Abstract
This chapter describes RESCUE, a method for specifying requirements for complex socio-technical systems which integrates human activity modelling, creative design workshops, system goal modelling using the i* notation, systematic scenario walkthroughs and best practice in requirements management. This method has been, and is being applied in specifying requirements for three separate systems in the domain of air traffic control. In this chapter, we present examples showing how the method can be applied in the context of a case study involving the specification of requirements for Countdown, a system to provide bus passengers with information about expected bus arrival times. While sharing some important similarities with systems used in air traffic control, we hope that this system is small and familiar enough to readers to provide meaningful insights into the application of the RESCUE process.

Keywords: socio-technical approach, requirements definition, case study, use case model, IS models, interdisciplinary teams, air traffic management, scenarios.

INTRODUCTION

In spite of recent advances in software engineering we still lack systematic and scalable requirements engineering processes for complex socio-technical systems. The domain of particular interest in this chapter is that of air traffic control, in which human air traffic controllers and technical, software-intensive systems are both integral parts of what can be seen as a complex socio-technical system controlling the movement of air traffic. 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 those used in air traffic control 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.

The RESCUE (Requirements Engineering with Scenarios for a User-centred Environment) process has been developed to address this need in the domain of air traffic control.
researchers have worked with staff at Eurocontrol (the European Organisation for the Safety of Air Navigation) to design and implement an innovative process to determine stakeholder requirements, which is specifically targeted towards the needs of that domain. Thus RESCUE focuses on specification of requirements for critical systems, where development of new systems is evolutionary rather than revolutionary, and where the emphasis is on getting requirements right, rather than speed to market.

The RESCUE process was initially developed to specify requirements for a system called CORA-2 (‘CORA’ stands for Conflict Resolution Assistant). CORA-2 is a system that will provide computerised assistance to air traffic controllers to resolve potential conflicts between aircraft. CORA-2 is part of a complex socio-technical system in which controllers and computers depend on each other to bring about the desired effect of avoiding collisions between aircraft in the sky. The RESCUE process is now being applied in the specification of requirements for two further systems in the domain of air traffic control: DMAN (Departure Manager), which is a system to support controllers in managing the departure of aircraft from major European airports, and MSP (Multi-Sector Planning), which is for scheduling aircraft from gate to gate across multiple, multi-national sectors.

This chapter will provide a description of the RESCUE process, together with examples taken from the application of RESCUE to a small case study. We then provide a brief review of related literature. Each of the major components of the RESCUE process will be explained, with references to the literature on which it has been based. Examples of artefacts, or models, generated during the course of the process will be given with reference to a small case study. The chapter ends with a brief consideration of future trends, as well as some overall conclusions.

BACKGROUND

There has recently been a trend in requirements engineering towards combining techniques in order to complement the deficiencies of one with the strengths of another. For example, Leveson et al (2001) describe a safety and human-centred approach to developing ATM tools that integrates human factors and systems engineering work. Several authors have combined use cases with other techniques for this reason. For example Rolland et al (1998) explored the possibilities of linking goal modeling and scenario authoring, and Santander and Castro (2002) presented guidelines for using i* organisational models in the development of use cases. RESCUE is a process that continues this tradition and combines use cases with a number of different techniques in a concurrent engineering approach in an effort to increase coverage of use cases, and provide some validation of use case models through the use of other models which can be checked against them.

Use cases have been argued to provide a good basis for developing socio-technical systems as they enable inter-disciplinary learning of the kind which may be necessary when development team members are drawn from the somewhat disparate disciplines of, for example, ethnography, human-computer interaction and software engineering (Wiedenhaupt et al, 1998). They are acknowledged to provide an intuitive ‘middle-ground abstraction’ which encourages stakeholder participation (Jarke and Kurki-Suonio, 1998), and are currently used in requirements elicitation by around half of all organisations included in a recent survey (Neill and Laplante, 2003).
However, a number of difficulties have been identified for those working with use cases alone. For example:

- We cannot specify a new system to support work without understanding how that work is currently done (e.g. Haumer et al, 1999) – in RESCUE we use human activity modeling to build this understanding;
- We cannot write detailed use cases without establishing the system boundaries – in RESCUE these are explored through the development of context and i* models (Yu, 1997);
- We cannot write use cases without knowing about dependencies between actors described in the use cases – in RESCUE these dependencies are thoroughly explored using i* models;
- We cannot write use cases without making at least some high-level design decisions – in RESCUE we use creativity workshops to do this;
- We cannot write testable requirements without knowing the context in which those requirements arise (Robertson & Robertson, 1999) – in RESCUE we provide this context by linking requirements to scenarios and use cases.

