Determining Socio-Technical Systems Requirements: Experiences with Generating and Walking Through Scenarios

Alistair Mavin
Praxis Critical Systems Ltd
alistair.mavin@praxis-cs.co.uk

Neil Maiden
Centre HCI Design, City University
n.a.m.maiden@city.ac.uk

Abstract

Scenarios are effective for discovering requirements, but we still do not understand what types of scenario and which walkthrough techniques are most effective. This experience paper reports the application of one scenario approach – CREWS-SAVRE – to discovering requirements for naval and air traffic management systems with BAE SYSTEMS and Eurocontrol respectively. Results from these experiences are used to investigate important questions about the effectiveness of structured scenario walkthroughs and the level of domain-specificity most beneficial to the discovery of requirements. Lessons learned suggest that systematic walkthroughs of simple scenarios that do not contain excessive domain knowledge are more effective for discovering system requirements. The paper provides the reader with simple-to-use guidelines for scenario-based requirements discovery.

1. Discovering Requirements with Scenarios

We know that scenarios are one of the more effective techniques for discovering stakeholder requirements [1]. However, scenarios can differ widely in their levels of abstraction and forms of representation, from narrative stories of how a system will behave [2] to computer-based animations of a system's dynamic behaviour [3]. We can also use scenarios in different ways, from helping us to understand a current system [4] to walkthroughs of a future system's behaviour to discover requirements for it [5]. However, there are no models and little reported evidence that tell us whether structured scenario walkthroughs and scenarios with more or less domain content lead to better requirements discovery. This paper reports results from recent, large-scale case studies of scenario-driven requirements processes to provide some answers.

The CREWS-SAVRE process and software tool were developed to deliver structured scenario walkthroughs with which to discover more complete stakeholder requirements [6, 7]. Design of its walkthrough tool was based on one cognitive principle often exploited during prototyping – that people recognise items, for example events, better than they recall them from memory [8]. CREWS-SAVRE generates and presents candidate alternative course events to stakeholders. The stakeholders then recognise which alternative courses are relevant to the new system and, where relevant, generate new requirements for them. This approach contrasts with existing use case methods (e.g. [2]) and scenario software tools (e.g. [9]) that primarily rely on people to recall the alternative courses themselves. Because people are better at recognising that recalling, we hypothesise that CREWS-SAVRE's guided scenario walkthroughs can help stakeholders to discover more complete requirements. However, the lack of large-scale case studies has meant that we have not been able to test this hypothesis empirically.

We also lack empirical data that indicates how domain-specific scenarios should be to discover requirements cost-effectively. For example, a simple scenario for an air traffic management (ATM) system might be expressed as *an air traffic controller communicating with an aircraft and the communication failing*. But the same scenario could also be expressed as *a tactical controller working the North Sea Sector at London ATC issuing a direction change to 270 to flight BA123, and the congested frequency meaning that the instruction is not issued successfully*. The second scenario probably needs more time and effort to produce, but does its more concrete content improve requirements discovery, and if it does, is the extra work needed to write it cost-effective? Case studies data are also needed to provide answers to these questions.

This paper reports results from two applications in which CREWS-SAVRE was applied to acquire requirements for a new naval system and a new ATM system. In both applications we extended the CREWS-SAVRE tool to generate domain-specific scenarios in the style of the North Sea sector scenario above. We then used the data about the resulting requirements and how they were arrived at to investigate whether structured scenario walkthroughs led to the discovery of more stakeholder requirements. We also used the data to investigate whether domain-specific scenarios enable stakeholders to discover more requirements, and whether generated CREWS-SAVRE scenarios can be specialised to the application domains at reasonable time and cost.

The remainder of the paper is in 9 sections. The next two describe CREWS-SAVRE and the two application domains. Section 4 presents a simple predictive model of the scenario walkthrough process. Sections 5, 6 and 7 describe how we specialised CREWS-SAVRE to the domains and how effective this specialisation was for requirements discovery. Section 8 draws some conclusions and reports lessons to apply when using scenarios for discovering requirements. The paper ends by describing ongoing and future technical extensions of CREWS-SAVRE and more controlled empirical studies of its use.

2. Generating and Walking Through Scenarios with CREWS-SAVRE

CREWS-SAVRE is a process with software tools that was developed as part of the EU-funded Framework IV 'CREWS' long-term research project to support systematic, domain-independent scenario generation and walkthrough process [7]. Features include:

- Guidelines for writing use cases;
- Automatic generation of scenarios from use cases;
- Automatic generation of alternative courses;
- Guided scenario walkthroughs.

A project team writes use case descriptions using the structured templates, and style and content guidelines from the CREWS-ECRITOIRE method [10], enhanced with temporal action-ordering rules. This description is then parameterised and entered into the CREWS-SAVRE tool, from which CREWS-SAVRE's two-step scenario generation algorithm generates one or more scenarios.

