System Goal Modelling using the $i^*$ Approach in RESCUE

Centre HCI Design
27th February 2003
Learning Objectives

Three key objectives

– Introduce system goal modelling
– Provide Eurocontrol staff with \(i^*\) skills
– To offer advice to Eurocontrol staff on how to avoid past pitfalls

Learning objectives for Eurocontrol team

– To understand this stream in the RESCUE process
– To understand and appreciate the need to model stakeholder goals/requirements
– To understand the basic \(i^*\) framework
– To be able to apply \(i^*\) key techniques
– To be able to develop simple \(i^*\) models using the REDEPEND software tool
Tutorial Timetable

A simple timetable

– Thursday 27th February 2003
  Am: Background and overview
      Determine system boundaries
      Basic i* syntax and semantics
  Pm: Develop Strategic Dependency (SD) models
      Use the REDEPEND tool

– Tuesday 25th March 2003
  Am: Develop Strategic Rationale Models
  Pm: Continue Strategic Rationale Models
      Use the REDEPEND tool again
Running Examples

A range of examples in the tutorial

- Simple *internet airline ticketing system* example to demonstrate main concepts
- Simple *Motor insurance claim processing* example
- More complex *automated bus indicators* example to undertake and experiment with
- Real life examples included from CORA-2 project

Numerous short examples available to practice with

- *Automated rail ticketing system* example
- *Airport security system* example
- *Train signal* example
- *Automatic airline check-in* example
Part 1: Background and Overview
Requirements Modelling Vs Description

Requirements description
– Informal requirements attributes and structures
– Document-based, not amenable to automated analysis
  • For example, analysis checklists and interaction matrices
    (Sommerville & Kotonya 1998)

Requirements modelling
– Future system model, amenable to automated analysis
– Models used to infer properties
  • Incomplete and inconsistent requirements, potential sources of
    problem in the new system, consequences of making decisions
– Costs associated with requirements modelling
Modelling Requirements Dependencies

Stakeholder requirements are often conflicting
  – No single socio-technical solution can satisfy them
  – Requirements engineers often have to make complex trade-offs between requirements based on their priority, importance, risk, cost and time-to-deliver

Requirements dependencies are critical
  – To understand the important trade-offs to make
  – Modelling requirements dependencies is the most important role of requirements modelling

Several available modelling approaches
  – $i^*$ goal modelling approach is one of the most established and effective
The $i^*$ (Eye-Star) Goal Modelling Approach

Requirements modelling and analysis
- From research at the University of Toronto
  - PhD Thesis of Eric Yu (http://www.cs.toronto.edu/~eric/)
  - Syntax and semantics for modelling complex types of associations between requirements and other important concepts

Extended to integrate with RESCUE
- Process guidance for $i^*$ system modelling
- Cross-referencing $i^*$ system models with other requirement models and descriptions
- Software tool support for $i^*$ modelling
RESCUE Synchronisation Points

First synchronisation stage

Second synchronisation stage

Third synchronisation stage

Fourth synchronisation stage

Make decisions

Make decisions

Make decisions

Make decisions
System Goal Modelling

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

**Second synchronisation stage**
- Descriptions of human activities
- Human activity model
- Determine system dependencies, goals and rationale
- Creative design workshops
- Extended use cases
- Requirements

**Third synchronisation stage**
- Refined human activity model
- Refined i*SD and SR models
- Refine system dependencies, goals and rationale
- Specify use cases

**Fourth synchronisation stage**
- Use case specifications
- Walkthrough scenarios
- Define and document requirements
- Scenarios associated with requirements
- Impact analysis
- Refine and change requirements
- Define system-level requirements
- System-level requirements and associated use cases
- Use case specifications and associated requirements
- Use case model and summaries
- Use case model and summaries
- Use case descriptions
- Define and document requirements
- System-level requirements
System Modelling Stream Basics

Develop a **Context Model**
- Establish the strategic actors (Context Diagram)

Develop a **Strategic Dependency Model**
- Model requirements-related dependencies between strategic actors

Develop single **Strategic Rationale Models**
- Model what each actor can accomplish itself
- Model what each actor depends on other actors for

Develop **integrated I* SD and SR models**
- Integrate the single-actor SR models
- Model negative and positive links between requirements to explore requirements trade-offs
Part 2: Determining System Boundaries
Setting the Context