Thus RESCUE aims to complement the deficiencies of use cases with a range of different techniques in order to better support the development of large socio-technical systems where confidence in the correctness and completeness of use cases, and hence requirements, is important.

**THE RESCUE PROCESS**

The RESCUE process consists of a number of sub-processes, organised into 4 ongoing streams. These streams run in parallel throughout the requirements specification stage of a project, and are mutually supportive. The streams focus on the areas of:

- Analysis of the current work domain using **human activity modelling** (based on work described in Diaper, 1989; Schraagen et al, 2000 and Vicente, 1999);
- **System goal modelling** using the i* goal modelling approach (Yu, 1997);
- **Use case modelling** and specification, followed by systematic **scenario walkthroughs** and scenario-driven **impact analyses** using the CREWS-SAVRE and CREWS-ECRITOIRE approaches (Sutcliffe et al, 1998);
- **Requirements management** using VOLERE (Robertson and Robertson, 1999) implemented in Rational’s requirements management tool RequisitePro in current rollouts of RESCUE.

In addition to these four streams, the RESCUE process uses the **ACRE** framework to select techniques for requirements acquisition (Maiden and Rugg, 1996), and **creative design workshops**, based on models of creative and innovative design (Maiden and Gizikis, 2001), to discover candidate designs for the future system, and to analyse these designs for fit with the future system’s requirements. Both ACRE and the creative design workshops have implications for all streams. ACRE is used to support the selection of methods for requirements acquisition at any point in the RESCUE process where there is a need for further requirements information. The creative design workshops use inputs from all streams as a baseline for creative thinking, and
provide outputs which inform the development of i* and use case models, as well as the identification of requirements for the future system.

Consistency between the various artefacts and deliverables produced at different stages in the RESCUE process is checked at five different points during the process. These 5 ‘synchronisation points’ provide the project team with different perspectives from which to analyse system boundaries, goals and scenarios, as follows:

- At the first point - the **boundaries** point - the team establishes first-cut system boundaries and undertakes creative thinking to investigate these boundaries;
- At the second point - **work allocation** - the team allocate functions between actors (human and technical) according to boundaries, and describe interaction and dependencies between these actors;
- At the **generation** point, required actor goals, tasks and resources are elaborated and modeled, and scenarios are generated;
- At the **completeness** point, stakeholders have walked through scenarios, and express all requirements so that they are testable;
- At the **consequences** point, stakeholders have undertaken walkthroughs of the scenarios and system models to explore impacts of implementing the system as specified on its environment.

An overview of the process is provided in figure 1.
The rest of this section provides a ‘stream by stream’ view of activities carried out in the RESCUE process. These activities will be illustrated by reference to the Countdown system, and a short overview of this system is given below.

**Countdown System Overview**

Countdown is the system currently used by London Buses to provide bus arrival times on indicators at bus stops all over London. While sharing some important similarities with systems used in air traffic control, this system is small and familiar enough to readers to provide meaningful insights into the application of the RESCUE process.

An example of the format used on current bus stop indicators is shown in figure 2.

```
1   207 ACTON MKT PL 1 min
2     83 GOLDERS GREEN 3 mins
3   207 SHEPHERDS BUSH 4 mins
.......Delays due to London Mayor's Show
```

**Figure 2: A simple depiction of a bus stop indicator**

Countdown carries a number of pieces of information to waiting passengers in a clear, easy-to-understand form:

- The order in which buses will arrive at the stop;
- The number of each bus;
- The destination of each bus – this information originates from the driver who keys a two-digit code into the system at the start of the journey;
- The time until the bus arrives – based on how long the central computer estimates it will take the bus to reach the bus stop from where it currently is;
- Base-line messages – the base line of the Countdown display can scroll messages across the screen from left to right every 90 seconds – messages convey general information on matters such as night buses and congestion.