In the first step, the algorithm generates 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. In the second step, the algorithm generates candidate alternative courses, which are expressed as 'what-if' questions for each normal course event, by querying a database that implements a simple model of abnormal behaviour and state in socio-technical systems. The model specifies 54 classes of abnormal behaviour and state using the structure shown in Figure 1. Some class hierarchies were derived from definitions of CREWS-SAVRE scenario concepts such as events and actions. Others were derived from error taxonomies in the cognitive science, human-computer interaction and safety-critical disciplines, and are reported at length in [6]. For example, the algorithm generates alternative courses about events not occurring and actions not completing by instantiating the EC- and AC-coded classes in Figure 1, and about human agent mistakes, machine failures and interaction failures by instantiating the PE1-, PE2- and PE3-coded sub-classes in Figure 1.

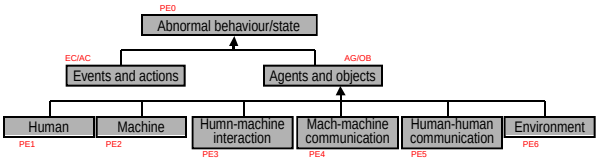

Figure 1. The original CREWS-SAVRE database.

CREWS-SAVRE generates such domain-independent alternative courses using domain-independent type models. Each normal course event is either the start or the end of an action that is typed as *cognitive*, *physical*, *communication*, *system-internal* or *composite*, and involves agents of different types, for example *human*, *machine* and *composite*. For each generated normal course event, the algorithm applies 17 domain-independent rules that link the action and agent types involved in the event with the 54 abnormal behaviour and state classes to generate candidate alternative courses that instantiate these classes. For example, if the event starts a cognitive-type action involving a human-type agent, the algorithm generates alternative courses that instantiate the cognitive-error classes that specialise PE1 (e.g. *what if the controller makes a slip?*). If the event starts a communication-type action involving 2 machine-type agents, then the algorithm generates alternative courses that instantiate the PE4 machine-communication classes (e.g. *what if communication between the machines is slow?*). This original rule structure and the number of classes, types and rules that implement it are depicted in Figure 2.

Normal course event		**IF** event has <agent> type or <action> type **THEN GENERATE** alternative with <abnormal> type	Alternative course event
Action type	*Agent type*		*Abnormal classes*
8 action and agent types		17 alternative course generation rules	54 classes of abnormal course

Figure 2. The structure of a CREWS-SAVRE alternative course generation rule, showing the number of action and agent types, rules and classes of abnormal behaviour and state in its domain-independent version.

Each generated scenario is delivered to stakeholders to be walked through in two forms – either as an interactive Microsoft Excel spreadsheet called the Excel Presenter that can downloaded by the session facilitator – or using the web-based Scenario Presenter tool shown in Figure 3.

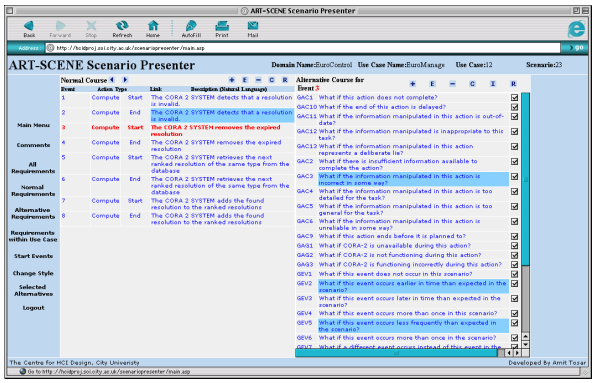

Figure 3. The Scenario Presenter Tool, available from www.soi.city.ac.uk/artscene.

Each scenario is in 4 parts. The left-side menu provides different functions for viewing the scenario and the requirements generated for it. The top-line buttons offer walkthrough functions (e.g. next or previous event) and functions to add, edit or delete events, 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 normal and alternative course event, the facilitator guides stakeholders to recognise whether:

- This event might occur;
- This event is relevant to the future system;
- Does the future system, as currently specified in the requirement document, handle the event?

If no requirements are specified to handle an event that is recognised as relevant, then omissions have been discovered and new requirements can be written, thus increasing requirements completeness. The tool documents all requirements and comments arising from the walk through.

Some alternative courses for one ATM scenario event generated by the original, domain-independent CREWS-SAVRE tool are shown in tabular form in Figure 4.

The controller issues a direction change to the pilot	
Gev1	What if this event does not occur in the scenario?
Gac1	What if this action does not complete?
Gev7	What if this event occurs less frequently than expected?
PE3	What if there is a communication failure?
PE1.2	What if the controller is physically unable to undertake this action?

Figure 4. A domain-independent scenario fragment generated by the CREWS-SAVRE algorithm. The alternative courses were generated for the start event *the controller issues a direction change to the pilot*. The codes of the left-hand side are the unique identifiers of the abormal behaviour and state classes from which each alternative was generated.

3. Applying CREWS-SAVRE to Discover Requirements – Naval Warfare and Air Traffic Management

We applied the CREWS-SAVRE process and software tool to discover requirements for two complex systems in two domains – naval warfare and air traffic management. In both domains we specialised the CREWS-SAVRE algorithm to generate scenarios with domain-specific alternative courses. Stakeholders then walked through these scenarios to discover requirements for: (i) a new naval system with BAE SYSTEMS as part of the bi-lateral SERPS project; (ii) a new air traffic management system with Eurocontrol as part of the CORA-2 project [11].

