# TDT4252 / DT8802 Enterprise Modelling and Enterprise Architecture

Sobah Abbas Petersen

Adjunct Associate Professor
sap@idi.ntnu.no

# Overview of lecture today

- Brief overview of previous lecture
- Continue with Actor-role oriented modeling
- **i*, Use Cases, combination of modelling languages and methods**

Based on the following articles:

– A03: Maiden et al. Model-driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study

# Overview from Previous Lecture

- We looked at i*, GRL and UCM
- General ideas:
  - Early Phase RE
  - Understanding "WHY?"
  - Modelling stakeholder interests and organisational perspectives
- Basic concepts:
  - Intentional actors and intentional relationships
  - Dependencies among actors
  - Internal goals of actors

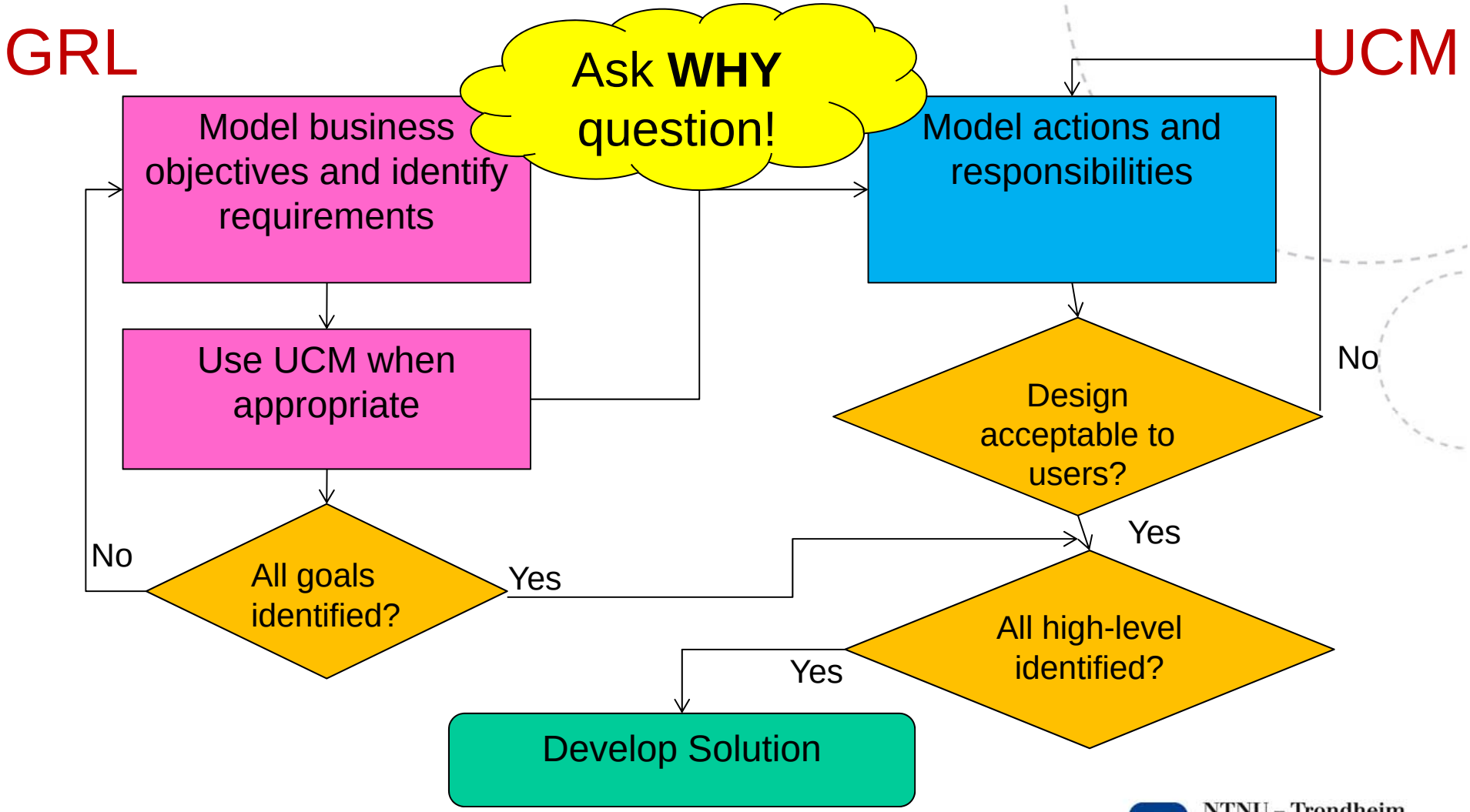
# i* Basic Concepts

- Actor
  - Perform task with a purpose (intentional)
  - Have goals, skills, responsibilities
  - Is dependent on other actors to achieve own goals
- Dependency in relation to
  - Resource (must get from another actor)
  - Task (that another actor must perform)
  - Goal (that another actor must achieve)
  - Soft-goal (that another actor must achieve)
- The above concepts are modelled in a Strategic Dependency Model (SD) and a Strategic Rationale Model

