Requirements Modelling for Socio-Technical Systems

Professor Neil Maiden
2010

Module Administration

Contact details
– 0044-20-7040-8412, N.A.M.Maiden@city.ac.uk

Web Sites
All material is available on City Space

Coursework dates
– Submission date: 12th March 2010
Module Objectives

To make you aware of
– Importance of effective requirements engineering
– Requirements for socio-technical systems
– The i* modelling approach

To give you
– Practical skills in one leading-edge method and technique for socio-technical system modelling

To make you think about
– Requirements engineering is part of a wider design process

Timetable

Wednesday 24th
13.00 Lecture: Socio-technical systems
13.15 Lecture: Requirements for socio-technical systems
13.45 Exercise: Writing requirements on socio-technical system
14.15 Lecture: Satisfaction arguments
14.35 Exercise: Writing satisfaction arguments
15.00 Lecture: Context modeling
15.20 Exercise: Context modeling
15.35 Lecture: Strategic dependency modeling
16.20 Exercise: Strategic dependency modeling
17.00 Close
**Timetable**

**Thursday 25th**

- **10.15** Lecture: Strategic relationship modeling
- **11.15** Exercises: Strategic relationship modeling
- **12.00** Lunch
- **13.15** Lecture: Examples of $i^*$ models
- **13.30** Lecture: Discovering requirements from $i^*$ models
- **14.20** Class discussion
- **14.40** Set coursework exercise
- **15.00** Close

---

**Module Reading Material**

Three good overall text books to use are:


Other useful textbooks are:


A more complete list in the module notes
Socio-Technical Systems

Socio-technical systems
- Systems in which the reciprocal interrelationship between humans and machines shapes both the technical and the social conditions of work
- Software actors, human actors, machine actors, organizational requirements

Requirements on socio-technical systems
- On whole socio-technical system
- On actors in socio-technical system
Example Socio-Technical System

Requirements for Socio-Technical Systems
What is a Requirement?
Definitions of a requirement
– Something that a product must do or a quality that the product must have (Robertson & Robertson 1999)
  • Interesting focus on product rather than software system
– Expression of the required phenomena that are shared between a machine (product) and the domain or environment (Jackson 1995)
  • Requirement refers to both the domain and the machine
– Requirements invariably contain a mixture of problem information, statements of system behaviour and properties, and design and implementation constraints (Sommerville and Sawyer 1997)
  • Requirements can be nasty little things to express

User and System Requirements
User requirements
– Expressed by users and other types of stakeholders to define desirable properties of the domain, independent of the system
System requirements
– Desirable properties of system introduced into domain
In the requirements process
– User requirements emerge before system requirements
– System requirements are specified so that the user requirements are met
But how does this happen....
The Structure of a Requirement

Use the VOLERE template
- Available from http://www.atlsysguild.com
- See also standards such as PS-005 (Mazza et al. 1994)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Requirement Type</th>
<th>Event/use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit criterion:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer satisfaction:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer dissatisfaction:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependencies:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conflicts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supporting materials:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>History:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supermarket Checkout System Example

Important attributes
- **Type:** Functional requirement
- **Description:** The customer shall receive a receipt from the checkout system
- **Rationale:** Customers need receipts as evidence...
- **Source:** Store management team
- **Fit Criterion**
  - For a representative sample of 100 customers and 1000 customer purchases with cash, debit cards, credit cards, cheques, club cards and special offer vouchers, the system prints the correct receipt with the correct information for 100% of all purchases. An independent human checker will test compliance by random sampling of 100 customers purchasing a representative range of items during one 8-hour test period, checking purchased items against items on the receipt.
- **Satisfaction:** 3  **Dissatisfaction:** 5
- **Dependencies:** Electronic cashier system
- **Supporting materials:** Supermarket document XYZ.
Another Supermarket Checkout Example

- **Type:** Performance requirement
- **Description:** The customer shall receive a receipt from the checkout system within a time acceptable to the customer
- **Rationale:** Customers get impatient if kept waiting
- **Source:** Customer representatives
- **Fit Criterion**
  - For a representative sample of 100 customers and 1000 customer purchases the system prints all receipts of less than 25 items in less than 3 seconds, receipts of 26-40 items in less than 5 seconds, and 99% of a random sample of all receipts in less than 10 seconds from pressing the 'transaction complete' button. An independent human checker will test compliance by random sampling of 100 customers purchasing a representative range of items during one 8-hour test period. Timings will be tested using an electronic stopwatch based on observation.
- **Satisfaction:** 4  
  **Dissatisfaction:** 4
- **Dependencies:** Previous Functional Requirement
- **Supporting materials:** Supermarket document XYZ.

