The RESCUE Scenario-Driven Requirements Engineering Process

Neil Maiden
Centre for HCI Design
26th February 2003
Presentation Overview

The RESCUE process and its authors
  – About the Centre
  – Research provenance, practical evaluation
  – RESCUE process characteristics
  – RESCUE process structure and roadmap

Key RESCUE stages and sub-processes
  – Key features of each RESCUE sub-process
  – Rationale for these sub-processes

Lessons learned
  – Successes and lessons learned with CORA-2 project
  – RESCUE process deliverables
  – RESCUE support resources
Centre Basics

University Research Centre
   – Independent department City’s School of Informatics
   – Details from www-hcid.soi.city.ac.uk

Largest HCI-related research group in London
   – 6 full-time academic staff
   – 4 visiting professors and other staff
   – 10 research staff, 8 PhD students, 1 administrator

World-leading expertise in
   – Systems and requirements engineering
   – Usability and accessibility for all, user diversity

3 million € research income since January 2000
What is RESCUE?

RESCUE: The basics

– Requirements with Scenarios for User-Centred Engineering
– A prescriptive process for systematic, scenario-driven requirements engineering in Eurocontrol

What constitutes RESCUE?

– Process guidance: what to do in what order
– Techniques to use to undertake each process
– Software tools to use to support the implementation of these techniques
– Outputs and deliverables from these processes

RESCUE is not exclusive

– Use other techniques and tools to undertake processes
Research Provenance, Practical Evaluation

Origins in basic research
- EU-funded 21903 CREWS project 1996-1999
- Extended in BAE SYSTEMS-funded SERPS project
- Innovative theories and techniques for generating scenarios and walking through them systematically

Widespread industrial application
- With BAE SYSTEMS to produce concept demonstrator within the SERPS project
- With Eurocontrol on extensive implementation and application of requirements engineering process for CORA-2 system
- With Dstl (ex-DERA) to use scenarios to ground requirements trade-off analyses using the $i^*$ approach
Our RESCUE Aims

The Centre’s aims with RESCUE

– To complete a successful basic research to exploitation cycle in scenario-driven requirements engineering
– To understand problems at first-hand when implementing a requirements engineering process
– To gather, analysis and publish data to evaluate the effectiveness of scenario-driven requirements engineering
– To disseminate research results in an effective manner
– To provide a process platform for integrating mature and compatible requirements engineering solutions

Summarise as

– Knowledge transfer and large-scale empirical research into requirements engineering
Your RESCUE Aims

More effective requirements engineering process
  – Greater, more cost-effective stakeholder involvement
    • Participatory techniques that encourage better communication
  – Stakeholder requirements are more complete
    • Ensure coverage of analysis to avoid missing requirements
  – Stakeholder requirements are more correct
    • Ensure precision to acquire the right requirements, and validate these requirements to ensure they are the right ones
  – Stakeholders requirements are more testable
    • Be able to hold contractors to account when delivering systems
  – Focus on important requirements and design issues
    • Concurrent engineering process supporting requirement-design trade-offs
RESCUE’s Key Characteristics

Characteristics to meet process goals

– Structured interaction with stakeholders and informed selection of requirements acquisition techniques - giving greater stakeholder involvement
– Walk through tool-generated scenarios that explore requirements coverage - giving more complete requirements
– Acquire and discover more correct requirements in agreed contexts given through scenarios - giving more correct requirements
– Linking requirements to use case and scenarios and imposing measurable fit criteria for requirements - giving more testable requirements
– Concurrent requirements and design practice - focusing on the most important requirements
What RESCUE Does Not Cover

Requirements activities outside RESCUE include

- **Negotiation** between different stakeholders about requirements in the requirements specification
- Detecting **inconsistencies** between requirements in the requirements specification
- **Safety case** analysis and **hazard** analysis

Possible alternative sources for process guidance

- Requirements negotiation: **EasyWinWin**
- Detecting requirements inconsistencies: **Voila/Voici**
- Safety cases and hazard analysis: Methodologies emerging from **University of York**
A Little More on Scenarios

Scenarios are popular and effective
  – Widespread requirements engineering technique (Weidenhaupt et al. 1998)
  – But generating and using scenarios has been problematic, lacking a tool-supported systematic approach

Systematic scenario-driven requirements process
  – Researched and evaluated in Centre since 1996

Advantages to requirements engineers
  – Structured requirements acquisition
  – Testable requirements in context
  – Reuse domain expertise
  – More effective communication
Human activity modelling

i* system modelling

Use case & scenario analysis

Requirement management

Managed interaction between the streams

RESCUE
Concurrent Systems Engineering
Why Four RESCUE Streams?

Why human activity modelling?
- Most ATM systems are socio-technical systems - people are as critical to system functioning as machines
- Social systems are overlooked source of requirements
- Stakeholder requirements advocate change which might negatively impact on working practices

Why $i^*$ system modelling?
- Establish system scope, structure and important goals
- ATM systems are inter-dependent socio-technical systems - model dependencies explicitly
- Independent cross-check with use case modelling
Why Four RESCUE Streams?

Why scenario-driven walkthroughs with use cases?
– Scenarios are an effective and cost-effective technique for acquiring testable stakeholder requirements
– Use case descriptions enable specifications from which we generate scenarios in CREWS-SAVRE
– Use case models can be automatically derived from $i^*$ models

Why requirements management?
– Store and manipulate large numbers of stakeholder requirements over a long project period
– Impose Quality Gatekeeper role with technologies needed to support it
– Baseline requirements engineering practices (e.g. writing testable requirements) on which to work
Why Four Concurrent RESCUE Streams?

- Human activity modelling
- i* system modelling
- Use case & scenario analysis
- Requirement management

Streams:
- Actors, goals and tasks
- Actors, goals, tasks and system boundaries
- Requirements and constraints
- Requirements
- Changes
FIRST SYNCHRONISATION STAGE

SECOND SYNCHRONISATION STAGE

THIRD SYNCHRONISATION STAGE

FOURTH SYNCHRONISATION STAGE

MAKE DECISIONS

MAKE DECISIONS

MAKE DECISIONS

MAKE DECISIONS

RESCUE SYNCHRONISATION POINTS
Why Synchronisation Stages?

Important decision-making stages in RESCUE

- Encourage **explicit decision-making** and documentation in the requirements process
- Provide **solid foundations** for the next stages in activity modelling, system modelling, scenario analysis and requirements documentation
- Decide and sign off **agreement** about the socio-technical system **boundaries**
- Reject requirements that cannot be satisfied by available technologies and changes in work practice
- Reject technologies, designs and ideas that do not meet stakeholder requirements
- Decide and **sign-off** deliverables from the 4 streams
QuickTime™ and a H.263 decompressor are needed to see this picture.