Drivers are directed to start, or sometimes curtail their journeys by route controllers, who monitor the position of all buses on a number of different routes on the central computer system. The location of the bus is currently calculated by the automatic vehicle location (AVL) system using information from roadside beacons placed at intervals along the bus route. However, this system has some disadvantages – for example, when a bus is diverted (maybe because of road works or an accident) onto a different route where there are no beacons, information held on the AVL system, and displayed on bus stop indicators, quickly becomes out of date and incorrect. To overcome this, a new system is planned which will use GPS technology to more accurately locate the position of buses at all times. An additional enhancement planned in the new system is that bus arrival information should be made available to potential passengers in a number of different
formats and on a number of different devices such as mobile phones and PDAs. This will enable passengers to plan their travel wherever they are – they will no longer need to be at the bus stop in order to know when the next bus is due.

**Human Activity Modeling**

In this RESCUE stream (shown as ‘Activity Modelling’ in figure 1) the project team develops an understanding of the current socio-technical system to inform specification of a future system. Human activity modeling focuses on the human users of the technical system, aiming to build understanding of the controllers’ current work – its individual cognitive and non-cognitive components and social and co-operative elements, as well as the environment in which it takes place – in order to specify a technical system that can better support that work. It draws on the literature of task analysis (eg Diaper, 1989), cognitive task analysis (eg Schraagen et al, 2000) and cognitive work analysis (eg Vicente, 1999). The stream consists of two sub-processes – data gathering and human activity modeling.

During the first sub-process (shown as ‘gather data on human processes’ in figure 1), data about all components of the activity model is gathered and recorded. Techniques to gather this data include observation of current system use; informal scenario walkthroughs, using scenarios representative of how the current system is used; interviews with representative human users; analysis of verbal protocols, or recordings of users talking through scenarios or tasks; and ethnographic techniques.

In the second sub-process (shown as ‘model human activity’ in figure 1) the project team creates a ‘human activity model’ by generating a number of ‘human activity descriptions’ corresponding to each of the major types of activity in the current system. An activity model is a repository of information about various aspects of the current system including:

- **Goals** - desired states of the system
- **Human actors** - people involved in the work of the 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.

Further information may be found in Maiden and Jones (2004a). Extracts from a human activity description relating to the passenger activity of making travel decisions in the current Countdown system are shown in figure 3.
Figure 3: Extracts from a human activity description for the Countdown system

As can be seen from the figure, the human activity description template provides place holders for each of the types of information identified above. It has been designed in a similar way to our use case description template (shown in figure 8) in which we describe the desired behaviour of the future system, so that information about how the current system supports the work of human actors can be used quite easily to help develop and check proposals for the future system. Note that actions in the normal course of the human activity description are broken down into their physical, cognitive, and communicative components. This information is used in generating scenarios to walk through as described later in the chapter.

Activity models developed in this way provide important sources of data for the development of use case models and use case authoring, as well as data for fit criteria for system requirements. In writing these activity descriptions, the team also obtains a better understanding of the work and application domains, which is essential for effective requirements acquisition.
System Modeling

In this RESCUE stream (shown as ‘system goal modelling’ in figure 1) the project team models the future system’s actors (human and otherwise), dependencies between these actors and how these actors achieve their goals, to explore the boundaries, architecture and most important goals of the socio-technical system. RESCUE adopts the established i* approach (Yu, 1997) but extends it to model complex technical and social systems, establish different types of system boundaries, and derive requirements.

The system modeling stream requires 3 analyses to produce 3 models. The first model, generated in the first sub-process of this stream (‘determine system boundaries’ in figure 1) is a context model, similar to that used in the REVEAL and Volere processes (see Hall, 2001 and Robertson & Robertson, 1999) but extended to show different candidate boundaries for:

- The technical systems, expressed in terms of software and hardware actors within the inner boundary (in the case of the Countdown system, shown in figure 4, the technical system is made up of two sub-systems);
- The redesigned work system, expressed primarily in terms of human actors, within the middle boundary (in figure 4, there are two human actors who interact directly with technical sub-systems, and one technical actor, which also receives data directly from one of the technical sub-systems);
- Other actors which are strongly influenced by the redesign of the new system, although they do not interact directly with it, are shown within the outer boundary (in figure 4, the only actors of this kind are passengers);
- Systems that interact with elements of the new socio-technical system but are not strongly influenced by its redesign are shown outside the outer boundary (in figure 4, this includes the GPS system, which provides data about bus locations, the communication system used in communications between drivers and route controllers, and the London Transport central computer system).