Requirements Writing Guidelines

**Structuring the VOLERE requirements description**
- **Practical guidelines** (Alexander & Stevens 2002)
  - **Sketch** then evolve the requirement
    - Difficult to get the wording right first time

**Common anatomy** of a good requirement
- **Actor type:** actor benefiting from the requirement - often the actor who is tested for requirements compliance
- **Result type:** normally the main verb
- **Desirable state** for actor to reach - normally an object and a qualifier that hints at the fit criterion

Most requirements shall adhere to this structure
Good Requirements Examples

Consider simple examples
– Most requirements should adhere to structure

<table>
<thead>
<tr>
<th>Actor type</th>
<th>The call centre operator...</th>
<th>The pilot ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result type</td>
<td>... shall be able to view ...</td>
<td>... shall control ...</td>
</tr>
<tr>
<td>Object</td>
<td>... details of a protected household ...</td>
<td>... the aircraft's angle of flight ...</td>
</tr>
<tr>
<td>Qualifier</td>
<td>... within two seconds of issuing a query.</td>
<td>... with one hand.</td>
</tr>
</tbody>
</table>

To Avoid When Writing Requirements

Some important heuristics
1. Avoid ambiguities, such as the term or
2. Don’t make multiple requirements and avoid terms such as and, or, with, also
3. Don’t build in let-out clauses such as if, when, but, except, unless, although, always
4. Don’t ramble
5. Don’t design the system
6. Don’t mix requirements and design
7. Don’t mix requirements and plans
8. Don’t speculate and avoid terms like usually, generally, often, normally, typically
9. Don’t play on ambiguous requirements
10. Don’t use vague undefinable terms
11. Don’t express possibilities and avoid terms like may, might, should, ought, could, perhaps, probably
An Expert Opinion: Tim Lister

“Drill down into the minutiae, then come back up - it bounces”

Functional Requirements

The most common type of requirement
- Something (service, behaviour or function) that a product must do (Robertson & Robertson 1999)

Examples are simple to find
- Supermarket: The customer shall receive a receipt from the check-out system
- LAS-CAD: The ambulance control system shall detect the location of all ambulances
- CORA-2: The air traffic controller shall receive all candidate resolutions from the CORA-2 system

How to test whether a functional requirement is met
- Measurable fit criterion in terms of predicted outcomes
- The receipt is printed, location of all ambulances are known, controller has access to all resolutions
Non-Functional Requirements

Also known as Quality requirements

- Express the desirable qualities of the product
- Predefined set of types: Performance, look-and-feel, device, usability, training, availability, maintainability, recoverability, portability, reliability, security, safety, contract and supplier-type requirements (based on Roman 1985, Robertson & Robertson 1999)
- Predefined types of units of measure and test, by type

Performance requirement

- Specifies time to do things, required throughput rates
- Measure using response times or times to undertake an action (speed), actions per time period (throughput), or numbers of units handled (load)
- Test using response timing, load tests, throughput tests

More Non-Functional Requirements

Look-and-feel requirement

- Specifies how end-users will perceive the product
- Measure using adherence to specified standards, use of colours, adoption of designs
- Test using observations of the product by independent assessors, standards compliance rules

Device requirement

- Specifies features, perhaps interactive, of the product
- Measure using adherence to specified standards, device features and attributes
- Test using observations of the product by independent assessors, standards compliance rules
More Non-Functional Requirements

Usability requirement
- Specifies how people will interact with the product
- Measure using task completion times, usage error-rates and usage rates and frequencies
- Test (with HCI techniques) with usability evaluation, protocol analysis and cognitive walkthrough techniques

Training requirement
- Specifies levels and nature of training to use the product
- Measure using training duration and outcomes
- Test using training course outcomes

Availability requirement
- Specifies nature and levels of access to the product
- Measure using times when available, %age downtimes
- Test using system-level trials and user experiences