3.1. The Naval Warfare Domain

BAE SYSTEMS uses a powerful simulator called the Operational Concept Demonstrator (OCD) to discover requirements for future naval systems. However, the OCD only simulates certain normal behaviours, so systems engineers have to depend on recall of possible abnormal events to discover all candidate requirements. To overcome this limitation, BAE SYSTEMS funded a project to integrate the CREWS-SAVRE and OCD tools in the naval warfare domain. Our task was to specialise CREWS-SAVRE so that it would generate alternative courses specific to naval warfare. Challenges included eliciting, modelling and reusing domain knowledge to explore whether such specialisation is feasible, and determining how cost-effective such scenarios are for discovering requirements. Scenarios that are too general will lead to missing requirements, while too detailed scenarios will lead to duplicate requirements and inefficient walkthroughs.

3.2. The Air Traffic Management Domain

We worked with Eurocontrol to design and implement an innovative process to discover 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. The project team used CREWS-SAVRE to discover requirements for the CORA-2 system. Again our task was to

specialise CREWS-SAVRE to generate scenario alternative courses specific to the ATM domain. Challenges included eliciting, modelling and reusing ATM domain knowledge to explore the feasibility and cost-effectiveness of such scenarios during requirements discovery. To demonstrate how this specialisation was delivered in the scenarios, a small fragment of the same ATM scenario generated in the ATM-extended CREWS-SAVRE tool is shown in Figure 5. In contrast to the earlier scenario, these alternative courses make explicit reference to events or states that can occur in the ATM domain.

The controller issues a direction change to the pilot	
EC3.6	What if the controller lacks confidence in his own decision making?
EC7.1	What if the pilot is not listening to messages?
EC7.4	What if the pilot is using incorrect settings?
EC9.1	What if there is a heavy sector load?
PE1.2	What if the control room is noisy?

Figure 5. An ATM domain-specific scenario fragment generated by the CREWS-SAVRE algorithm. Again the codes of the left-hand side are the unique identifiers of the classes from which each alternative was generated.

4. A Tentative Model of Discovering Requirements from Scenarios

To inform the two case studies, we developed a simple model to predict:

- The types of requirement generated from alternative courses, classified according to the originating classes of abnormal behaviour and state?
- Whether stakeholders recognise domain-specific alternative courses that are relevant to the normal course event?

To predict what types of requirement were likely to be generated, we applied published quality models [e.g. 12] to align the types of units of measure needed to test different requirement types with the classes of abnormal behaviour and state used to generate the alternative courses in CREWS-SAVRE. The resulting model predicts that CREWS-SAVRE scenario walkthroughs will generate more requirements that are functional, performance, look-and-feel, information, usability, reliability and recoverability requirements, based on the definition of these types in [13]. Other requirements types, such as security, safety, maintenance and portability requirements, are not predicted to occur in the same numbers during the walkthroughs. Table 1 summarises the model and part of its rationale, linking classes of alternative courses to requirements types.

Alternative course classes	**Requirement type**
Actions not occurring	Functional
Abnormal event timings	
Actions not completing Abnormal event timings	Performance
Non-adherence to interface and design standards	Look-and-feel
Missing and incorrect information	Functional Information
Human agent errors Human agent-interaction errors	Usability
Machine agent errors Machine-agent interaction errors	Reliability, Recoverability

Table 1. Predicted types of requirements generated from different alternative course types.

The model predicts, for example, that stakeholders will generate functional requirements that specify either *monitor* or *warn* functions in response to an *action not occurring*, or functions to undertake an action that replaces the non-occurring action. Likewise, alternative courses about abnormal event timings and actions that do not complete will lead to performance requirements expressed in terms of event timings and throughput. Alternative courses about missing or incorrect information will generate functional and information requirements to make the information available.

Recognising whether each generated alternative course is *relevant* is an essential pre-requisite to discovering new requirements from scenario walkthroughs. The design of CREWS-SAVRE is based on evidence that people are better at identifying errors of commission rather than omission [14], that is they are better at recognising incorrect rather than missing alternative courses. From this general trend in human cognition for recall to be weaker than recognition (e.g. [8]) we predict that stakeholders will discover more requirements using CREWS-SAVRE alternative courses that are presented to them to recognise as relevant than they will from unaided recall of such events. The downside, of course, is that not all machine-generated alternative courses will be relevant to each event.

So how can we design CREWS-SAVRE walkthroughs so that stakeholders *recognise* the alternative courses that are relevant to each event? Again human cognition, and in particular theories of basic categories, can provide an answer. People categorise their knowledge about the world hierarchically and form what are called basic categories – an optimum level of abstraction in a mental hierarchy – to enable recognition and recall of this knowledge [15]. To most people the concept of an *aircraft* is a basic category, *means of transport* a higher-level category and *Boeing 737* a lower-level category. However, increasing domain expertise tends to result in lower-level basic categories, for example an experienced air traffic controller might have basic categories such as *commercial airliner*, *personal aircraft* and *military jet*. Therefore, our model predicts that stakeholders – in our applications expert naval

engineers and air traffic controllers – will be more successful at recognising alternative courses that are equivalent to their basic-level categories of abnormal behaviour and state. That is, the stakeholders will recognise the lower-level classes of abnormal behaviour and state that encapsulate the expertise specific to the domain elicited earlier from experts in that domain.