Determine system boundary
- From experience - an agreed system boundary will improve the requirements process
- Different boundaries arise in most complex socio-technical systems

Determine strategic actors of system
- Actors can be human and/or adjacent systems
- People/systems who have an interest in the product - they will build it, manage it, use it, or in some way be affected by its use
Setting the Context

Simple system scoping
- Use extended 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
Context Diagrams

Simple representation

- Useful to develop a first-cut context diagram
- Separates what the project team will design or redesign, and what is beyond its scope (and helps to obtain stakeholder agreement!)
- Provides the baseline for more complicated *SD model and use case models

Notation to use

- Established data flow diagram (DFD) notation
- Circles define the future system to design/redesign
- Use arrows to indicate flow of data to and from the system from external actors
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
Indicate Different System Boundaries

Computer-based systems to design or redesign
  – Main software development focus
  – These systems are often seen as the target systems

Users whose work is being designed or redesigned
  – Primary users - their work is changed by the computerised system
  – Redesign their work as part of socio-technical system

Existing systems or people influenced by system
  – Systems that will change to accommodate the new system and its users, but not dependent on it

External systems that do not change
  – No consequences due to introduction of the new system
Example: Context Model for CORA-2

- Flight data processor
- Environment data base
- Trajectory predictor
- Conflict detector
- Systems coordinator

- CORA-1, PAC, TED
- Tactical
- Planner
- CORA-2

- Departure manager
- En-route manager
- Arrival manager
How Boundaries Relate to Other Models

- Use case model boundary
- I* SR model boundaries
- Limits of requirements
- Dependencies beyond our interest

- CORA-2
- Tactical
- CORA-1, PAC, TED
- Planner
- Trajectory predictor
- Environment data base
- Flight data processor
- Department manager
- En-route manager
- Arrival manager
- Systems co-ordinator
- CORA-1, PAC, TED
- Dependencies beyond our interest
- I* SR model boundaries
- Limits of requirements
Modelling Actors

External systems, people and organisations
- Easier to recognise - adjacent systems tend to have well-defined boundaries
- Model roles of external people rather than people themselves - some people have different requirements depending on their role
- For example: radar, airlines, trajectory editor

Internal system roles
- Roles to be fulfilled by future system
- System modelling without design decisions
- More thorough actor model leads to more accurate requirements expression
- For example: planning and tactical controllers, conflict display device, controller communication mode
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

Adjacent system characteristics give us 3 roles

- **Active**: Dynamic systems that initiate events to achieve some goal or purpose
- **Autonomous**: Independent systems that act independently
- **Co-operative**: Predictable systems that are used to bring about some desired outcome
Active Adjacent Systems

Dynamic systems
- Initiate events to achieve some purpose or goal

Common characteristics
- Their behaviour is dynamic
- Able to interact with or participate in the work
- Are often human beings
- Initiate events to achieve purpose or goal
- Can predict this system’s behaviour (within reason)

Example

Customer
Example of an Active Adjacent System
From CORA-2

Question
– Why is the controller an active adjacent system?
Autonomous Adjacent Systems

Independent systems
  - Act independently

Common characteristics
  - Behave independently of other systems
  - Communicate through one-way data flows
  - Are often external bodies such as outside department, customers who do not direct interact with your system

Examples

- Airline
- NATS
- Govt
Example of an **Autonomous Adjacent System**
From CORA-2

Question
- Why is the conflict detector an autonomous adjacent system?
Co-operative Adjacent Systems

Common characteristics
  – Behaviour is predictable
  – Communication achieved through simple request-response dialogues
  – Co-operative systems often store data or provide predictable services - can be looked at as ‘black boxes’
  – Typical examples other systems that contains a used database, an operating system, or a system that provides a documented and immediate services

Examples

Airline timetables
Credit checking system
Example of a Co-operative Adjacent System
From CORA-2

Question
– Why is the trajectory editor a co-operative adjacent system?
Cross Model Checks in RESCUE

Compare context model and use case model
- Each actor in the use case model corresponds to one or more actors in the context model
- The boundary in the use case model corresponds to the level-1 boundary in the context model

Compare context model and human activity model
- Each actor in the human activity model corresponds to one or more actors in the context model
- Each data flow in the human activity model corresponds to one or more data flows in the context model
Exercises:
Context Modelling
Motor Insurance Claim Processing

Learning objective
– To practice the development of context diagrams

Problem
– An insurance company wants to minimise payments to owners who claim. Specific repair body shops are hired to carry out repairs. The company also wants to keep car owners happy so that they renew their policies. One method used to keep customers happy is to offer courtesy cars to car owners who’s cars are being repaired. The company is responsible for hiring the courtesy cars from a hire firm. The company must adhere to the standards set by its governing body.