More Non-Functional Requirements

Maintainability requirement
- Specifies the acceptable levels of upgrade of product
- Measure using time and resources to maintain
- Test using simple maintenance tasks

Recoverability requirement
- Specifies the repair of the product in case of failure
- Measure using time and likelihood to recover
- Test using simple maintenance and recovery tasks

Portability requirement
- Specifies platforms that product needs to operate on
- Measure using the names and versions of products and operating systems
- Test using usage trials on range of products
More Non-Functional Requirements

Reliability requirement
– Specifies levels of failure supported in the product
– Measure using mean-time between (defined) failures
– Test using product reliability trials, customer evidence

Security requirement
– Specifies levels of illegal access to the product
– Measure using specified access functions, and mean-time between breaches
– Test using security experts

Safety requirement
– Specifies how safe is the product
– Measure using number/risk of injuries in total/over time
– Test using health and safety compliance techniques

Example Non-Functional Requirements

Example statements
– PR: the check-out system shall respond to an operator request quickly
– DR: the check-out system shall have a credit card reader device
– UR: the operator shall use the check-out system without making errors
– AR: the check-out system shall be available for use by the operator during shop-opening hours
– MR: the maintenance team shall maintain the check-out system with an agreed maintenance period
– RR: the check-out system shall operate with accepted levels of system error
Exercise
Writing Requirements on Socio-Technical Systems

Requirements for Baggage Carousel
Consider a typical airport baggage carousel
- Actors in socio-technical system include carousel, passenger, porter, baggage handler

The task
- Write different requirements statements on the different actors in the baggage carousel socio-technical system
Lecture
Satisfaction Arguments

Satiation Arguments

From REVEAL method (Hammond et al. 2001)
- Relates domain knowledge to introduction of a new system in an environment (Jackson 1995)

Satisfaction argument
- Given properties of domain D, combined with the specification of the behaviour of new system S, requirements R on new system will hold

Consider a pedestrian crossing
- R: People cross the road safely
- S: Specification of zebra crossing
- D: Vehicles will stop
Writing Satisfaction Arguments

Use simple tables to structure each argument

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
<th>Domain properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customers checkout</td>
<td>Checkout staff trained</td>
<td>Training reduces errors. Trained staff do not need to seek advice from colleagues.</td>
</tr>
<tr>
<td>Customers checkout</td>
<td>Scan item barcodes</td>
<td>Barcode scanning is quicker than typing barcodes.</td>
</tr>
<tr>
<td>Customers checkout</td>
<td>Laser print receipts</td>
<td>Laser printers are the fastest type of printer available.</td>
</tr>
</tbody>
</table>

Another Satisfaction Argument

Taken from Shinkansen passenger example

- Requirement is passengers can board quickly
- Specification of system includes platform layout markings, onboard train indicators, and trained guards on the platform
- Domain properties such as orderly queuing

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
<th>Domain properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passengers can board</td>
<td>Marked platform queues</td>
<td>Passengers will adhere to the queuing system and will not stand in other places.</td>
</tr>
<tr>
<td>Onboard train indicators</td>
<td></td>
<td>Passengers will prepare to disembark before arrival.</td>
</tr>
<tr>
<td>Guards on platform</td>
<td></td>
<td>Passengers will respond to advice from guards to aid quick boarding.</td>
</tr>
</tbody>
</table>
Exercise
Writing Satisfaction Arguments for a Baggage Carousel

Satisfaction Argument Exercise
Consider a typical airport baggage carousel
– One requirement R for the user already exists
  – The passenger shall reclaim baggage successfully

The task
– Use your knowledge to define specifications S that lead to satisfaction of the requirement R
– Produce a satisfaction argument showing that R can be satisfied through satisfaction of S
Lecture
Context Modelling

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 i* 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

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**

- Departure manager
- En-route manager
- Arrivial manager
- Flight data processor
- Environment data base
- Trajectory predictor
- Conflict detector
- Systems co-ordinator
- CORA-1, PAC, TED
- Planner
- Tactical

**Example: Bank ATM System**

- Customer
- Cashier
- ATM
- Maintenance staff
- Other bank systems
Context Diagram for Countdown

Exercise
Context Modelling
Context Modelling Exercise

Learning aim
– To develop $i^*$ modelling skills, to understand different types of agent dependencies in a socio-technical system

Car insurance application
– An insurance company introduces a new computer system to minimise payments to car owners who claim. Clerks in the company operate the system. Appraisers are hired by the company to keep claims to a minimum, but are dependent on the company for work. The company also wants to keep car owners happy so that they renew their policies. Car owners want fair assessments of car damage. They are likely to get garages to give repair estimates that maximise insurance payouts.