“Drill down into the minutiae, then come back up - it bounces”
The Overall RESCUE Process

**First synchronisation stage**
- Gather data on human processes
  - Descriptions of human activities
- Model human activity
  - Human activity model

**Second synchronisation stage**
- Determine system boundaries
  - i* context model
- Determine system dependencies, goals and rationale
  - i* SD and SR models
- Refine system dependencies, goals and rationale
  - Refined i* SD and SR models

**Third synchronisation stage**
- Develop use case model
  - Creative design workshops
- Describe use cases
  - Use cases
- Specify use cases
  - Use case specifications
- Walkthrough scenarios
  - Scenarios associated with requirements

**Fourth synchronisation stage**
- Define system-level requirements
- Define and document requirements
  - System-level requirements
- Define and document requirements
  - System-level requirements and associated use cases
- Define and change requirements
  - Scenarios associated with requirements
- Impact analysis
  - Refine and change requirements
- Use cases
  - Use case descriptions
- Use case specifications and associated requirements
Why each step in RESCUE?

- Gather data on human processes
- Model system boundaries in terms of external actors
- Determine system boundaries in terms of dependencies with actors
- Baseline system with requirements and evidence
- Model human activity
- Describe required system behaviour
- Specify use cases to enable automatic scenario generation
- Discover requirements in innovation
- Assess changes to discover +/-ve impacts of requirements
- Refine and change requirements
- Manage large numbers of changing requirements effectively
- System-level requirements
- Define system-level requirements
- Define and document requirements
- Requirements
- Define and document requirements
- Use case specifications and associated requirements
- Use use case specifications and associated use cases
- Requirements
- Define and document requirements
- Impact analysis
- Scenarios associated with requirements
- Refine and change requirements
- Walkthrough scenarios
- Discover requirements in synchronisation stage
- Use case specifications
- Define and document requirements
- Use case model and summaries
- Use case model
- Develop use case model
- Refine system boundaries, goals and rationale
- SD and SR models
- Refined SD and SR models
- Creative design workshops
- Define system-level requirements
- Define and document requirements
- System-level requirements
Other Sub-Processes (ACRE)

Purpose
– To raise awareness of acquisition techniques available
– To aid technique selection and use

Scope
– Communication between stakeholders
– Not stakeholder analysis

Audience
– Practitioners who do acquisition

Source
RESCUE: The First Stage
Establishing System Boundaries
RESCUE Boundary Stage

RESCUE first-stage processes

- Gathering data about human activities
- ACRE: Acquiring stakeholder requirements
- $i^*$ system context modelling
- Use case modelling
- High-level system requirements
- Formal requirement inspections
RESCUE’s Boundaries Stage
Gather Data on Human Processes

First synchronisation stage

- Gather data on human processes
  - Descriptions of human activities
- Determine system boundaries
  - i* context model
- Model human activity
  - Human activity model
  - i* SD and SR models

Second synchronisation stage

- Determine system dependencies, goals and rationale
  - Extended use cases
- Creative design workshops
- Use cases
  - Use case model and summaries
  - Requirements
  - System-level requirements

Third synchronisation stage

- Refine system dependencies, goals and rationale
  - Refined i* SD and SR models
- Describe use cases
- Specify use cases
  - Use case descriptions
  - Use case specifications
  - System-level requirements and associated use cases

Fourth synchronisation stage

- Walkthrough scenarios
  - Use case specifications and associated requirements
  - Scenarios associated with requirements
- Define and document requirements
- Define system-level requirements
- Define and document requirements
- System-level requirements

Extended use cases

Impact analysis

Refine and change requirements

System-level requirements

Use case specifications

Creative design workshops

Use cases

Use case model and summaries

Requirements

Gather data on human processes

Determine system boundaries

Determine system dependencies, goals and rationale

Develop use case model

Define system-level requirements

Define system-level requirements

Define and document requirements

Define and document requirements

Use case specifications

Requirements
Gathering Data for Activity Modelling

• Usually need a combination of different techniques
  – Different techniques tend to elicit different kinds of data
  – Data gathered using one technique may be used to confirm inferences made on the basis of data from another technique

• Focus on identifying non-prescribed elements and understanding less obvious actions
  – E.g non-direct communication (watching and listening), looking over someone’s shoulder can be important in achieving overall goals
  – Often identified through inference

• Do not pre-judge relevance of information
  – Inherent non-determinism of complex and open socio-technical systems means that any data may be relevant

• Identify constraints and sources of variability
  – Environment and context constrain, subjective factors lead to inter- and intra-controller variability
Observation

Analyst observes the work domain and activities
- Records findings using notes, audio and video recording
- Done in the laboratory or in the field

Use the data for
- Gathering information about activities which have observable behaviour or stages (not cognitive activities)
- Gathering information about activities that involve many individual steps which may be omitted in a verbal description
- Gathering information when a task is difficult to verbalise

Limitations include
- Time-consuming to do
- Involves inference, and expertise in interpreting the data
Interviews

Obtain information difficult to elicit in other ways
- For example general principles, background knowledge, reasons behind behaviour, infrequent events

Limitations
- Time-consuming to conduct, transcribe and analyse
- Danger of analyst bias towards own interests, knowledge and beliefs

Three different types of interview
- Structured: pre-determined questions, fixed order
- Semi-structured: questions determined in advance but re-ordered, explained, elaborated or omitted
- Unstructured: no pre-determined questions, interviewer has area of interest but conversation develops freely
Use of Verbal Protocols

Verbal protocols are reports given by a person

- Collected either during an activity (concurrent protocol) or after (retrospective protocol)
- Collected alongside observational data

Useful for

- Collecting detailed information on many aspects of task

Limitations

- Time-consuming as protocols must be transcribed and carefully coded
- Verbalising and thinking about a task can change its nature
- Subjects may be continually revising their ideas about how they carry out an activity
Use of Verbal Protocols Continued

Concurrent protocols
- Require the subject to report what they are doing while they are doing it
- Relies on working memory and are susceptible to working memory capacity limitations, interference (from other cognitive processing) and decay (forgetting)
- Interferes with task performance (helps or hinders)
- Focuses more on physical actions

Retrospective protocols
- Require the subject to report what they did after task
- Require subject to remember what they did, but can be supported by video of task performance
- Better for explanations of cognitive aspects of tasks
- More time-consuming
Ethnography

Potential role in RESCUE

- Origins in anthropology; literally ‘writing the culture’
- Used in social sciences to understand the social organisation of activities, and hence work
- Aims to find the order in an activity rather than impose frameworks
- Users are observed doing everyday activities
- Observers immerse themselves in the user’s environment, participate in work, go to meetings, join in conversations, read documents etc
- Make the implicit explicit (non-prescribed elements)
- Users may be so familiar with their environment and tasks that they don’t see their importance
Example Data from Activity Modelling

Typical data that is acquired from an actor

- **Actor**: controller 5
- **Triggering problem**: Catch on SLR3519 and BRA935
- **Goals**: Negotiate with next sector who solves the catch up (*non-prescribed*), verify aircraft capabilities to climb higher
- **Resources**: Co-ordination
- **Resource management strategies**: Negotiate load sharing
- **Constraints**: Destination (BRA), aircraft performance
- **Actions**: Co-ordinate radar transfer, descend BRA, climbing one of the two
More Example Data