A completed example of a context model for the Countdown system is shown in figure 4.
Figure 4: Context model for the Countdown system

Figure 5: The Countdown Strategic Dependency Model
The second model is the i* Strategic Dependency (SD) model, which describes a network of dependency relationships among actors identified in the context model. A first cut of this model is produced in the second sub-process of the system goal modelling stream (‘determine system dependencies, goals and rationale’ in figure 1), and then refined in the third sub-process (‘refine system dependencies, goals and rationale’). In an SD model, 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. For further explanation of the i* notation see Yu (1997). An SD model showing the main dependencies between actors relevant to the Countdown system is shown in figure 5.

The third model is the i* Strategic Rationale (SR) model, which provides an intentional description of how each actor achieves its own goals and soft goals. First cut and refined versions of this model are developed in the second and third sub-processes of the system goal modelling stream as described above. The SR model includes goals, tasks, resources and soft goals from the SD model, as well as task decomposition links, means-end links, and contributes-to-soft goal links which provide a more detailed view of each individual actor’s behaviour (Yu, 1997). An SR model for the Countdown system is shown in figure 6.

To support i* modeling, we have developed REDEPEND, a graphical modeling tool developed as a plug-in to Microsoft Visio that enables the team to construct and analyse i* SD and SR models (Maiden et al, 2002).
This stream provides key inputs to use case modeling. Context and i* models define the system boundaries that enable use case modeling and authoring. Furthermore, the i* SR models provide a basis for validating use case descriptions prior to scenario walkthroughs.

**Creativity Workshops**

In this RESCUE process, the team carries out some high-level creative design activities in parallel with on-going requirements work. This process, which takes inputs from, and provides outputs to, sub-processes in each of the RESCUE streams, is shown as ‘creative design workshops’ in figure 1. Further information about how creativity workshops are run can be found in (Maiden and Jones, 2004a) and (Maiden et al, 2004). A brief summary is provided below.

Workshop activities were designed based on 3 reported models of creativity from cognitive and social psychology. Firstly, we design each workshop to support the divergence and convergence of ideas described in the CPS model (Daupert, 2002). As such each workshop period, which typically lasts half a day, starts from an agreed current system model, diverges, then converges towards a revised agreed model that incorporates new ideas at the end of the session. Secondly, we design each workshop period to encourage one of 3 basic types of creativity identified by Boden (1990) – exploratory, combinatorial and transformational creativity. Thirdly, we design each period to encourage 4 essential creative processes reported in Poincare (1982): preparation, incubation, illumination and verification.

Exploratory creativity is encouraged by asking stakeholders to reason about the future system using analogies from different domains such as textile design and musical composition, and combinatorial creativity is triggered by random idea generation and parallels with, for example, fusion cooking. The use case model and descriptions are used as essential inputs and structuring mechanisms for all new requirements and design ideas. Throughout a workshop, each use case is displayed on a separate pin board. The facilitators instruct participants that all new requirements and other ideas generated during the workshop should be related to one or more use cases, indicated by the posting of the requirement or idea on the relevant board. After each half-day session, the use cases, requirements and ideas are reviewed, leading to some rewriting of the use cases prior to the next session. As such, the outputs from the workshop are better structured to enable a RESCUE project team to write detailed use case descriptions more effectively.

This concurrent design process benefits the requirements process in two ways. Firstly, the candidate design space reduces the number of requirements to consider by rejecting requirements that can not be met by current technologies. Secondly, high-level decisions about a system's boundaries enable the team to write more precise use cases and generate more precise scenarios that, in turn, enable more effective requirements acquisition and specification.

**Scenario-driven Walkthroughs**

In this RESCUE stream (shown as ‘use case modelling’ in figure 1) the team develops a use case model, writes use case descriptions, then generates and walks through scenarios to discover and acquire stakeholder requirements. We have applied results from the EU-funded CREWS project (Sutcliffe et al, 1998) to provide method guidance for use case authoring, 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 (‘develop use case model’ in figure 1) employs inputs from the context model (developed in the system modeling stream), to investigate different system boundaries. The outcome is a use case diagram with use cases and short descriptions that are inputs into use case authoring. An example of a use case diagram for the case study is shown in figure 7.

![Use case diagram for the Countdown system](image)

**Figure 7: Use case diagram for the Countdown system**

In the second sub-process (‘describe use cases’ in figure 1), the team writes detailed *use case descriptions* using a structured template derived from use case best-practice (e.g. Cockburn, 2000). Use cases are described as in UML, but remembering that they should define interactions between human and other actors at levels 2, 3 and 4 of the context diagram, as well as interactions with the technical system. Extracts from a completed use case description relating to the Countdown system are shown in figure 8. 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, for our air traffic management (ATM) projects, an extensive lexicon of ATM nouns and verbs elicited from controllers. To write each description the team draw on outputs from the other streams – human activity models, strategic dependency and rationale models, stakeholder requirements, and innovative design ideas from the creativity workshops.