Task
– Using this and your general knowledge of claims, develop a context model identifying all actors for the socio-technical system

Lecture
Strategic Dependency 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 and trade-offs are the most important role of requirements modelling

Several available modelling approaches
- i* goal modelling approach is 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

Two model types to be familiar with
- Strategic Dependency model
  - Models requirements-related dependencies between strategic actors in the future system
- Strategic Rationale model
  - Models rationale, decompositions and contribution links between important goals of different system actors
  - Rationale overlaps with design rationale ideas from earlier
**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 be introduced

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
Strategic Dependency 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>Passenger</td>
<td>web-site</td>
<td>tickets purchased (G)</td>
</tr>
<tr>
<td>Passenger</td>
<td>web-site</td>
<td>to buy ticket (T)</td>
</tr>
<tr>
<td>Passenger</td>
<td>web-site</td>
<td>for ticket reference (R)</td>
</tr>
<tr>
<td>Passenger</td>
<td>web-site</td>
<td>purchased quickly (SG)</td>
</tr>
<tr>
<td>Airline</td>
<td>passenger</td>
<td>revenues maximized (SG)</td>
</tr>
<tr>
<td>Web-site</td>
<td>Airline</td>
<td>reliable information (SG)</td>
</tr>
</tbody>
</table>

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 Model
For Internet Airline Ticketing System

Think about clusters of dependencies

Dependency Types and Delegation
Choosing the right dependency type can be tough
  – It is all about delegation from depender to dependee
  – In simple terms, soft goal and goal dependencies indicate more delegation

Simple heuristics
  – Soft goal and goal dependencies - full delegation
  – Task dependencies - partial delegation - teamwork
  – Resource dependencies - minimum delegation
Strategic Dependency Model
Slightly revised model taking delegation into account
– Internet Airline Ticketing System

For banking with an ATM
Strategic Dependency Model

Dependencies between
- Customers and London Underground

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
Guidelines for Wording i* Dependencies

Goals
- Wording of goals should describe a desirable state
  - \(<\text{desirable state}>\): Ticket purchased, car repaired

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

Tasks
- Active verbs describing how something is done
  - \(<\text{do task}>\): Purchase tickets online

Resource
- Noun describing resource
  - \(<\text{resource}>\): Conflict information, 5 seconds, ticket

Model Adjacent Systems

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

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
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

CORA-2: Strategic Dependency Model
SD Model for Countdown

- Comms system
- Driver
- GPS
- Countdown Display
- Destination arrived at
- Make travel decision
- Inputted safely
- Defined destination
- New route received
- Communicate safely
- Buses status received
- Re-route effectively
- Traffic information updated
- AVLS system

- London Transport
- Make routes effectively
- Locate bus
- Display bus information
- Bus information up-to-date
- Route controller
- Onboard Bus System

REDEPEND Software Tool
REDEPEND (RE DEPENDencies)
- Centre prototype for developing i* SD and SR models
- MS-Visio plug-ins to draw and analyse models
Model Checking with REDEPEND

Model checking
- $i^*$ models are large, complex and necessitate computational model checking
- Check for unrecognised connections, invalid connections and model bottlenecks

Exercise

$i^*$ Strategic Dependency Modelling
**i* Strategic Dependency Modelling**

**Learning aim**
- To develop *i* modelling skills, to understand different types of actor dependencies in a socio-technical system

**City’s web-based fees payment system**
- E-Fees is intended to improve payment of student fees, managed by finance staff who also handle payment enquiries. Students want a quick and reliable online means to pay their fees, but can make queries to finance staff. Finance staff want to reduce time spent processing fees and ensure data with the student records system is consistent. The e-Fees system also needs to undertake credit checks on a credit check system
- Using this and your general knowledge of paying student fees, develop a context model (12 minutes) and a Strategic Dependency model (35 minutes)