Typical data that is acquired from another actor

- **Actor**: controller 3
- **Triggering problem**: SAB874, SAB888, same exiting point
- **Goals**: Do not deal with the problem or deal with the problem
- **Resources**: Time to closest point (enough for delegation)
- **Resource management strategies**: Delegate solution to next sector, eliminate uncertainties as soon as possible
- **Actions**: Assign low altitude to climbing aircraft
Inputs To Other Sub-Processes

- Develops understanding of **system boundaries**
  - what may be automated, what is to remain under the control of the controllers (e.g. handling exceptional conditions)
- Identifies **non-prescribed** as well as prescribed elements of the system
  - non-prescribed goals, actors, resources and resource management strategies may make the system more flexible or increase redundancy, therefore increasing safety
- Inspires high-level **design visions**
- Provides **contexts** for scenario walkthroughs
- **Source of data** for $i^*$ system modelling
- Identifies data for **fit criteria** for requirements
ACRE: Acquiring Requirements

First synchronisation stage
- Gather data on human processes
- Determine system boundaries
- Model human activity
  - Descriptions of human activities
  - Human activity model
  - i* context model

Second synchronisation stage
- Determine system dependencies, goals and rationale
- Refine system dependencies, goals and rationale
- Use cases
  - Extended use cases
  - Use case descriptions
  - Use case specifications
  - Requirements
  - System-level requirements and associated use cases
  - i* SD and SR models

Third synchronisation stage
- Describe use cases
- Specify use cases
- Define and document requirements
  - Use case specifications and associated requirements
  - Use case model and summaries
  - Use case model and summaries
  - System-level requirements
  - System-level requirements

Fourth synchronisation stage
- Creative design workshops
- Define system-level requirements
  - System-level requirements
  - Define system-level requirements
  - Define system-level requirements
- Scenarios associated with requirements
- Impact analysis
- Refine and change requirements
- Walkthrough scenarios
  - Scenarios associated with requirements

- Use case descriptions
- Define and document requirements
  - System-level requirements
  - Refined i* SD and SR models
- Refine system dependencies, goals and rationale
  - Refined i* SD and SR models
  - Refined i* SD and SR models

System-level requirements
- Define use case model
- Develop use case model
- System-level requirements
- System-level requirements
- System-level requirements
- System-level requirements
- System-level requirements
- System-level requirements
- System-level requirements
Acquiring Stakeholder Requirements

First phase of requirements engineering process

Two basic approaches

– Mining requirements out of documents
– Face-to-face communication between requirements engineers and stakeholders

acquisition, negotiation & agreement

system requirements, problem domain facts
From Elicitation to Negotiation

Changes to requirements acquisition practice

Implications
- Wider range of available techniques than assumed
- Interleaves with modelling and negotiation activities
Diverse Sources of Requirements

Requirements come from lots of different places...

- Policies, aims, objectives: high-level goal statements
  - e.g. “send a man to the moon”
- Problems which initiate requirements to change an existing, malfunctioning system
- Examples of systems observed by stakeholders
  - e.g. “I want one like that…”
- Expert recommendation of system enhancement
  - e.g. changes through technological enhancement
- Imposed from the outside world
  - e.g. changes in legislation, regulations, etc

These sources are often overlooked
- So look in lots of different places for requirements
ACRE’s Dimensions and Techniques

6 dimensions

12 basic methods

Observation
Unstructured/structured interviews
Protocol analysis
Card sorting
Laddering
Repertory grids
Brainstorming
Rapid prototyping
Scenario analysis
RAD/JAD
Ethnographic studies

Purpose of requirements
Types of knowledge
Acquisition context
Observable phenomena
Internal filtering of knowledge
Method interdependencies
ACRE’s 6 Dimensions

1) Purpose of the requirements
   – Bespoke, procured or off-the-shelf systems

2) Knowledge type
   – Acquire behaviour, process or data knowledge

3) Internal filtering of knowledge
   – Access to future system or current domain knowledge

4) Observable phenomena

5) Acquisition context
   – Number of stakeholders, time available, and environment

6) Technique interdependencies
Determine System Boundaries

First synchronisation stage:
- Gather data on human processes
- Determine system boundaries
- Model human activity
- Use cases

Second synchronisation stage:
- Descriptions of human activities
- Model human activity
- Use case descriptions
- SD and SR models

Third synchronisation stage:
- Develop use case model
- Describe use cases
- Specify use cases
- Use case specifications
- Use case model and summaries

Fourth synchronisation stage:
- System-level requirements
- Define and document requirements
- Use case specifications and associated requirements
- Use case model and summaries
- Extended use cases
- Requirements

Impact analysis
- Scenarios associated with requirements
- Refine and change requirements
- Refined i* SD and SR models

系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Creative design workshops
- i* context model
- i* context model
- i* context model

Walkthrough scenarios
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Requirements
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Use case descriptions
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Use case specifications
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Use case model and summaries
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Extended use cases
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Use cases
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

i* context model
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Determine system boundaries
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Gather data on human processes
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求

Model human activity
- 系统级要求
- 定义和文档要求
- 需求规格说明书
- 扩展用例
- 要求
i* Context System Modelling

Simple system scoping
- Use context data flow diagrams (DFDs) to indicate system boundary or system boundaries
- Model states what systems and actors are outside the system or interest
- Draw several system boundaries to indicate the different social, socio-technical and technical systems, producing a simple onion model

Adjacent systems
- Systems that supply the work (products or systems) with information, or receive information and services from the work (Robertson & Robertson 1999)
- Useful for thinking about actors and their dependencies
Example Context Model for CORA-2

CORA-2

- Flight data processor
- Environment data base
- Trajectory predictor
- Conflict detector
- Systems co-ordinator
- CORA-1, PAC, TED

Departure manager
En-route manager
Arrival manager
Develop Use Case Model

First synchronisation stage
- Gather data on human processes
- Determine system boundaries
- Develop use case model
- Define system-level requirements

Second synchronisation stage
- Model human activity
- Determine system dependencies, goals and rationale
- Use cases
- Use case model and summaries
- Define and document requirements
- System-level requirements

Third synchronisation stage
- Refine system dependencies, goals and rationale
- Extended use cases
- Describe use cases
- Define and document requirements
- Use case descriptions
- System-level requirements and associated use cases

Fourth synchronisation stage
- Refined SD and SR models
- Use case specifications
- Use case specifications and associated requirements
- Walkthrough scenarios
- Define and document requirements
- Scenarios associated with requirements
- Impact analysis
- Refine and change requirements
Use Case Modelling

From the UML, use it as is….