The second half of this paper describes how we tailored CREWS-SAVRE to the naval warfare and ATM domains, the success of this tailoring with respect to the predictive model, and lessons learned to improve scenario-driven requirements processes that we have gained from the experiences.

5. Tailoring CREWS-SAVRE to the Naval Warfare and ATM Domains

Tailoring CREWS-SAVRE to the naval warfare and ATM domains involved:

- Eliciting knowledge about abnormal domain behaviours and states from domain experts;
- Modelling these abnormal behaviours and states as UML class specialisation hierarchies;
- Extending the CREWS-SAVRE software tool to generate scenarios with domain-specific alternative courses.

5.1. Eliciting Domain Knowledge

We elicited domain knowledge from experts who worked in pairs and were encouraged to converse with each other. This technique, known as constructive interaction [16], overcomes the unnatural aspects of 2 elicitation techniques – informal scenario walkthroughs and laddering – that we combined to elicit knowledge about abnormal behaviours and states in each domain.

During a session, each pair was asked to describe, then walk through, simple scenarios of how relevant current naval and ATM systems work. During the walkthrough we applied laddering [17], a form of semi-structured interview that involves repeatedly asking a small set of probe questions to provide hierarchical classifications of the elicited knowledge. At each scenario event in the normal course scenario the elicitation team asked the 5 questions shown in Table 2.

1. What unexpected or undesired behaviour can occur at this event?
2. In what circumstances could this behaviour occur?
3. This is an example of what class of exception?
4. What other examples of this class could occur at this event?
5. What other classes of exception can occur at this point?

Table 2. The laddering elicitation probes applied.

Raw data, in the form of written notes and audio tape recordings, described informal class hierarchies of abnormal behaviour and state containing both class- and instance-level behaviours and states.

5.2. Modelling Class Hierarchies

Results were modelled as UML class hierarchies. The name of the domain experts responsible for offering each exception class was shown on the model to trace its source. The models were then validated with the same domain experts through iterative model revision, validation and amendment.

5.3. Tailoring CREWS-SAVRE

Each modelled class of abnormal behaviour or state was given an identifier consistent with the original CREWS-SAVRE classes, then added to the CREWS-SAVRE database with links to its super- and sub-class abnormal behaviour and state classes in the relevant hierarchy. Next, the elicitation team worked with the domain experts to specify rules that would generate domain-specific alternative courses by instantiating these new classes for different types of scenario normal course event. For both domains we developed a lexicon to describe agents and objects (lexicon nouns) involved in the actions (lexicon verbs) describing the event itself. The purpose of the lexicon was to generate the most relevant alternative courses based on agent, object and action types for each normal course event in the scenario. So, for the example event *the controller issues a change of direction to the pilot*, CREWS-SAVRE generates alternative courses that instantiate abnormal behaviour and state classes relevant to *verbal communication* (the action type), a *human controller* (one agent type) and a *human pilot* (the other agent type).

The tailored CREWS-SAVRE tool was then applied to generate and walk through scenarios in the two domains.

6. Results: CREWS-SAVRE in the Naval Warfare Domain

For the naval warfare domain, we elicited 112 new classes of abnormal behaviour and state structured in specialisation hierarchies. This knowledge was elicited from 2 pairs of experts walking through 2 scenarios in just over 5 session hours. These classes were modelled using UML class hierarchies, then implemented in the CREWS-SAVRE tool to generate scenarios with domain-specific alternative courses. Figure 6 shows part of the class hierarchy for *Exceptional platform state*. CREWS-SAVRE uses it to generate alternative courses such as *what if the submarine is in a vulnerable state?* and *what if the submarine is at periscope depth?*

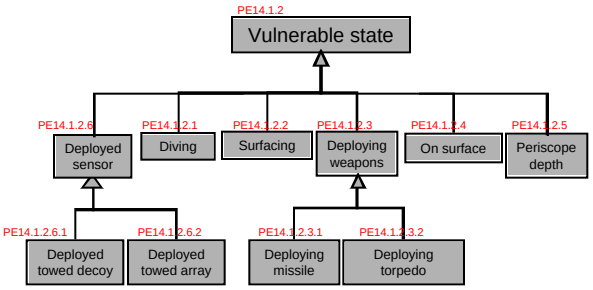

Figure 6. Exception Classification for *Exceptional platform state.*

The models developed directly from the expert data revealed that the domain experts described not only the *cause* of the abnormality but also its *consequence*. For example, the class *low salinity of water* was only cited in the context that this exception caused *poor sonar performance*. Similarly, *shallow water* was only considered relevant when it caused *restricted movement*. To reflect this cause-consequence distinction, we added new class hierarchies to describe the abnormal environmental states that cause abnormal states encountered by naval platforms.