---

**Lecture**

**Strategic Rationale Modelling**
Lecture Structure

Reminder
- Strategic Dependency models

Strategic Rationale models
- New semantics and syntax
- How to produce a model
  - Practice producing a SR model
- Linking the models together
  - CORA-2 Strategic Rationale model

REDEPEND software tool

Conclusions

Strategic Rationale (SR) Modelling

Describes actor goals, tasks, resources and soft goals
- SR model specifies what actors accomplish themselves
- Adding SD model specifies what need other actors to accomplish

Three additional semantics added to SD semantics
- Task decomposition links
  - Decompose task into sub-components of all types
- Means-end links
  - A relationship between an end and a means for attaining the end
- Contributes-to soft goal links
  - A means-end link with a softgoal as the end
Task Decomposition Links
Decompose task into sub-components of all types
– All sub-components need to be “completed” for a task to be performed, so logical “AND” between them

– Answers “what?” questions

Task Decomposition Links
Four types of task-decomposition link
– Task sub-goal decomposition
– Task sub-task decomposition
– Task resource decomposition
– Task sub-goal decomposition
Means-End Links
Relating requirements and solutions
– Relationship between an end (a goal or resource) and a means (how to do something often expressed as a task) for attaining the end
– Models alternative ways (tasks) for accomplishing a goal
– Provides a logical “OR” relationship between sub-components
– Provides answers to “HOW” and “WHY” questions

Four main types of Means-End Links
Goal-task link
– End is specified as goal and means is specified as task
– Goal might have different means ends links

Resource-task link
– Task indicates how resource can be obtained

Goal-goal link
– Permits the reduction of goals to sub-goals

Task-task link
– Represents one method for decomposing a task into possible alternative subtasks, related with a logical “OR” relationship
Contribute-To Soft Goal Links
A means-end link with a soft goal as the end
– Represents the positive or negative contribution of a process element towards the achievement of a soft goal
– Identifies requirements trade-offs by positive or negative contributions to satisfaction of the soft goal
– Provides an implied logical “OR” relationship between sub-components with +ve and -ve contributions

Contribute-To Soft Goal Links
A means-end link with a soft goal as the end
– Note that no means can be specified for the achievement of a goal that cannot be clearly defined
– The only restriction is that the end should always be a soft goal
– Different types of contribute-to soft goal links are permissible

– Purchase online and internet connections both contribute positively, while seeking to minimise own work does not
SR Models and Satisfaction Arguments

*i* means-ends links
- Define how concepts contribute to goal attainment and soft goal achievement - positively and negatively

REVEAL satisfaction arguments
- Given properties of domain D, combined with the specification of the behaviour of new system S, requirements R on new system will hold

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
<th>Domain properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passengers can board quickly</td>
<td>Marked platform queues</td>
<td>Passengers will adhere to the queuing system and will not stand in other places.</td>
</tr>
<tr>
<td>Onboard train indicators</td>
<td>Passengers will prepare to disembark before arrival.</td>
<td></td>
</tr>
<tr>
<td>Guards on platform</td>
<td>Passengers will respond to advice from guards to aid quick boarding.</td>
<td></td>
</tr>
</tbody>
</table>

Heuristics for Rationale Dependencies

*i** heuristics to guide rationale modelling
- Task decomposition rules imply no ordering
- Sub-goal of a task to be treated as a pre-condition on the task being completed
- Sub-soft goal of a task must be satisfied by the completion of task, that is post-condition
- When deciding whether to model a concept as a task or goal, always choose a goal unless there are 2 specific ways of doing something
- A task is often decomposed into at least one sub-task, one sub-goal and one sub-resource
- Model the right goals, soft goals, resources and tasks of each actor - mis-allocation is common
- Ask pre-defined questions of each model to check its completeness
How to Produce Strategic Rationale Model

Three key stages
1. Model what each actor can accomplish itself
   • If start with soft goals there are no mandated solutions then decompose soft goal with contribution hierarchies, and explore different means of achieving ends (design ideas)
   • If start with high-level tasks then a solution, such as online purchasing, has been mandated, then ask why questions to establish related goals and soft goals
   • Ensure SR model is complete with respect to SD model
2. Model what each actor depends on other actors for
   • Include dependencies from and to actors on SD model
3. Integrate the single-actor SR models together
   • Use the dependencies from the SD model