– An actor is a type of anything external to the system, human or machine
– A single person or system can instantiate several different actors by playing different roles
– Actors come in two kinds:
  • Primary actors, using system in daily activities
  • Secondary actors, enabling primary actors to use system
Use Cases

Use cases
- Specification of system behaviour to undertake a major system goal or service
- Focus on high-level goals, ignore associations
- Structures the system in terms of services and goals

The use case model
- Model shows the collection of use cases
- Characterises the behaviour of whole system
Define System-Level Requirements
Managing Requirements with VOLERE

Underpin process with leading-practice processes
- Adopt practice from Atlantic System Guild’s VOLERE requirements process
- Implement VOLERE requirements shell in Rational’s Requisite Pro requirements management tool
First synchronisation stage

- Gather data on human processes
- Determine system boundaries
- Develop use case model
- Define system-level requirements

Second synchronisation stage

- Model human activity
- Determine system dependencies, goals and rationale
- Describe use cases
- Define and document requirements

Third synchronisation stage

- Refine system dependencies, goals and rationale
- Specify use cases
- Define and document requirements
- Use case descriptions

Fourth synchronisation stage

- Impact analysis
- Refine and change requirements
- Scenarios associated with requirements
- Define and document requirements
- Use case specifications and associated requirements
- Walkthrough scenarios

Descriptions of human activities
i* context model
i* SD and SR models
Use cases
Use case model and summaries
System-level requirements
System-level requirements and associated use cases
Use case specifications
Extended use cases
Refined i* SD and SR models

First Synchronisation Stage
First Synchronisation Stage

Formally examine outcomes from first stage

- Cross-check the activity modelling data, context models, use case models, stakeholder requirements in VOLERE data base for completeness and correctness
- External actors in context models are same as actors in use case models
- Actors identified during activity modelling are same as actors in use case models
- Services and functions related to use cases in use case models map to functional requirements in VOLERE data base
- Produce set of between-model mappings to deliver traceability and offer integrated view of new system and its boundaries
Requirement Inspections

Ensure quality of requirements

- Impose VOLERE Quality Gateway on all requirements entering the requirements data base
- Formal inspection meetings (Sommerville & Sawyer 1997) on the resulting requirements documents and other models

Inspection meetings

- Cost-effective approach that can avoid stakeholder confrontations
- Make decisions to resolve problems about system boundaries in the form of published actions
- Use Quality Gateway checklist (Robertson & Robertson 1999) to drive inspections
- Possible to review up to 20 requirements per hour
RESCUE: The Second Stage
Allocating Work
RESCUE Work Allocation Stage

RESCUE second-stage processes
- Analyse data about human activities
- $i^*$ system Strategic Dependency modelling
- First-cut $i^*$ system Strategic Rationale modelling
- Use case descriptions
- Stakeholder requirements
- Formal inspections of requirements and other models
RESCUE’s Work Allocation Stage
**Model Human Activity**

**Centre for HCI Design**

**First synchronisation stage**
- **Gather data on human processes**
- **Determine system boundaries**

**Second synchronisation stage**
- **Model human activity**
- **Determine system dependencies, goals and rationale**

**Third synchronisation stage**
- **Refine system dependencies, goals and rationale**
- **Specify use cases**
- **Use case descriptions**

**Fourth synchronisation stage**
- **Walkthrough scenarios**
- **Define and document requirements**
- **Impact analysis**
- **Refine and change requirements**
Analysis and Modelling

• Allow requirements to emerge from analyst’s understanding of work domain and human activity
  – not all requirements are just ‘out there’ waiting to be collected
• Identify and include non-prescribed components
• Model all possible variations in human activity
  – don’t try to prescribe the ‘right’ sequence of actions
• Link variations to sources of variability and subjective factors
  – both inter- and intra-controller differences in performance may be due to different trade-off criteria, attitudes to risk, trust in own knowledge, trust in other controllers, belief that pilots will behave as expected, etc
The Resulting Activity Models

Activity models contain important concept types

- **Goals** expressing the desired states of future system
  - High-level functional and local goals, individual and collective goals, prescribed and non-prescribed goals
- **Human actors** describing all people involved in system
- **Resources** that are means available to actors to achieve their goals
- **Resource management strategies** expressing how actors achieve their goals with the resources
- **Constraints** that are the environmental properties which affect decisions
- **Actions** undertaken by actors to solve problems or achieve goals
- **Contextual features** - that influence decision-making
Example Activity Model Goals

From the air traffic control domain

– High-level functional goals
  • Monitor the traffic, maintain aircraft conformance anticipate future conflicts, anticipate opportunities for traffic improvement and better use of the airspace, find solutions that satisfy the highest number of constraints

– Individual and collective (prescribed and non-prescribed) goals
  • Maintain a good understanding of the situation, keep workload at an acceptable level, do not forget anything important, be cooperative

– Local (prescribed and non-prescribed) goals
  • Confirm conflicts between aircraft, allocate conflict solution, detect conflict in the next sector, resolve conflict
Example Activity Model Actors/Resources

Actors, resources and management strategies

– **Actors**
  • Radar controller, planner controller, controllers from other sectors and centres, cockpit

– **Resources (prescribed and non-prescribed)**
  • Default knowledge, co-operation, communication, procedures, tools, time, pilot capabilities/states/intentions

– **Resource Management Strategies (prescribed and non-prescribed)**
  • Workload regulation, time horizon, risk assessment (e.g., probability of making a wrong anticipation), co-operation and communication
Example Activity Model Constraints/Actions

Constraints, actions and traffic context

– **Constraints**
  - Aircraft type/capabilities/status, letters of agreement (LoAs), top of descent (ToD), preferred flight level/route, flight origin and destination

– **Actions**
  - Co-ordination of flight level and heading, flight level assignment, heading assignment, communication of information to the next sector

– **Traffic context**
  - Traffic density, aircraft exiting on the same beacon, conflict originating in the present sector
Determine system dependencies, goals and rationale
Modelling the System with $i^*$

Model system goals, actors and boundaries
- Exploit $i^*$ approach from University of Toronto
- Syntax and semantics for modelling complex types of associations between requirements
- Particular emphasis on systems dependencies and trade-offs between non-functional requirements

Support $i^*$ with REDEPEND prototype
- In-house software tool developed with MS-Visio
- Modelling check and propagation mechanisms
- Integration with AHP multi-criteria decision-making tool

Use REDEPEND to develop 2 $i^*$ model types
- Actor dependencies and their goal-rationale structures
i* Strategic Dependency Model

Strategic Dependency model
- Model dependencies between strategic actors
- Actors include the new system, actors that are influenced by new system, and actors that constrain the design of the new system

Network of dependency relationships among actors
- Depender who is the actor who “wants” something
- Dependee who has the “ability” to provide something
- Dependencies expressed in terms of important requirements, tasks and resources

Provides important advantages
- Scopes the new system
- Discover important new requirements
Example Strategic Dependency Model
For Internet Airline Ticketing System

- Maximise revenues
- Income
- Ticket reference
- Purchase quickly
- Tickets purchased
- Maximise new custom
- Reliable information
- Airline
- Website
- Credit check system
- Credit rating
- Check credit card
CORA-2: Strategic Dependency Model