We also produced a simple lexicon of naval domain agents, objects and events. We extended the original CREWS-SAVRE types using the lexicon, for example a *communication-type* action could be described as a *broadcast-type* or a *sense-type* action. The result, depicted in Figure 7, was 83 new naval domain action, agent and object types that extend the original 8 domain-independent types, and 112 new classes of abnormal behaviour that specialise the original 54.

Normal course event			**IF** event has <agent> type or <object> type or <action> type **THEN GENERATE** alternative with <abnormal> type	Alternative course event
Action type	*Agent type*	*Object type*		*Abnormal classes*
83 action, agent and object types			129 alternative course generation rules	112 classes of abnormal course

Figure 7. The structure of a CREWS-SAVRE alternative course generation rule extended to the naval warfare domain.

Unfortunately, opportunities to generate and walk through CREWS-SAVRE naval domain scenarios with BAE SYSTEMS were limited. One 36-event normal course scenario of an air-launched missile attack was generated to demonstrate integration with the OCD and presented to BAE SYSTEMS stakeholders. Responses were favourable, but the project had inadequate resources to continue the work, although it led to ongoing collaboration as part of the SIMP project.

On the other hand, these results revealed that tailoring CREWS-SAVRE to generate scenarios with domain-specific alternative courses was possible and potentially cost-effective, and motivated the larger exercise with the ATM domain.

7. Results: CREWS-SAVRE in the Air Traffic Management Domain

We elicited 138 new classes of abnormal behaviour and state specific to ATM from 4 experienced air traffic controllers who walked through 5 scenarios in just over 8 session hours. This knowledge was again modelled using UML specialisation hierarchies and implemented in the CREWS-SAVRE tool to generate scenarios with ATM-specific alternative courses. We also elicited, from 2 controllers, a lexicon of 140 agents, objects and actions that extended the original CREWS-SAVRE types to the ATM domain. For example, the *machine-agent* type was specialised to the *radar*- and the *aircraft-type*. Finally, we worked with one of these 2 controllers for a further 2 days to develop over 500 domain-specific rules to generate domain-specific alternative courses for different ATM domain actions, agents and objects. The extended CREWS-SAVRE rule structure is shown in Figure 8.

Normal course event			**IF** event has <agent> type or <object> type or <action> type **THEN GENERATE** alternative with <abnormal> type	Alternative course event
Action type	*Agent type*	*Object type*		*Abnormal class*
140 action, agent and object types			527 alternative course generation rules	138 classes of abnormal course

Figure 8. The structure of a CREWS-SAVRE alternative course generation rule extended to the ATM domain.

The class hierarchy describing the abnormal *radar* agent behaviours and states is shown in Figure 9. Generated alternative courses for normal course events that involve the radar agent include *What if there is poor radar performance?* (from EC5.4) and *What if the transponder is not selected?* (from EC5.2.1). Other class hierarchies were specified using knowledge about *system data*, *non-standard aircraft performance*, *controller cognitive behaviour*, *human-machine interface presentation*, *pilot behaviour*, *weather*, *air space*, *the control room environment*, *radio*, and *human-human communication*.

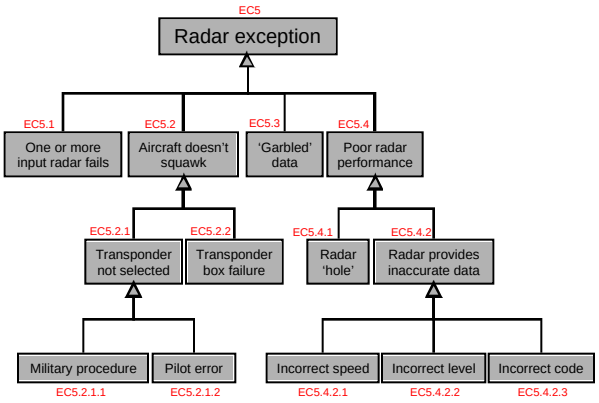

Figure 9. Part of the ATM abnormal behaviour model, showing sub-classes relating the radar agent.

The CORA-2 team using the specialised CREWS-SAVRE tool generated scenarios for 10 of the 22 use cases specified for the CORA-2 system. A facilitator then led stakeholders to walk through these 10 scenarios using the Excel Presenter tool similar to that shown in Figure 2. Each walkthrough lasted 3-4 hours and took place during a 3-week period. The scenarios contained a total of 254 normal course and 3050 generated alternative course events. The 10 sessions led to 189 new stakeholder requirements for CORA-2, 136 of which were retained after removing duplicates. This almost doubled the number of requirements established using other techniques over the preceding 6 months, and gave the team confidence that most of the key requirements were being considered.

The total number of requirements, according to the type attributed by CORA-2 team members, are shown in Figure 3. Most discovered requirements were functional (70.9%) and the majority of the remainder were reliability- and usability-type requirements. No requirements of some types – availability, maintainability, safety and security – were discovered. A paired t-test of the number of functional and non-functional requirements generated per walkthrough revealed that each scenario generated significantly more functional than non-functional requirements (paired t-test, $t=1.71$, $P<0.02$).