Task
– Develop a simple context diagram
Automated Bus Indicators

Learning objective
– To practice the development of context diagrams

Problem
– Countdown is the new scheme being implemented by London Buses across London. It is designed to remove one of the principal deterrents to bus travel – uncertain waiting times. You might have seen the digital displays at bus stops – Countdown carries a number of pieces of information to waiting passengers in a clear, easy-to-understand form.

Task
– Define the key strategic actors
– Sketch first-cut context diagram

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>ACTON MKT PL</th>
<th>1 min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>207</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>83</td>
<td>GOLDERS GREEN</td>
<td>3 mins</td>
</tr>
<tr>
<td>3</td>
<td>207</td>
<td>SHEPHERDS BUSH</td>
<td>4 mins</td>
</tr>
</tbody>
</table>

......Delays due to London Mayor’s Show
1 207 ACTON MKT PL  1 min
2 63 GOLDERS GREEN  3 mins
3 207 SHEPHERDS BUSH  4 mins

Delays due to London Mayor’s Show...

Countdown indicator at bus stops

Road-side beacon

AVL Unit  Modem  Ticket machine
Odometer  Mobile radio
Units on the bus

Voice radio

Communication link every 30 seconds

X25 Network

BT Line

Central system

Route controller  Garage system
Context Diagram for Motor Insurance

- Hire firm
- Association of British Insurance
- Customer
- Body shop

Insurance Company
Part 3: Developing a Strategic Dependency Model
**i* Modelling Basics**

Key modelling semantics

- **Intentional strategic actor**
  - Intentional aspects such as objectives, rationale & commitments

- **Goal (functional requirement)**
  - Condition or state of the world that can be achieved or not

- **Task**
  - One particular way of attaining a goal - a detailed description of how to accomplish a goal

- **Resource**
  - Physical or informational objects in the world availability (e.g. the finished product of some action) available for use in a task

- **Softgoals (non-functional requirements)**
  - Goals that cannot be so sharply defined, such as goals that describe properties or constraints of the system being modelled
Some Words About Actors

Actors include the new system to introduce
Actors include actor roles

– A single user/adjacent system can instantiate several different actors
– Same actors can have different goals or requirements depending on their role
– Understanding different roles provides a deeper understanding of the context
– Important to make distinction between the roles of actors in the $i^*$ SD model

Returning to our airline ticketing example
– A passenger can fulfil several roles

- Purchaser
- Complainant
- Traveler
Strategic Dependencies

Developed from the context model

- Describes the network of relationships and dependencies among strategic actors
- Opportunities available can be explored
- Matching the depender who is the actor who “wants” and the dependee who has the “ability”

I want...

I can provide...

- Passenger
- Ticket reference
- Purchase quickly
- Website
- Ticket reference
- Purchase quickly
Strategic Dependency Modelling Cont.

Network of dependency relationships among actors
- Depender who is the actor who “wants” something
- Dependee who has the “ability” to do that something

Dependency relationships
- Actor X is dependent on actor Y for obtaining a resource
  - The letter 'D' on the dependency link is oriented from X to Y
- Actor Y is dependent on actor X for achieving a goal
  - The letter 'D' on the dependency link is oriented from Y to X
Four Dependency Relationship Types

Goal Dependency
- Depender depends upon the dependee to be able to bring about certain state in the world

Task Dependency
- Depender depends upon dependee to be able to carry out task

Resource Dependency
- Depender depends upon dependee for the availability of entity

Softgoal Dependency
- Depender depends upon dependee to perform some task that meets the softgoal or to perform the task in a particular way.
Strategic Dependency (SD) Modelling

Network of dependency relationships among actors
- Depender who is the actor who “wants” something
- Dependee who has the “ability” that something