Flight Data Processing System

Conflict Detector

Trajectory Editor

Clearance Assistant

Sequencing tools

Environment database

System Co-ordinator

CONTROLLER using cora 2

CORA-2: Strategic Dependency Model
i* Strategic Rationale Model

Describes desirable actor processes

- In terms of goals, tasks, resources and soft goals
- Actors accomplish goals and tasks
  - SR model specifies what actors accomplish themselves
  - Adding SD model specifies what need other actors to accomplish
- Models rationale, decompositions and contribution links between important requirements of actors
- Innovative modelling of high-level requirements
- Contribution links between requirements suggest important trade-offs to make early on
Strategic Rationale Model
For Internet Airline Ticketing System

- Passenger
- Purchase quickly
- Purchase cheaply
- Destination selected
- Select flight(s)
- Online purchase
- Tickets purchased
- Go travel agent
- Credit card
- Online purchase
The REDEPEND Prototype
Version 2 delivered in July 2002
Describe use cases
Use Case Authoring: CREWS-ECRITOIRE

Specify system behaviour with use cases

- Traditional UML use case modelling, informed with the task models from the i* modelling
- Well-grounded use case authoring with CREWS-ECRITOIRE method using case frame grammars (Rolland et al. 1998)
- ATM-specific use case language in terms of lexicon of CORA-2 agents, objects and actions to use
- Structured with CORONET use case templates

Delivers use case specifications

- More effective use case analysis
- Direct inputs into CREWS-SAVRE scenario generation and walkthrough tools
## Use Case Specification Template

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Unique ID for Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Name of author</td>
</tr>
<tr>
<td>Date</td>
<td>Date Use Case was written</td>
</tr>
<tr>
<td>Source</td>
<td>Source of Use Case</td>
</tr>
<tr>
<td>Actors</td>
<td>Actors involved in Use Case (from the Use Case Model)</td>
</tr>
<tr>
<td>Problem statement (now)</td>
<td>Description of current problem</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pecis</th>
<th>Informal scenario description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Requirement that the use case DECOMPOSES</td>
</tr>
<tr>
<td>Constraints</td>
<td>Requirements that impose CONSTRAINTS on the required behaviour in the use case</td>
</tr>
<tr>
<td>Added Value</td>
<td>Benefit of Use Case above and beyond the original scenario from the original system</td>
</tr>
<tr>
<td>Justification</td>
<td>Why is the Use Case needed?</td>
</tr>
<tr>
<td>Triggering event</td>
<td>Event or events that can trigger the Use Case</td>
</tr>
<tr>
<td>Preconditions</td>
<td>Necessary conditions for the Use Case to occur</td>
</tr>
<tr>
<td>Assumptions</td>
<td>Explicit statement of any assumptions made in writing the Use Case</td>
</tr>
<tr>
<td>Successful end states</td>
<td>Successful outcome(s) of the Use Case</td>
</tr>
<tr>
<td>Unsuccessful end states</td>
<td>Unsuccessful outcome(s) of the Use Case</td>
</tr>
</tbody>
</table>

### Normal Course

1. **Event 1**
   - [EN] System requirement ENABLES Event 1
   - [CO] System requirement CONSTRAINTS Event 1

2. **Event 2**
   - [EN] System requirements ENABLES to Event 2
   - [CO] System requirement CONSTRAINTS Event 2

   ... 

### Variations

1. If [condition] then [variation statement] (related to Event 1)

   ... 

### Alternatives

1. If [condition] then [alternative course statement] (related to Event 1)

   ...
Define and Document Requirements
Integrating Requirements and Use Cases

Formalise use case-requirements association
- Implement meta-model in Requisite Pro requirements management software tool
- Management baseline for precise, testable requirements

- Enables precise traceability for each requirement derived through scenario analysis
**Make All Requirements Testable**

Testable requirements are essential

- Unambiguous requirements as each stakeholder requirement description is expanded into logical, quantifiable test of the requirement
- Strong requirement taxonomies guide definition of these measurable fit criteria
  - Advise types of units of measure for each requirement type
  - *Usability-type requirements* measured in terms of *user-error frequency, task completion times and frequency of use*
- Each requirement associated to one or more use cases and scenarios
  - Provides event and usage context for scoping the fit criterion, thus making the requirement even more precise

Testable requirements control acceptance testing
Creative Workshops

Recognise concurrent systems engineering
  – High-level design decisions are inevitable during the requirements engineering process
  – Provoke creative thinking about requirements to push new technologies and opportunities
  – Enable more effective use case authoring based on high-level decisions about system boundaries

Exploit theories to provoke creative thinking
  – Exploratory creativity through open-ended analogies
  – Transformational creativity with solution domain experts transforming initial ideas into concrete requirements
  – Combinatorial creativity to formulate new requirements from existing requirements and design ideas
Examples from Creative Workshops

Extreme scenario collages

Designs from other domains

Inventive group working and role-playing
Second Synchronisation Stage

First synchronisation stage:
- Gather data on human processes
- Determine system boundaries
- Develop use case model
- Define system-level requirements
- Use case model and summaries
- System-level requirements
- SD and SR models
- Extended use cases
- Use cases
- Requirements

Second synchronisation stage:
- Model human activity
- Determine system dependencies, goals and rationale
- Describe use cases
- Define and document requirements
- Use case descriptions
- System-level requirements and associated use cases
- i* context model
- i* context model

Third synchronisation stage:
- Refine system dependencies, goals and rationale
- Specify use cases
- Define and document requirements
- Use case specifications
- Use case specifications and associated requirements
- Refined i* SD and SR models
- Extended use cases
- Use cases

Fourth synchronisation stage:
- Walkthrough scenarios
- Define and document requirements
- Scenarios associated with requirements
- Use cases
- Impact analysis
- Refine and change requirements
- System-level requirements

Centre for HCI Design
Second Synchronisation Stage

Formally examine outcomes from second stage

- Cross-check the activity models, \( i^* \) SD and SR models, use case descriptions, stakeholder requirements in VOLERE data base for completeness and correctness
- Use cases in use case models are the same as tasks in \( i^* \) SR models
- Use case descriptions are consistent with descriptions of activities in human activity models
- External actors in \( i^* \) SD models are same as actors in use case models and descriptions
- Produce set of between-model mappings to deliver traceability and offer integrated view of new system
Requirement Inspections… Again

Ensure quality of requirements

- Impose VOLERE Quality Gateway on all requirements entering the requirements data base
- Formal inspection meetings (Sommerville & Sawyer 1997) on the resulting requirements documents and other models

Inspection meetings

- Cost-effective approach that can avoid stakeholder confrontations
- Make decisions to resolve problems about work allocations in the form of published actions
- Use Quality Gateway checklist (Robertson & Robertson 1999) to drive inspections
RESCUE: The Generation Stage

Centre for HCI Design
RESCUE Generation Stage

RESCUE third-stage processes
– Continue $i^*$ system Strategic Rationale modelling
– Use case specifications
– Stakeholder requirements
– Formal inspections of requirements and other models