Cross-Checking SD and SR Models

The SD model supports the SR model
– Elements of SD model become elements of SR model

Learn the simple heuristic for Actor A
– IF Actor A is a depender is a dependency relationship in the SD model THEN the depended-upon element is modelled in Actor A’s SR model
– A depender wants something - that wanted goal, soft goal, task or resource is included in relevant SR model
– The wanted element is linked using the SD dependency links to other actors in the SD model
Modelling Dependencies on 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

Strategic Rationale Model
Bank customer and ATM
– Extends the SD model
Strategic Rationale Model
Customers and London Underground
– Extends the SD model

Strategic Rationale Model: CORA 2

**Passenger**
- Travel
- Read about travel news
- Pay
- Costs minimized
- Travelled reliably
- Destination achieved

**London Underground**
- Run trains
- Revenue
- Vandalism minimized
- Maintain trains
- Advertise travel news

**Resolution generation**
- Generate resolutions
- Provide resolutions
- Provision of resolutions for display
- Update resolutions
- Manage resolution change
- Co-ordination to be triggered
- Manage invalid resolution
- Manage updated resolution

**Resolution update**
- Generated resolutions
- Invalid resolution
- CORA 2
- Resolution revision
- Reserved resolution revision
- Evaluate one resolution
- Initiate resolutions calculation
- Identify resolution change
- Provide update for resolution availability
- Initiate update with new trajectory
- Handle resolution input
- Support co-ordination

**Implementation**
- Support implementation
- Manage proposing co-ordination
- Manage receiving co-ordination
- Manage proposing co-ordination discontinuation
- Manage receiving co-ordination discontinuation
- Manage co-ordination discontinuation
- Be supported for giving instructions
- Be aware of resolution availability
- Be reminded to implement an overall best-ranked resolution

**Co-ordination**
- Co-ordination details
- Co-ordination result

**Support to revise resolution**
- Display Cost value
- Communicate with TED
- Edited resolution
- Resolution to be modified
- Resolution to be modified

**Resolution to be modified**
- Edited resolution

**Co-ordination**
- Co-ordination to be triggered
- Co-ordination discontinuation trigger
Exercises
Strategic Rationale Modelling

Modelling a Fast Food Restaurant

Learning objective
– To practice modelling \( i^* \) strategic rationale models

Problem
– Think about a typical fast-food outlet. A customer depends on an employee for food. To do this the employee uses a new cash till - the new system - to support this work

Task
– Produce an \( i^* \) strategic rationale model for SD model shown right
Lecture
Examples of $i^*$ Models

SD Model for Infringement Detection
SR Model for Infringement Detection

SD Models for Tomato Traceability
SR Model for Tomato Traceability

SD Model for Dairy Traceability
**SR Model for Dairy Traceability**

---

**i* Strategic Rationale Modelling**

**Learning aims**
- To develop i* modelling skills, to understand the nature of actor dependencies in a multi-actor SR model

**An airport security system**
- Assume that there are 2 actors - airport security and passenger. Passengers want to be safe but not inconvenienced, and security want to check all passengers whilst maximising the throughput of people. Security check both the person and the hand baggage using different devices that we hope are reliable. Just think about going through security at an airport. Here is a simple SD model.

---

[Diagram of airport security system model]
Airport Security Gates

Lecture
Discovering Requirements from i* Models
RESCUE Requirements Process

Modeling precedes textual specification
- Produce and analyze system-wide models
Challenge
- Transform i* requirements models into textual requirement statements structured in VOLERE shells

i* System Models for DMAN
DMAN system
- Departure manager for major European airports

15 actors
46 dependencies
Generating Text Requirement Statements

Manual application of 19 simple patterns
- One or more requirements on the future system that must be satisfied for the i* model dependency to hold for the future system

Two-phase pattern development
1. Map i* & VOLERE concepts with RESCUE meta-model
2. Design 19 patterns using RESCUE requirements types

Patterns P3 and P10

FR (NSA shall attain G), FR (ASA shall provide NSA with R), FR (NSA shall receive R from ASA), NFR (NSA shall attain G accurately), NFR (NSA shall attain up-to-date G), NFR (ASA shall provide NSA with up-to-date R), NFR (ASA shall provide NSA with R accurately)