Explore first of all using dependencies tables

<table>
<thead>
<tr>
<th>Subject</th>
<th>Noun</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Agent</td>
<td>for something</td>
</tr>
<tr>
<td>Pilot</td>
<td>Controller</td>
<td>to be safe (SG)</td>
</tr>
<tr>
<td>Pilot</td>
<td>Controller</td>
<td>for instructions (R)</td>
</tr>
<tr>
<td>Controller</td>
<td>Pilot</td>
<td>to redirect aircraft (T)</td>
</tr>
<tr>
<td>Student</td>
<td>Neil</td>
<td>to learn well (SG)</td>
</tr>
<tr>
<td>Neil</td>
<td>Student</td>
<td>to deliver lecture (T)</td>
</tr>
<tr>
<td>Customer</td>
<td>Airline</td>
<td>to have tickets bought (G)</td>
</tr>
</tbody>
</table>
Heuristics for Modelling Dependencies

*i*+ heuristics to guide dependency modelling

- Model dependencies between local actors - treat them as transitive, and avoid modelling duplicate dependencies

- Boundaries - if depender goals and soft goals to be tested for compliance, then actor is part of the socio-technical system

- Depender always initiates and owns the task

- Where possible, transform task- and resource-type dependencies into goal- and soft-goal-type dependencies by asking why does the depender need to undertake the task or have the resource?

- Model task-type dependencies if there are different ways of achieving a goal - otherwise model goal-type dependencies
Exercise:

$i^*$ Strategic Dependency Modelling
Motor Insurance Claim Processing

Learning objective
- To practice the identification of dependencies

Problem (as above)
- An insurance company wants to minimise payments to owners who claim. Specific repair body shops are hired to carry out repairs. The company also wants to keep car owners happy so that they renew their policies. One method used to keep customers happy is to offer courtesy cars to car owners who’s cars are being repaired. The company is responsible for hiring the courtesy cars from a hire firm. The company must adhere to the standards set by the governing body.

Task
- Write simple dependency sentences
Automated Bus Indicators

Learning objective
– To practice the identification of dependencies

Problem
– Countdown is the new scheme being implemented by London Buses across London. It is designed to remove one of the principal deterrents to bus travel – uncertain waiting times. You might have seen the digital displays at bus stops – Countdown carries a number of pieces of information to waiting passengers in a clear, easy-to-understand form.

Task
– Write some simple sentences about the dependencies between different actors in the domain
## Insurance Claim Dependencies

<table>
<thead>
<tr>
<th>Subject</th>
<th>Noun</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car owner</td>
<td>Insurance co</td>
<td>to be covered ((G))</td>
</tr>
<tr>
<td>Car owner</td>
<td>Insurance co</td>
<td>to be processed quickly ((S))</td>
</tr>
<tr>
<td>Car owner</td>
<td>Insurance co</td>
<td>to claim payment ((T))</td>
</tr>
<tr>
<td>Car owner</td>
<td>Car owner</td>
<td>to receive honest claims ((S))</td>
</tr>
<tr>
<td>Car owner</td>
<td>Garage</td>
<td>to have happy customers ((S))</td>
</tr>
<tr>
<td>Car owner</td>
<td>Garage</td>
<td>to have car repaired ((G))</td>
</tr>
<tr>
<td>Car owner</td>
<td>Insurance co</td>
<td>to have repair quickly ((SG))</td>
</tr>
<tr>
<td>Garage</td>
<td>Garage</td>
<td>to be listed garage ((G))</td>
</tr>
<tr>
<td>Garage</td>
<td>Insurance co</td>
<td>for repair costs ((R))</td>
</tr>
<tr>
<td>Garage</td>
<td>Garage</td>
<td>to keeps costs low ((S))</td>
</tr>
<tr>
<td>Insurance co</td>
<td>Hire firm</td>
<td>to have regular vehicles ((G))</td>
</tr>
<tr>
<td>Insurance co</td>
<td>Hire firm</td>
<td>to have happy customers ((S))</td>
</tr>
<tr>
<td>Insurance co</td>
<td>Insurance co</td>
<td>to have repeat business ((S))</td>
</tr>
<tr>
<td>Insurance co</td>
<td>Insurance co</td>
<td>to maximise income ((S))</td>
</tr>
<tr>
<td>Insurance co</td>
<td>Insurance co</td>
<td>to maintain standards ((S))</td>
</tr>
<tr>
<td>Association</td>
<td>Insurance co</td>
<td>to be member of Assoc. ((G))</td>
</tr>
</tbody>
</table>
Guidelines for Wording \textit{i*} Dependencies