A shorter stage than the previous two
– Preparing for the fourth stage
RESCUE’s Third Stage

First synchronisation stage:
- Gather data on human processes
- Determine system boundaries
- Develop use case model
- Define system-level requirements
- Use cases and summaries
- SD and SR models
- Creative design workshops
- Describe use cases
- Define and document requirements
- Use case descriptions
- System-level requirements and associated use cases
- Extended use cases
- Requirements

Second synchronisation stage:
- Model human activity
- Determine system dependencies, goals and rationale
- Refined SD and SR models
- Use case specifications
- Use case specifications and associated requirements
- Refined i* context model
- i* context model
- Human activity model
- Use cases
- Scenarios associated with requirements
- System-level requirements

Third synchronisation stage:
- Refine system dependencies, goals and rationale
- Specify use cases
- Define and document requirements
- Use case specifications
- Refined i* SD and SR models
- Use case model and summaries
- Requirements

Fourth synchronisation stage:
- Scenarios associated with requirements
- Impact analysis
- Refine and change requirements
- System-level requirements
- Walkthrough scenarios
- Define and document requirements
- Use case specifications
- Refined i* SD and SR models
- Use cases and summaries
- SD and SR models
- Creative design workshops
- Describe use cases
- Define and document requirements
- Use case descriptions
- System-level requirements and associated use cases
- Extended use cases
- Requirements
Refine System Dependencies, Goals and Rationale
Integrate $i^*$ SD and SR Models

Connect actor SR models using dependencies
- Include each SD model dependency between actors or their elements in the SR model
- Mechanical process if SD and SR models are effectively cross-checked
- IF actor has no SR model THEN link dependency to the actor
Producing a Single SR Model

Integrate partial SR models
- Connect using all dependency links from SD model
- Add additional dependency links that emerge from the richer SR models
- Check the model for correctness, completeness and consistency
Strategic Rationale Model
For Internet Airline Ticketing System

- **Airline**
  - Maximise revenues
  - Repeat business
  - Income

- **Passenger**
  - Purchase quickly
  - Purchase cheaply
  - Tickets purchased

- **Online purchase**
  - Go travel agent
  - Select flight(s)
  - Credit card
  - Destination selected

- **Repeat business**
- **Maximise revenues**
Specify use cases
Specifying Use Cases

Enables automatic scenario generation

- Deliver important **productivity benefits** with CREWS-SAVRE module
- **Inputs** are use case descriptions written according to authoring guidelines
- **Outputs** are use case specifications and set of scenarios generated from these specifications

Key activities

- **Refining** the use case to ensure atomic actions
- **Typing** all use case actions and actors
- Setting scenario generation **parameters**, with focus on the alternative course generation
- **Testing** the generated scenarios with pilot runs

Undertaken in conjunction with Centre staff
Example Use Case Specification

CREWS-SAVRE Use case

The use case has been revised, structured and typed for scenario generation in CREWS-SAVRE. It is a simple use case and hence scenario, effective to test the walkthrough processes.

CREWS-SAVRE Types

The use case identifies one agent and its type.

- CORA 2
- Machine

CREWS-SAVRE Use Case

The use case is structured. Some action descriptions have been rewritten. Each action is typed. Each agent in each action is identified. The CORA verb from the action language is identified where required.

<table>
<thead>
<tr>
<th>ID</th>
<th>Action</th>
<th>Type</th>
<th>Agent-1</th>
<th>Agent-2</th>
<th>CORA verb</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The CORA 2 SYSTEM updates the reminder points for action on the changed best one resolution.</td>
<td>Compute</td>
<td>CORA 2</td>
<td></td>
<td>Display</td>
</tr>
<tr>
<td>2</td>
<td>The CORA 2 SYSTEM updates the indication display with the new ranked resolutions.</td>
<td>Compute</td>
<td>CORA 2</td>
<td></td>
<td>Display</td>
</tr>
</tbody>
</table>

CREWS-SAVRE Parameters

The following exception types must be switched on in the use case modeller:

- GAc1, 10, 11, 12, 2, 3, 4, 5, 6, 9
- GAg1, 2, 3
- GEv1, 2, 3, 4, 5, 6, 7
- GOb1, 2, 3, 4
- P1, P4, P5
- All PE, except PE1,2,2, PE1,2,3 and PE8
- All EC, exceptions: please

These exceptions were narrowed down after a walkthrough of the first scenario with expertise from Eurocontrol staff.
Resulting CREWS-SAVRE Scenarios

CREWS-SAVRE

- Automatic scenario generation from use case specifications (Sutcliffe et al. 1998, Maiden 1998)
- Specialisation of scenario generation and walkthroughs to the air traffic control domain

- Non-standard aircraft performance
- Unexpectedly high performance
- Unexpectedly low performance
- High performance military aircraft

138 exception classes
Effective reuse of domain expertise
First synchronisation stage

- Gather data on human processes
- Determine system boundaries

Second synchronisation stage

- Model human activity
- Determine system dependencies, goals and rationale

Third synchronisation stage

- Refine system dependencies, goals and rationale
- Describe use cases

Fourth synchronisation stage

- Walkthrough scenarios
- Impact analysis

- Develop use case model
- Use cases

- Define system-level requirements
- System-level requirements

- Use case model and summaries
- Use case descriptions

- Define and document requirements
- System-level requirements and associated use cases

- Define use case model
- Extended use cases

- Use case specifications
- Use case specifications and associated requirements

- Define and document requirements
- Scenarios associated with requirements

- Refine system dependencies, goals and rationale
- Refined i* SD and SR models

- Describe use cases
- Use case model and summaries

- System-level requirements
- Requirements

- Creative design workshops
- i* context model

- Use cases
- i* context model

- Requirements
- Descriptions of human activities

- Human activity model
- i* SD and SR models

- System-level requirements
- Use case specifications

- Extended use cases
- Refined i* SD and SR models

- Use case model
- i* context model

- First synchronisation stage

- Second synchronisation stage

- Third synchronisation stage

- Fourth synchronisation stage
Third Synchronisation Stage

Formally examine outcomes from third stage

- Cross-check the use case specifications and $i^*$ SR models for completeness and correctness
- Establish and agree actor types based on high-level design decisions
- Establish and agree system boundaries
- Involvement of different actors in use cases
- Produce set of between-model mappings to deliver traceability and offer integrated view of new system
Requirement Inspections… Again

Ensure quality of requirements

– Impose VOLERE Quality Gateway on all requirements entering the requirements data base
– Formal inspection meetings (Sommerville & Sawyer 1997) on the resulting requirements documents and other models

Inspection meetings

– Cost-effective approach that can avoid stakeholder confrontations
– Make decisions to resolve problems about generated use cases, scenarios and i* SR models in the form of published actions
– Use Quality Gateway checklist (Robertson & Robertson 1999) to drive inspections
RESCUE: The Fourth Stage
Coverage using Scenario Walkthroughs
RESCKUE’s Fourth Stage