Once each use case description is complete and agreed, the team produce a parameterised *use case specification* from it (this is the ‘specify use cases’ sub-process in figure 1) to generate...
scenarios automatically using the CREWS-SAVRE software tool (Sutcliffe et al., 1998). Different types of action in the normal course of the use case lead to the generation of different alternative courses, which will be used to guide the stakeholders to consider how the future system should respond in the event of, for example, cognitive slips or communication errors as described below. A more detailed description of how this is done in the ATM domain is given in Mavin & Maiden (2003).

<table>
<thead>
<tr>
<th>Provide information for travel decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
</tr>
<tr>
<td>Author</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Actors</td>
</tr>
<tr>
<td>Problem statement (now)</td>
</tr>
<tr>
<td>Precis</td>
</tr>
<tr>
<td>Functional Requirements</td>
</tr>
<tr>
<td>Non-functional Requirements</td>
</tr>
<tr>
<td>Added Value</td>
</tr>
<tr>
<td>Justification</td>
</tr>
<tr>
<td>Triggering event</td>
</tr>
<tr>
<td>Preconditions</td>
</tr>
<tr>
<td>Assumptions</td>
</tr>
<tr>
<td>Successful end states</td>
</tr>
<tr>
<td>Unsuccessful end states</td>
</tr>
<tr>
<td>Different walkthrough contexts</td>
</tr>
<tr>
<td>Normal Course</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Variations</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Alternatives</td>
</tr>
</tbody>
</table>

Figure 8: Extracts from a use case description for the Countdown system

The fourth sub-process (‘walkthrough scenarios’ in figure 1), is pivotal to RESCUE, and involves walking through each generated scenario with stakeholders using bespoke software tool
support. Each scenario may be delivered for walkthrough in two forms – either through the web-based Scenario Presenter tool shown in figure 9, or as an interactive MicroSoft Excel spreadsheet that can downloaded by the session facilitator. Facilitators walk through the scenario with relevant stakeholders, guided by the Scenario Presenter, to consider each normal course event and each alternative course linked to that normal course event in turn. The same scenario may be considered in a number of different ‘walkthrough contexts’ in which stakeholders are asked to make different assumptions about the human or environmental context in which it takes place, for example considering how passengers may act differently at night or in bad weather. A scribe uses the tool to document all requirements and comments relating to each event. For example, when considering the event ‘The passenger looks at the Countdown display’ and the alternative ‘What if passenger has some unusual physical characteristics that affect his/her behaviour during this action?’, the scribe is asked to add a new functional requirement that ‘The Countdown system shall provide an audio facility’, as shown in figure 9. Further guidance on how to conduct a scenario walkthrough can be found in (Maiden and Jones, 2004a) and Maiden (2004).

The final sub-process (‘impact analysis’ in figure 1) uses a sample of these scenarios in order to investigate how the system, as specified, will impact key social and environmental factors such as job security, actor roles and responsibilities, and access to information. This is done in a series of impact inspection meetings using ‘reading techniques’ such as those identified in Travassos et al (2002). Questions about the potential impact of the proposed future system have been identified based on work by Heath et al (1993), Hughes et al (1997) and Viller and Sommerville (1999).

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

**Managing Requirements**

In this fourth RESCUE stream (‘requirements management’ in figure 1) the project team documents, manages and analyses requirements generated from the other 3 streams. Each requirement is documented using a modified version of 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 use cases and use case actions and places them in context, thus making it much easier to write a measurable fit criterion for each requirement.

This use case-driven requirement structure carries over into the requirements document itself, to improve both the readability of the document and the understandability of each requirement statement. The document is 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 10.
In ATM projects carried out to date, RESCUE requirements have been documented using Rational’s RequisitePro. Outputs from other streams, such as use case, context and i* models are all included in the document. The team also 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 of value to stakeholders.