FR (STA shall do T), FR (NSA shall do subtask of T) * can be multiple tasks, FR (NSA shall provide R to STA), NFR (NSA shall provide R to STA on time)

Overall Results

Major contribution to DMAN specification
- 85 pattern application instance in SD model
- 214 new requirement statements, approximately 25% of total
- Manual application - human judgment that required 10 working days
- All statements expressed using VOLERE shell

<table>
<thead>
<tr>
<th>Requirement</th>
<th>FR226</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Functional</td>
</tr>
<tr>
<td>Description</td>
<td>DMAN shall calculate the optimized runway capacity</td>
</tr>
<tr>
<td>Rationale</td>
<td>From SD model</td>
</tr>
<tr>
<td>Source</td>
<td>Dependency [ATC tower supervisor depends on DMAN to attain goal runway capacity optimized]</td>
</tr>
<tr>
<td>Dependencies</td>
<td>FR225 (ATC tower supervisor shall optimize the runway capacity)</td>
</tr>
</tbody>
</table>
Detailed Results

Most applied patterns
- 8 instances of P3 led to 26 requirement statements
  - New system depends on adjacent system to achieve goal
- 8 instances of P6 led to 23 requirement statements
  - Adjacent system depends on new system to achieve goal
- 13 instances of P8 led to 25 requirement statements
  - Stakeholder depends on new system to achieve goal
- 11 instances of C1 led to 34 requirement statements
  - Goal dependency with composite goal

Consequences
- 207 of the 214 requirements included in final DMAN specification
- 157 of 187 system-wide requirements from SD model

Discussion and Future Work

Automatic requirement generation
- Implement in REDEPEND
- Generate candidate requirements that engineers can recognize as relevant
- Less dependent on human recall

Natural language generation
- Case frame grammars to generate textual statements

Satisfaction arguments (Rawlings et al. 2001)
- DMAN requirements lacked structure
- Abstract “why” arguments to instantiate
Lecture
Walking Through i* Models to Explore the Impact of Requirements

Previous Experiences

Weaknesses with i*
1. Inadequate semantics for means-end links
2. Poor integration with requirements processes

Two i* extensions
1. Integrate satisfaction arguments into i*
2. Link i* models to text requirements specifications

Extend REDEPEND
- Our i* environment
CAIT: Infringement Detection System
Reference model linking software requirements to safety-related soft goals

Extend $i^*$ Means-End Semantics
Satisfaction argument
– Given properties of domain $D$, combined with the specification of the behaviour of new system $S$, requirements $R$ on new system will hold
Integration with $i^*$ via conceptual model
**CAIT: Infringement Detection System**

- **R:** Be aware of alerts
- **S:** Display picture
- **D:** The ATCO is monitoring the air picture. Alerts are only communicated through the air picture.

**Argumentation:** The alert is reported with the air picture that is displayed with the CWP, to provide a single information source. It is the only source of alert information for the civil ATCO.

---

**Extending the Requirements Process**

**Determine goals impact**
- Start from non-compliant tasks
- Propagate non-compliant elements through model

**Use satisfaction argument**
- Invalidate domain properties
- Invalidate argumentation

**Requirement/i* model matrix**

<table>
<thead>
<tr>
<th>Functional requirement</th>
<th>CAIT: Record alerts [T]</th>
<th>Civil ATCO: L-T memory of infringements</th>
<th>Engineering: Set active regions for CAIT [T]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAIT alerts shall be logged automatically</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>It shall be possible to set active regions.</td>
<td></td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation
Applied to NATS CAIT system requirements
- Three analysts developed 10 satisfaction arguments
- Two analysts reviewed impact of 25 CAIT software requirements on safety-related goals
- More process guidance and explanation of satisfaction arguments was needed
- Writing complete and correct satisfaction arguments remains challenging
For next version of process and REDEPEND

Class Discussion
Coursework Exercise

Develop one i* SD and one SR model

- **What:** Group work, working in groups of 4 students
- **Deadline:** Friday 6\(^{th}\) March 2009, 17.00 hours
- **To:** N.A.M.Maiden@city.ac.uk
- **How:** Single PDF document, sent as attachment or link to download with one click

The domain

- London Heathrow T5 baggage handling system
- Coursework document available for download
- We will provide model answers after the deadline