First synchronisation stage
- Gather data on human processes
- Determine system boundaries
  - *context model
- Model human activity
- Gather data on human activities
  - *context model

Second synchronisation stage
- Determine system dependencies, goals and rationale
  - Extended use cases
- Describe use cases
- Use cases
- Use case model and summaries
  - Use case descriptions

Third synchronisation stage
- Specify use cases
- Use case specifications
  - Use case specifications and associated requirements
- Define system-level requirements
- Define and document requirements
  - System-level requirements and associated use cases
- Define and document requirements
- Use case specifications

Fourth synchronisation stage
- Develop use case model
- Use cases
  - Requirements
  - Scenarios associated with requirements
- Define and document requirements
- Scenarios associated with requirements
- Impact analysis
  - Refine and change requirements
- Refine system dependencies, goals and rationale
  - Refined *SD and SR models
- System-level requirements and associated use cases
CREWS-SAVRE Scenario Walkthroughs

Scenario walkthrough workshops
- Three-hour structured walkthroughs with scenarios
- Explore normal course then alternative courses

1. Is event relevant?
2. Is event handled?
3. Add requirement

Traceable requirements - to scenario, event and alternative course
Three Different Scenario Representations

1. Use case descriptions
   - Effective for simple normal course walkthroughs
   - Paper-based, no bespoke tool support

2. Scenarios in ExcelPresenter
   - Excel-based scenario presentations
   - Simple to use, self-contained scenarios
   - Limited functionality, usability can be improved, and difficult to update requirements automatically

3. Scenarios in Scenario Presenter
   - Web-based scenario presentations
   - Centrally-held scenario repository available to all
   - Improved functionality and usability
   - Able to add new requirements automatically
### Scenarios: ExcelPresenter Prototype

**Walkthrough scenario normal course, event order is machine-generated**

**CREWS-SAVRE machine-generates “what-if” alternative courses to discover missing requirements**

**Scenarios generated automatically as interactive Excel spreadsheets**

<table>
<thead>
<tr>
<th>Normal Course Event</th>
<th>Alternative Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event 1</td>
<td>Event 2</td>
</tr>
<tr>
<td>Event 2</td>
<td>Event 3</td>
</tr>
<tr>
<td>Event 3</td>
<td>Event 4</td>
</tr>
<tr>
<td>Event 4</td>
<td>Event 5</td>
</tr>
</tbody>
</table>

**Microsoft Excel - BAE SYSTEMS HARDILL LONG**

**CREWS-SAVRE SCENARIO PRESENTER**

- Add New Event
- Change Status
- Edit Event
- Add Event
- Delete Event
- Add for All Courses
- Add to File Table

**Characteristics for event of alternative course**

- Event 1: Action type
- Event 2: Action type
- Event 3: Action type
- Event 4: Action type
- Event 5: Action type

**CREWS-SAVRE scenario normal course, event order is machine-generated**

**CREWS-SAVRE machine-generates “what-if” alternative courses to discover missing requirements**

**Scenarios generated automatically as interactive Excel spreadsheets**

- Event 1: Action type
- Event 2: Action type
- Event 3: Action type
- Event 4: Action type
- Event 5: Action type
### Scenarios: Scenario Presenter
Available at www.soi.city.ac.uk/artscene

---

**ART-SCENE Scenario Presenter**

**Normal Course**

<table>
<thead>
<tr>
<th>Event</th>
<th>Action Type</th>
<th>Link</th>
<th>Description (Natural Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Communicative</td>
<td>End</td>
<td>Air platform emits surveillance radar</td>
</tr>
<tr>
<td>2</td>
<td>Communicative</td>
<td>End</td>
<td>Air platform emits surveillance radar</td>
</tr>
<tr>
<td>3</td>
<td>Communicative</td>
<td>Start</td>
<td>Air platform listens for surveillance radar</td>
</tr>
<tr>
<td>4</td>
<td>Communicative</td>
<td>Start</td>
<td>Air platform listens for surveillance radar</td>
</tr>
<tr>
<td>5</td>
<td>Communicative</td>
<td>End</td>
<td>Air platform detects false target jamming</td>
</tr>
<tr>
<td>6</td>
<td>Communicative</td>
<td>End</td>
<td>Air platform detects false target jamming</td>
</tr>
<tr>
<td>7</td>
<td>Communicative</td>
<td>Start</td>
<td>Air platform detects false target jamming</td>
</tr>
<tr>
<td>8</td>
<td>Communicative</td>
<td>Start</td>
<td>Air platform detects false target jamming</td>
</tr>
<tr>
<td>9</td>
<td>Communicative</td>
<td>Start</td>
<td>Air platform detects false target jamming</td>
</tr>
</tbody>
</table>

**Alternative Course for Event 1**

<table>
<thead>
<tr>
<th>ID</th>
<th>Description (Natural Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1g4</td>
<td>What if agent is unavailable at this time</td>
</tr>
<tr>
<td>G1g2</td>
<td>What if agent is not functioning</td>
</tr>
<tr>
<td>G1c13</td>
<td>What if information is a lie</td>
</tr>
<tr>
<td>G1c12</td>
<td>What if information is detailed</td>
</tr>
<tr>
<td>G1c11</td>
<td>What if information is insufficiently detailed</td>
</tr>
<tr>
<td>G1c6</td>
<td>What if insufficient information available</td>
</tr>
<tr>
<td>G1c3</td>
<td>What if information is incorrect</td>
</tr>
<tr>
<td>G1b1</td>
<td>What if agent is functioning incorrectly</td>
</tr>
<tr>
<td>G1c7</td>
<td>What if insufficient information available</td>
</tr>
<tr>
<td>G1c1</td>
<td>What if event does not occur</td>
</tr>
<tr>
<td>G1c1</td>
<td>What if event does not occur</td>
</tr>
<tr>
<td>G1c1</td>
<td>What if information is out of date</td>
</tr>
</tbody>
</table>

---

**Main Menu**
- View Norm
- View Art
- View Use Cases
- View Requirements
- Change Style
- Logout
Acquiring Requirements From Scenarios

1. Discover requirements from scenarios
   - Primary task is acquiring and documenting new stakeholder requirements
   - Document requirement basics (description, source, type, outline fit criterion) and scenario trace (scenario, event, alternative)

2. Manage requirements from scenarios
   - Primary task is to elaborate and refine the new stakeholder requirements
   - Link each requirement to scenarios event(s)
   - Use source scenario(s) to establish effective fit criterion for each requirement
   - Pass requirements through VOLERE Quality Gateway
Fourth Synchronisation Stage

1. **First synchronisation stage**
   - Gather data on human processes
   - Determine system boundaries
   - Determine system dependencies, goals and rationale
   - Model human activity

2. **Second synchronisation stage**
   - Descriptions of human activities
   - i* context model
   - i* context model
   - Human activity model
   - SD and SR models