Requirement type	Total
Functional	95
Back-up	1
Performance	5
Reliability	16
Security	2
Training	3
Usability	12
Undefined type	2
Total	136

Table 3. Totals of requirements by type discovered during the CORA-2 scenario walkthroughs.

In retrospective interviews, CORA-2 team members believed that many of these 136 requirements tended to be more complete and precise than the stakeholder requirements acquired using other techniques. According to the team's Quality Gatekeeper, "*the functional requirements derived from the walkthrough were lower level than the existing functional requirements. The derived requirements were also much more meaningful as they were derived and discussed in the proper context – use case events*". The Gatekeeper also stated that, "*The majority of the [original] requirements were based on the CORA-2 system providing some kind of functionality to the controller or linked to the calculation of the resolution. From the walkthroughs similar requirements were derived but also functional requirements that address such areas as communication with other systems, checking and validating information and reporting functions*", a conclusion supported by analysis of the documented new functional requirements. One reason for the greater number of functional requirements came from the walkthrough facilitator, who reported that, "*due to time constraints the emphasis was often on functional rather than non-functional requirements*", and worded facilitation prompts accordingly.

Post-walkthrough questionnaire responses from 7 key participants shown in Table 4 revealed that the walkthroughs were usable and understandable, but there were individual differences of opinion over the effectiveness of the walkthroughs to discover requirements.

Question	Low	High	Comments
How well do you understand the technique?	5	6	Some alternative courses are difficult to understand
How easy was it to discover requirements?	4	6	Exhaustive elicitation process
How well did the tool support you in discovering requirements?	2	7	Impressive coverage Problems with the alternative courses

Table 4. Questionnaire responses on a scale of 1-7 (1=very badly, 7=very well) from 7 stakeholders who participated in the CORA-2 scenario walkthroughs.

We also examined where each new requirement came from. Of the 136 new requirements for the CORA-2 system, 51 originated from consideration of normal course events in the scenario and 83 from alternative course events. A further 2 new requirements could not be attributed to a normal or an alternative course event. Of the 83 requirements originating from alternative course events, 79 (95.2%) of these courses were instances of domain-independent classes of abnormal behaviour and state. This was not just an effect of the number of domain-independent and domain-dependent alternative courses generated in the 10 scenarios – 646 (21.2%) of these courses were instances of the domain-specific classes.

We also sought new requirements for future versions of CREWS-SAVRE from the facilitators and stakeholders to improve the process and software tool. Summaries of some important requirements are given

in Table 5. Overall, the feedback was positive. Many of the new requirements were motivated by the need to give users as much control as possible over the scenario artefact during a scenario walkthrough.

Users shall be able to add, edit and delete any normal or alternative course event at any time during the scenario walkthrough.
All parts of the scenario are visible to the users at any time during a walkthrough.
Users shall be able to add, edit and delete stakeholder requirements at any time during the scenario walkthrough.
Users shall be able to select different views of a scenario at any time during a scenario walkthrough.

Table 5. Important requirements to improve the walkthrough process and tools.

8. Conclusions and Lessons Learned

Structured scenario walkthroughs in the ATM domain were considerably more effective for discovering requirements than the brainstorming and interview techniques applied earlier in the CORA-2 process. Most stakeholders found the walkthrough process easy to understand and use, but had different opinions about its effectiveness, in spite of large numbers of new requirements being acquired. Most new requirements were functional, due in part to the facilitation prompts, and most were more testable and had greater coverage than the earlier requirements.

In contrast, evidence to support domain-specialisation of the scenarios was weak. Our approach to eliciting and modelling domain knowledge in the two domains was relatively cost-effective. The CREWS-SAVRE tool extended with this knowledge generated scenarios with full sets of alternative courses that the ATM domain experts were able to work with throughout the walkthroughs, although questions arose about some individual alternative courses. However, most generated requirements were not associated with these domain-specific alternative courses, and this warrants some explanation.

The scenario walkthroughs were undertaken as part of an industrial project with complex time, resource and confidentiality constraints. Because of this, we were unable to intervene in or capture real-time data about the scenario walkthroughs themselves, and so lack the first-hand data about how recognition of alternative courses occurred. One possibility, which exploits CREWS-SAVRE's design, is that the stakeholders separated the recognition of relevant alternative courses from the subsequent attribution of new requirements to the alternative courses – in short what triggered the new requirement was not what was recorded in the tool. CREWS-SAVRE presents alternative courses to users as demonstrated in Figure 10 – domain-independent and high-level domain-specific alternatives often precede lower-level domain-specific ones. We conjecture that stakeholders sometimes browsed the complete list of alternative courses for each normal course event to recognise relevant alternatives, but attributed new requirements to the more general alternative course to ensure their greater applicability in the final specification. However, without first-hand data, this explanation remains conjecture and we cannot rule out whether domain-specific scenarios can be more effective.

