to Developing New Air Traffic Management Tools', Proceedings Fourth USA/Europe Air Traffic Management R&D Seminar.

ICAO, 1994, 'Human Factors in CNS/ATM systems. The development of human-centred automation and advanced technology in future aviation systems' ICAO Circular 249-AN/149.

Jacobson I., Booch G. & Rumbaugh J., 2000, 'The Unified Software Development Process', Addison-Wesley-Longman.

Liu L. & Yu E., 2001, 'From Requirements to Architectural Design – Using Goals and Scenarios', Proceedings first STRAW workshop, 22-30.

Marti, P. & Moderini, C., 2002, 'Creative design in safety critical systems', Proceeding of the Eleventh European Conference of Cognitive Ergonomics, ECCE11, Catania, Italy,. September 2002, 159-166.

Maiden N. & Gizikis A., 2001, 'Where Do Requirements Come From?", IEEE Software September/October 2001 18(4), 10-12.

Maiden N.A.M. & Rugg G., 1996, 'ACRE: Selecting Methods For Requirements Acquisition, Software Engineering Journal 11(3), 183-192.

Praxis, 2001, 'REVEAL: A Keystone of Modern Systems Engineering', White Paper Reference S.P0544.19.1, Praxis Critical Systems Limited, July 2001.

Randall, D., Hughes, J. and Shapiro, D., 1994, 'Steps toward a partnership: Ethnography and system design', in Requirements Engineering: Social and Technical Issues, M. Jirotka and J. Goguen (eds), Academic Press, 241-258

Robertson S. & Robertson J., 1999, 'Mastering the Requirements Process', Addison-Wesley-Longman.

Santander V. & Castro J, 2002, 'Deriving Use Cases from Organisational Modeling', Proceedings IEEE Joint International Conference on Requirements Engineering (RE'02), IEEE Computer Society Press, 32-39.

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

Vicente, K., Cognitive work analysis, Lawrence Erlbaum Associates, 1999.

Wright P., Dearden A. & Fields R., 2000, 'Function Allocation: A Perspective from Studies of Work Practice', International Journal of Human-Computer Studies 52(2), 335-355.

9. Biographies

Neil Maiden is a Reader and Head of the Centre for HCI Design, an independent research department in City University's School of Informatics. He received a PhD in Computer Science from City University in 1992. He is and has been a principal and co-investigator of several EPSRC- and EU-funded research projects including SIMP, CREWS and BANKSEC. He is also founder and manager of City University's SAP R/3 Laboratory. His research interests include frameworks for requirements acquisition and negotiation, scenario-based systems development, component-based software engineering, ERP packages, requirements reuse and more effective transfer of academic research results into software engineering practice. Neil has over 80 journal and conference publications. He is also co-founder and treasurer of the British Computer Society Requirements Engineering Specialist Group. Centre details are available from www-hcid.soi.city.ac.uk.

Sara Jones is a senior member of the Research Staff in the Centre for HCI Design at City University in London, working on ongoing projects funded by Eurocontrol. She received her PhD in Computer Science from City University in 1991. She was previously a Principal Lecturer at the University of Hertfordshire in the UK. Sara had written numerous academic journal and conference papers in HCI, software engineering and requirements engineering.

Mary Flynn is project manager of the CORA-2 project at Eurocontrol in Bretigny near Paris, where she has worked for the last 17 years.