3. **Third synchronisation stage**
   - Refine system dependencies, goals and rationale
   - Refine system dependencies, goals and rationale
   - Refined i* SD and SR models
   - Use case descriptions
   - Use case specifications

4. **Fourth synchronisation stage**
   - Use case model and summaries
   - Use case model and summaries
   - Use case model and summaries
   - System-level requirements and associated use cases
   - Use case specifications and associated requirements
   - Walkthrough scenarios
   - Define and document requirements
   - Define and document requirements
   - Scenarios associated with requirements
   - Refine and change requirements
   - Impact analysis
   - Define and document requirements
   - Define and document requirements
   - Define and document requirements
   - Refine system dependencies, goals and rationale
   - Refined i* SD and SR models
   - Use case model and summaries
   - System-level requirements and associated use cases
   - Use case specifications and associated requirements
   - Scenarios associated with requirements

**Creative design workshops**

**System-level requirements**

**Use cases**

**Use case model and summaries**

**Extended use cases**

**Use case descriptions**

**Use case specifications**
**Fourth Synchronisation Stage**

Formally examine outcomes from fourth stage

- Cross-check the use case descriptions, generated scenarios and stakeholder requirements
- Ensure that each requirement is linked to the system, use case, use case action or alternative course through the walkthrough scenarios
- Check whether each use case action is linked with requirements of the right types
- Check that the requirement fit criterion is grounded in the linked use cases

Establish an integrated requirements document

- All requirements linked through use cases that provide structure for the requirements
Requirement Inspections… Again

Ensure quality of requirements

– Impose VOLERE **Quality Gateway** on all requirements entering the requirements data base
– Formal inspection meetings (Sommerville & Sawyer 1997) on the resulting requirements documents and other models

Inspection meetings

– Cost-effective approach that can avoid stakeholder confrontations
– Make decisions to resolve incomplete requirements in the form of published actions
– Use **Quality Gateway** checklist (Robertson & Robertson 1999) to drive inspections
RESCUE: The Fifth Stage
Consequences
RESCUE’s Fifth Stage
Impact Scenarios

Use impact scenarios to explore consequences
– Projections of future system use in the environment
– Builds on existing use cases specification and scenarios
– Explore positive/negative impacts on the environment

Initial validation of the requirements in context
– Based on human analyses based on guided approach
RESCUE: Lessons Learned from the CORA-2 Project
CORA-2 General Lessons Learned

Overall lessons learned from CORA-2

– Start the process at the beginning of the project
  • CORA-2 work had already began before process was applied, so a certain degree of reverse-fitting had to take place
– Ensure that the project team understand the process
  • Due to late start, CORA-2 team members were learning segments of the process while undertaking other processes
  • Resulted in lack of commitment to various processes
– Do widespread staff training at the beginning
  • Have thorough training in the process and techniques before the requirements process starts - avoids re-working!
– Ensure sufficient stakeholder commitment and access
  • Requirements come from stakeholders - you need them!
– Implement process/technique-specific improvements
CORA-2 Successes: * System Models

CORA-2 team

- Delivered a complete SD model for CORA-2
  - Useful in determining system boundaries, one of the problems throughout the CORA-2 requirements process
- Developed some SR models for CORA-2 actors
  - Combines cognitive task models of desirable controller behaviour - interesting experiment that remains to be validated
- Applied REDEPEND version-1
  - Reported bugs, identified requirements implemented in v2

Lessons learned

- Use * modelling approach at beginning of process
- Ensure effective method and tool training at start
CORA-2 Successes: Creative Workshops

CORA-2 workshops

- **Surfaced** and **discovered** numerous requirements and ideas (>150) not known to the process beforehand
- Offered opportunities to “**clear the air**” about CORA-2
- Some techniques were more effective than others
  - Extreme scenarios, random scenario generation, dynamic working in small groups, exploring information visualisations, creating a risk-free space to explore, storyboards of new ideas
  - Analogical reasoning with non-ATM domains and fusing ideas together in novel ways were less successful

Lessons learned

- A space for creative design is beneficial
- Fewer but longer workshops with clearer system scope
- Reuse ideas generated in the CORA-2 workshops
CORA-2 Successes: Use Cases

Use case specifications

– Specified over 20 key CORA-2 use cases
– Writing the use cases requires other knowledge
  • How other dependent systems behave - numerous variations
  • Well-defined CORA-2 boundaries - what is the start event?
  • The essential design givens and the decisions made during CORA-2 design workshops
– Complete set of use case specifications were developed
– But writing the use cases led to numerous issues to fix

Lessons learned

– Interleave use case specification with the creative design workshops
– Use case specification needs more time and domain expertise
CORA-2 Successes: Scenario Walkthroughs

Extensive use of scenario walkthroughs
- Backbone of CORA-2 scenario-driven requirements discovery and specification process
- Discovered large number (150+) of new CORA-2 functional and non-functional requirements
- Structured process for effective communication
- Scenarios are quick to generate from use case specifications

Lessons learned
- Improve scenario walkthrough tool
  - Better scenario presentation, more useful functions
- Scenario short-cuts are needed to quicken walkthroughs
- Cost-benefits: a small number of generic alternative courses generate most new CORA-2 requirements
CORA-2 Successes: Activity Modelling

Little to report

- CORA-2 team did not follow the delivered work activity modelling process
- Unplanned cognitive task modelling took place using * semantics

This part of process remains untried

- Process, method, tutorial and training are all available
CORA-2 Successes: Requirements

Requirements management was successful

- Requirements were derived from $i^*$ models
- VOLERE shell was an effective and usable structure for describing system requirements, as implemented in Requisite Pro
- Fit criteria for most requirements have been derived using process type-measure rules

Lessons learned

- Ensure more stable use cases before acquisition
- Innovative meta-model pushes the limits of current requirements management tools
- Stakeholders must learn to see requirements in the context of use cases, not in isolation
- Writing fit criteria needs domain expert involvement
CORA-2 Successes: Impact Analyses

Little to report
  – CORA-2 team unable to use impact analysis as insufficient time prior to ORD delivery

This part of process remains untried
  – Again the process, method, tutorial and training are all available
RESCUE Process Resources

Team has access to following resources

- 200-page process document providing process, technique and tool overview
- Set of Microsoft Powerpoint documents describing process in more detail
- Set of interactive tutorials and training exercises
- REDEPEND, ExcelPresenter and ScenarioPresenter software modules
- Academic papers describing research and experience
- RequisitePro project structure and requirements types
- Use case authoring language and ATM specialisation of exception classes
- Reusable non-functional requirements and design ideas from the CORA-2 specification
Resources Available to Support RESCUE

Centre can make following resources available

– Centre academic staff
  • Dr Neil Maiden and Steph Wilson for process design, training, process and specification health checks

– Centre research staff
  • Dr Sara Jones (expertise in RE and HCI), Dr Cornelius Ncube (expertise in RE), Ms Sharon Manning

– Possible project team members
  • Honour degree City graduates in Software engineering, business computing and computer science, educated in systems and requirements engineering with techniques that form sections of the process