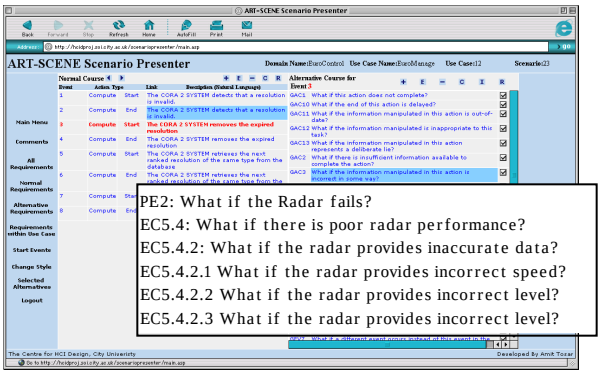

Figure 10. Example Scenario Alternative Courses.

Another possible reason for the apparent ineffectiveness of domain-specific alternative courses is that the larger number of domain-independent alternative courses in the 10 generated scenarios might have created a training bias. Stakeholders might have become more skilled at exploring a small number of recurring alternative courses to minimise the cognitive effort needed during a 3-hour walkthrough. The requirements and their sources provide some evidence to support this explanation. Thirty-three of the 79 requirements linked with the domain-independent alternative courses were generated by just 5 of the 54 possible classes of abnormal behaviour and state – *object is not functioning*, *information is out of date*, *event occurs more than once*, *incorrect event occurs* and *object cannot participate at this time*. We do not know whether this is due to a possible training bias that arose from frequent use of these alternative courses, or whether it is because the participants found the alternative courses more relevant, or both. Future work will investigate the reasons and the strengths and weaknesses of this behaviour to improve CREWS-SAVRE scenario walkthroughs.

Overall, however, the results from the naval warfare and ATM domains are conclusive enough for us to be able to recommend 5 lessons for organisations using scenarios and use cases to discover requirements from stakeholders:

1. Make your walkthroughs domain-specific;
2. Impose structured walkthroughs;
3. Stakeholders own the scenarios;
4. Encourage exploration with scenarios, and;
5. Combine different walkthrough styles.

Each lesson is presented in turn.

8.1. Make your walkthroughs domain-specific

Eliciting, modelling and reusing knowledge about abnormal behaviours and states in the naval warfare and ATM domains proved to be tractable. Our method resulted in large and validated models (>100 new classes in either domain) in a short period of time with relatively few resources. The potential for reusing the ATM models is considerable – we are now applying CREWS-SAVRE to discover requirements for two other projects within Eurocontrol's ASA programme with little extension of the models. If you are not using CREWS-SAVRE, you can still use these unusual domain models manually during scenario or use case walkthroughs in the form of checklists for each event. These checklists can be viewed as extensions to methods such as HAZOPS to include unusual domain behaviour not directly associated with hazards or errors, but still useful for discovering requirements.

8.2. Impose structured walkthroughs to discover requirements

The structure imposed by CREWS-SAVRE's tool led to better walkthrough coverage and greater requirements completeness. We believe that organisations can impose structure during walkthroughs without the CREWS-SAVRE tool by imposing rigorous facilitation processes or developing makeshift tools themselves. Facilitation guidelines can ensure that each scenario or use case event is given equal consideration, especially if combined with the alternative course checklists from the previous lesson. Makeshift walkthrough tools can also help. For example, side-by-side projection of MS Word documents, one containing the normal course, the other containing the alternative course checklists, can encourage stakeholders to recognise candidate alternative courses using the principles of CREWS-SAVRE at a low set-up cost to an organisation.

8.3. Stakeholders own the scenarios

After the walkthrough sessions, the facilitators and stakeholders revealed that they wanted far more control over our scenarios during walkthroughs, especially as the scenarios were machine-generated. In the new Scenario Presenter tool shown in Figure 2, we have implemented the extensive end-user tailorability requested. For example, a user can to remove generated alternative courses or add new ones from the data base with just one mouse click, thus giving us a structured and flexible solution-first design strategy [18]. More widely, this means that the stakeholders, rather than the engineers, can write, edit and manipulate the use cases and scenarios, in contrast to much current engineering practice. We are not advocating end-user development here, rather that organisations adopt use case and scenario representations and tools that support stakeholder interaction and tailoring, sometimes in place of more complex scenario and simulation approaches.

8.4. Encourage exploration with scenarios

Results from the ATM domain suggests that stakeholders might have explored numerous candidate alternative courses before generating requirements, although more data is needed before we can accept this explanation as plausible. However, we are already extending our Scenario Presenter tool with new features to allow users to zoom in and out on selected alternative courses to make these courses more or less domain-specific at the click of button. The purpose of this feature is to encourage the guided exploration of less domain-specific alternative courses, thus supporting the stakeholder behaviour that we hypothesise took place during alternative course walkthroughs. We are also planning to add cause-consequence associations identified in the naval warfare domain so that users can seek explanations of alternative courses. This also means that use case and scenario walkthroughs should encourage participants to search, perhaps by browsing large number of different scenario pathways and alternatives.

8.5. Combine different walkthrough styles

Finally, the acquisition of functional rather than non-functional requirements, due to the facilitation style and time constraints, suggests that different styles with different stakeholder groups can be effective. CREWS-SAVRE gives us the capability to generate different scenarios with different alternative courses from one use case for different walkthrough participants and purposes. Organisations should adapt their facilitation guidelines and the granularity of their walkthrough prompts to match the participants and types of requirement that they wish to acquire. Rather than have all stakeholders participate in a single walkthrough, undertake several smaller, tailored walkthroughs to acquire requirements systematically from different stakeholder viewpoints. Furthermore, do not think that you will be able to generate or author the right use case or scenario first time. Instead, be prepared to prototype and rework until it is fit for its purpose.