# Main Concepts in GRL

- **Actor:** an active entity that carries out actions to achieve its goals.
- **Agent**: an actor with concrete, physical manifestations, such as a human, or a machine.
- **Goal**: to depict business objectives and system requirements (functional and non-functional).
- **Tasks**: to represent different ways to achieve goals.
- **Means-end reasoning**: to explore alternative solutions.
- **Social context**: modelled in terms of dependency relationships among the agents.

# GRL and UCM: Parallel & Interactions



# UCM Central Concepts

- Central concepts
  - Start points (preconditions, causes)
  - End points (postconditions, effects)
  - Responsibilities (tasks to be performed)
  - Components (objects in the system)
  - Use case path: connect start points, responsibilities and end points
  - Decomposition
  - Control-flow: OR-join. OR-fork, AND-join, AND-fork, timer, abort, failure points, shared responsibilities

# A03: Maiden et al. Model-driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study

# A03 (Maiden et al)

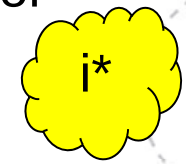
- i* and other techniques used in a large project: Air traffic Management (ATM)
- Motivation:
  - Large socio-technical systems needs to be analyzed from a number of perspectives.
  - Need to use different modelling techniques.
  - Scalability  of techniques.
  - It must be possible to synchronize the models
    - Conflicts, omissions, ambiguities (semantic quality of the overall model)
    - Compare to the need for synchronization of other models (requirements, design, code)
- The paper present an overall method RESCUE (Requirements Engineering with Scenarios for User-Centred Engineering)
  - Usage of several modelling techniques in concert
    - Use cases, i*, human activity modeling, requirements statements

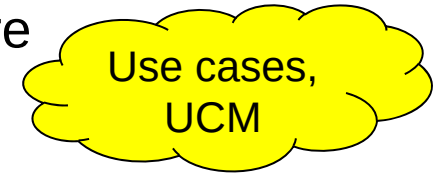
# Example Case: DMAN

- To specify operational requirements for a system for scheduling and managing departures from European major airports.
- DMAN is a complex socio-technical system involving a range of human actors (tower controllers, pilots, etc.) interacting with computer-based systems (related to aircraft, air movement and supporting aircraft movement).