1. **The passenger looks at the Countdown display**  
   FR28 The Countdown system shall provide an audio facility

2. **The Countdown display shows the bus arrival information for the relevant route(s)**  
   FR3 The Countdown system shall display the order in which the next three buses will arrive at the journey starting point, the number of each bus, the destination of each bus and the estimated time until the arrival of each bus  
   RR2 95% of estimated bus arrival times shall be correct to within 30 seconds

**Figure 10: Extracts from the Countdown requirements document**
Managing the RESCUE Process

RESCUE is a complex process and depends crucially on managing the activity carried out in different streams to ensure consistency between the different artefacts shown above, and to ensure that work in each stream can draw on the others as needed. Central to this are the checks carried out at the five synchronization points identified at the beginning of this section, and shown in figure 1. In overview, these checks are as follows. Further information and examples can be found in Maiden and Jones (2004b).

At the end of stage 1, data about human activities and the context model are used to check the completeness and correctness of the use case model. System-level requirements are used to check use case summaries.

Most cross checking is done at stage 2 in order to bring the human activity and first-cut $i^*$ models to bear on the development of correct and complete use case descriptions. Components of the human activity descriptions are checked against the $i^*$ models and use case descriptions, with particular attention being paid to areas where the future system will be different from the existing one. $i^*$ models and use case descriptions are checked against each other, and a first set of requirements is derived from both the $i^*$ models and the use case descriptions.

At the end of stage 3, use case specifications are checked using the $i^*$ models, and the refined $i^*$ models are used to check the requirements database.

Checks carried out at the end of stage 4 and 5 relate solely to the internal structure of the requirements database, as no new artefacts are generated during these stages.

Future Trends

To identify requirements for future socio-technical systems, we need integrated requirements processes which draw on human- as well as techno-centric disciplines. This has implications both for the teams of people carrying out the work, as well as the tools, techniques and artefacts which are a part of the processes used. Teams must be drawn from a variety of disciplines such as human-computer interaction, ethnography, cognitive task analysis and software engineering (see, for example, Viller and Sommerville, 1999). Tools, techniques and artefacts must enable the intertwining of inputs from each of these disciplines. We believe that our scenarios provide a valuable tool for capturing insights regarding current work practices, as well as detailed knowledge about human-computer interaction, and integrating them into a framework which is now familiar to most software engineers. To further explore the application of these techniques in new contexts, we have recently developed a version of our scenario walkthrough tool for Personal Digital Assistants (PDAs), so that scenario walkthroughs can be done in the work context, thereby linking contextual inquiry and structured walkthrough techniques for requirements discovery (Seyff et al, 2004).
Conclusion

In this chapter, we have presented RESCUE, a concurrent engineering approach to requirements specification, which combines use cases with a number of different techniques from both software engineering and other disciplines. We have learnt from our experience with RESCUE that use cases, when complemented by other techniques – i* modeling, creativity workshops, CREWS-SAVRE scenario walkthroughs, etc – do indeed provide a solid foundation for the specification of requirements for complex socio-technical systems. We do not attempt to capture all of the information derived through this process using any single formalism, but rather see the strength of our approach as its use of multiple, integrated, representations, which support a systematic, analytic, yet creative approach to the development of a final requirements document. On the strength of our experience to date, we argue that RESCUE focuses more attention on human elements of a socio-technical system, provides better support for identification of socio-technical system boundaries, and embodies a more systematic approach to the discovery of requirements through scenario walkthroughs than other main-stream approaches based on use cases alone.

RESCUE has already been successfully applied in specifying requirements for the CORA-2 system introduced earlier (see Maiden et al, 2003a; 2003b), and at the time of writing, we have almost completed the application of RESCUE to the specification of requirements for Eurocontrol’s future DMAN Departure Management system, working together with the UK National Air Traffic Service. Applying the RESCUE process in CORA-2 led to the generation of an operational requirements document that had the confidence of stakeholders and passed reviews carried out by staff who were outside of the CORA-2 requirements team. The requirements document for CORA-2 contained approximately 400 requirements, structured using 22 use cases. Numbers for DMAN are likely to be similar. However, such an approach is not cheap. For both CORA-2 and DMAN, considerable training was required before the process could be applied. In each case, the requirements phase of the project has taken around 9 months, with 2 full-time requirements engineers as well as additional effort from stakeholders and domain experts. However, the process has worked, and investment in training is now yielding a good return as RESCUE is rolled out to two other projects in the ATM domain. We look forward to reporting fully on our experiences in these two projects in due course.

References


1 Note that the term ‘human activity modeling’ (or sometimes ‘activity modeling’ for short) as used here is distinct from the term ‘activity modeling’ as used by the HCI community in relation to design reasoning.