9. Future Work

We are currently implementing the 5 lessons learned ourselves in order to improve the CREWS-SAVRE

process and tool. The Scenario Presenter tool is being enhanced to include end-user tailorability and to allow users to explore alternative courses, as described in the third and fourth lessons.

To implement the first lesson, we are re-examining the scenario generation rules for the ATM domain to understand why only 1 in 5 alternative courses were domain-specific prior to re-testing the algorithm when generating scenarios for Eurocontrol's new Departure Manager system. Based on the fifth lesson, we are developing local scenario generation templates to generate scenarios with different classes of alternative course for different walkthrough participant groups such as usability experts, algorithm designers and air traffic controllers. And finally, to give the stakeholders even greater ownership of the scenarios, we are developing a new version to operate on a wireless PDA device.

At the same time, we are continuing our research and are hoping to underpin these improvements with empirical data from controlled experiments of experienced analysts who will use the Scenario Presenter to discover requirements for an example system. We will combine concurrent and retrospective protocol analyses of the walkthroughs [19] with eye-tracking data that will tell us where on the screen the analyst is looking to provide first-hand data about how and why people recognise alternative courses and requirements. Dependent variables that will be investigated will include the presence and ordering of different domain-independent and domain-specific alternative courses.

Finally, we recognise that not all requirements are likely to arise from the systematic approach supported by CREWS-SAVRE. Rather, scenario walkthroughs also need to encourage creative thinking about possible requirements [20]. In our related CRÈME project, we are investigating whether different scenario prompts that replace the alternative courses can encourage different styles of creative thinking such as combinatorial creativity and analogical reasoning during walkthroughs. We look forward to reporting the results from all of our future work in new publications.

Acknowledgements

The authors wish to thank Eurocontrol and all those involved in the CORA-2 project for their support and involvement in the work reported in this paper. In particular we are grateful to Nina Limbachia, Perminder Sahota and Anja Wever for all of their efforts.

References

1. Weidenhaupt K., Pohl K., Jarke M., Haumer P., 1998, 'Scenario Usage in Systems Development: A Report on Current Practice', IEEE Software, 15(2), 34-45.
2. Jacobson I., Booch G. & Rumbaugh J., 2000, 'The Unified Software Development Process', Addison-Wesley-Longman.
3. Haumer P., Heymans P., Jarke M. & Pohl K., 1999,' Bridging the Gap Between Past and Future in RE: A Scenario-based Approach', Proceedings 4th IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, 66-73.
4. Carroll J.,M., 2000, 'Making Use: Scenario-based Design of Human-Computer Interactions', MIT Press, Cambridge Mass.
5. Rolland C., Souveyet C. & Ben Achour C., 1998, 'Guiding Goal Modelling with Scenarios', IEEE Transactions on Software Engineering, 24(12), 1055-1071.
6. Maiden N.A.M., 1998, 'SAVRE: Scenarios for Acquiring and Validating Requirements', Journal of Automated Software Engineering 5, 419-446.
7. 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.
8. Baddeley, A.D., 1990, 'Human memory: Theory and practice', Lawrence Erlbaum Associates, Hove.
9. Alspaugh T.A., Anton A.I., Barnes T & Mott B.W., 1999, 'An Integrated Scenario Management Strategy', Proceedings 4th IEEE Symposium on Requirements Engineering, IEEE Computer Society Press, 142-149.
10. 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.
11. Maiden N.A.M., Jones S. & Flynn M., 2003, 'Innovative Requirements Techniques Applied to ATM', to appear in Proceedings ATM'2003, Budapest June 23-27 2003.
12. Franch X. & Carvallo J., 2003, 'Using Quality Models in Software Package Selection', IEEE Software Jan/Feb 2003, 34-41.
13. Robertson S. & Robertson J., 1999, 'Mastering the Requirements Process', Addison-Wesley-Longman.
14. Eysenck, M.W. & Keane, M.T., 1995, 'Cognitive Psychology', Psychology Press, Hove, 1995.
15. Lakoff G., 1987, 'Women, Fire and Dangerous Things: What Categories Reveal About the Mind', University of Chicago Press, Chicago.
16. Mijake N., 1986, 'Constructive Interaction and the Iterative Process of Understanding', Cognitive Science 10, 151-177.
17. Rugg, G. & McGeorge, P., 'Laddering', Expert Systems, 12 (4), pp339-346, 1995.
18. Carroll J.M., 2002, 'Scenarios and Design Cognition', Proceedings 10th Joint International Conference on Requirements Engineering, IEEE Computer Society Press, 3-5.
19. Ericsson K.A. & Simon H.A., 1984, 'Protocol Analysis', MIT Press.
20 Maiden N. & Gizikis A., 2001, 'Where Do Requirements Come From?", IEEE Software September/October 2001 18(4), 10-12.