Goals
- Wording of goals should describe a desirable state
  \begin{itemize}
  \item <desirable state>: Ticket purchased, car repaired
  \end{itemize}

Soft goals
- Describe some properties or constraints on that state
  \begin{itemize}
  \item <desirable state> <adjective | adverb>: Ticket purchased quickly, car repaired cheaply
  \end{itemize}

Tasks
- Active verbs describing how something is done
  \begin{itemize}
  \item <do task>: Purchase tickets online
  \end{itemize}

Resource
- Noun describing resource
  \begin{itemize}
  \item <resource>: Conflict information, 5 seconds, ticket
  \end{itemize}
Put It All Together in a Model
For Internet Airline Ticketing System

Think about clusters of dependencies
CORA-2: Strategic Dependency Model

Flight Data Processing System
Conflict Detector
Trajectory Editor
Clearance Assistant
Sequencing tools
Environment database
System Coordinator
Controller using cora 2

CORA 2: Strategic Dependency Model
Cross Model Checks in RESCUE

Cross-model checks at this stage

- Compare \( i^* \) SD model and activity models to check that goals, resources, constraints and context in activity modelling appear, where relevant, in \( i^* \) SD model.

- Compare \( i^* \) SD model and use case model to check that the external actors in \( i^* \) SD model are equivalent to the external actors in use case model.

- Also check that each task dependency in the \( i^* \) SD model has a corresponding use case in use case model.

- Compare \( i^* \) SD model and system-level and use case requirements to check that each goal and soft-goal that the future system achieves (according to the \( i^* \) SD model) is described in the system requirements specification and stored in the requirements database.
Exercise:

$i^*$ Strategic Dependency Modelling
Motor Insurance Claim Processing

Learning objective
– To practice producing the Strategic Dependency model

Problem (as above)
– An insurance company wants to minimise payments to owners who claim. Specific repair body shops are hired to carry out repairs. The company also wants to keep car owners happy so that they renew their policies. One method used to keep customers happy is to offer courtesy cars to car owners who’s cars are being repaired. The company is responsible for hiring the courtesy cars from a hire firm. The company must adhere to the standards set by the governing body.

Task
– Produce a Strategic Dependency model
Automated Bus Indicators

Learning objective
- To practice producing the Strategic Dependency model

Problem
- Countdown is the new scheme being implemented by London Buses across London. It is designed to remove one of the principal deterrents to bus travel – uncertain waiting times. You might have seen the digital displays at bus stops – Countdown carries a number of pieces of information to waiting passengers in a clear, easy-to-understand form.

Task
- Produce a Strategic Dependency model
SD Model (1) for Motor Insurance

- Car owner
  - Repair quickly
  - Car be repaired
  - Be listed garage
  - Repair costs
  - Garage

- Insurance company
  - Keep costs low
  - Repair costs be covered
  - Process quickly
  - Claim not fraudulent
  - Customer be happy
  - Association of British Insurance
    - Be member
    - Regular vehicles received
    - Customer be happy
    - Repeat business
    - Maximise income
    - Claim excess payment
    - Claim not fraudulent
    - Customer be happy
    - Repeat business
    - Maximise income

- Hire firm

- Customer be happy

- Be member

- Repair quickly

- Be listed garage

- Keep costs low
SD Model (2) for Motor Insurance

- Car owner
  - Car be prepared
  - Repair quickly
- Repair costs
  - Repair quickly
  - Keep costs low
- Fair appraisal
- Appraiser
  - Keep costs low
  - Minimal repairs
  - Secure employment
- Association of British Insurance
  - Be member
  - Adhere to standards
- Insurance company
  - Be listed body shop
  - Repair costs
  - Appraise damages
- Customer be happy
  - Claim not fraudulent
  - Process quickly
- Claim excess payment
  - Repairs be covered
  - Repairs be covered
- Car be prepared
  - Repair quickly
Summarising Context and SD Modelling

Part of the RESCUE Boundaries stage
- Model system boundaries in terms of actors, data flows between, and dependencies between actors
- * modelling is not an end, but a means to explore, analyse and negotiate system boundaries
- Spend time exploring boundaries - you will need this platform to specify requirements effectively

How to proceed
- Practice, practice, practice
- * modelling takes some practice, but applied proficiently, it is a very useful technique