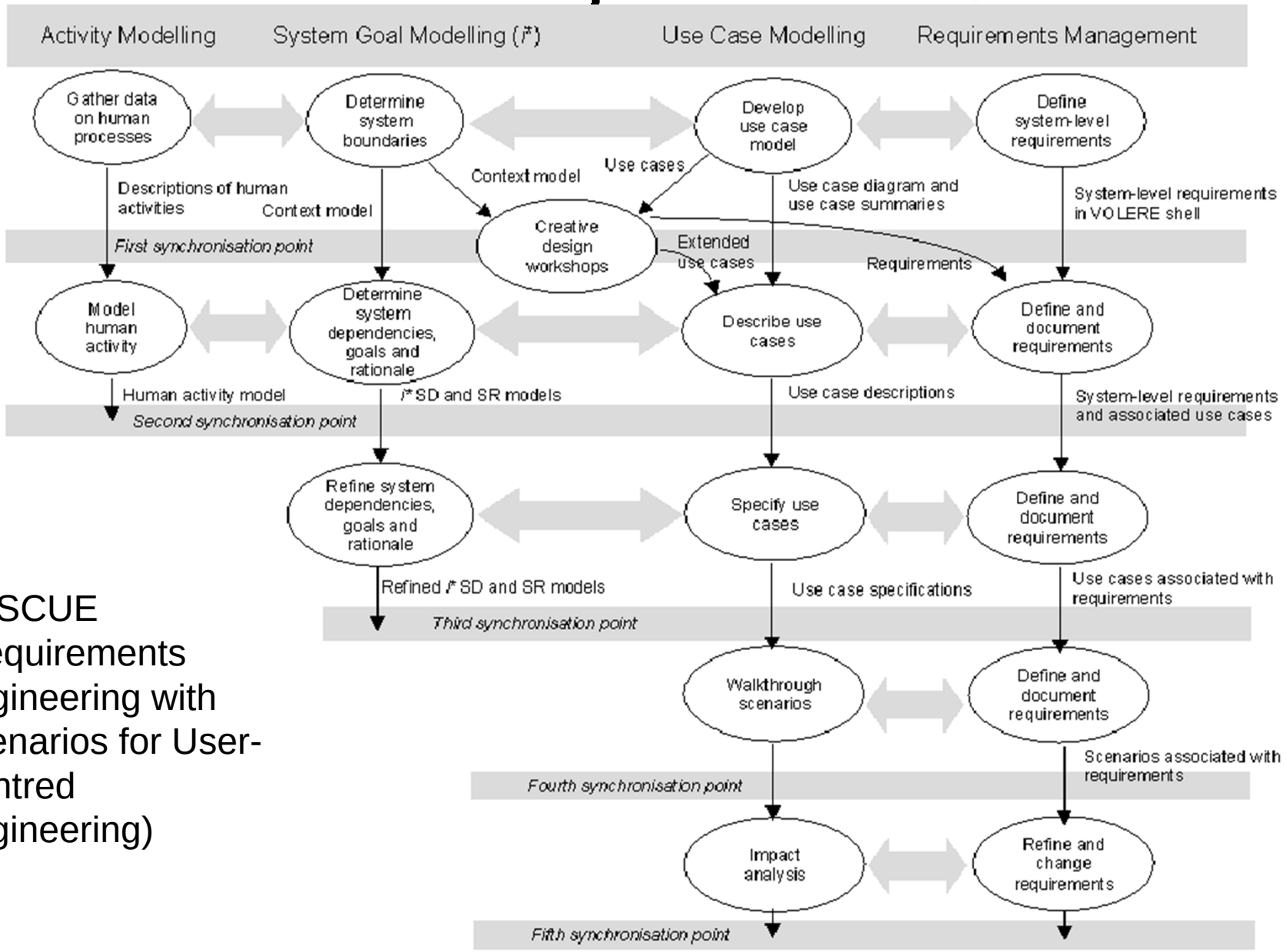
# Different modelling streams

- Human activity modelling: provides understanding of how people work.
- System Modelling: to model the future system boundaries, actor dependencies and system goals.
- Use case modelling and scenario-driven walkthroughs: enables effective communication with stakeholders and acquire complete, precise and testable requirements.
- Managing requirements: to handle the outcomes of the other 3 streams effectively.



# The RESCUE process




RESCUE (Requirements Engineering with Scenarios for User-Centred Engineering)

# Terminology and abbreviations

- ATM: Air Traffic Management
- ATCO: Air Traffic Controller
  - Runway ATCO
  - Departure Clearance ATCO
- TMA : Traffic Manager
- CTOT: Calculated take-off time (slot time)
- DMAN: The system to be made
- TACT, A-SMGCS: Existing systems

# Air Traffic Control (1)

The Heathrow control tower, showing the departure and arrival runway controllers in the foreground, other tower controllers in the background, and a chute along which flight strips pass to manage a departing flight

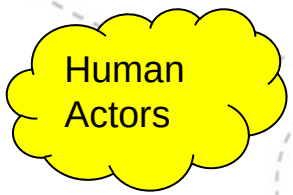
# Air Traffic Control (2)


There are many styles of strip layouts. A strip contains at least:

- Aircraft Identification (e.g. aircraft registration or a flight number)
- Aircraft Type as the relevant 4-letter ICAO designator (e.g. B744 for a Boeing 747-400).
- Level (Assigned Altitude).
- Departure and Destination.
- At least one time in four figures (other times can be shortened to minutes only).

# Human Activity Modeling

- Textual description of
  - Activities done by humans
    - Goals
    - Resources – means to achieve the goals.
    - Physical actions
    - Cognitive actions
  - How are they currently working (as-is situation)
- Step 1: collect data (unstructured)
  - Observations, scenario walkthrough, interviews, protocol analysis recordings
- Step 2: structure description



# Human activity modeling

| |
|---|
| **Goals:** Decision made to when the next aircraft can line up, Pilot given line-up clearance, Strip positioned correctly in the bay, LVP or MDI procedures adhered to, if in effect |
| **1. Departure/Air controller decides which aircraft can next line up and when**<br>Resources - strip<br>Physical actions - touch strip, look at airfield, aircraft, holding point and runway, move to look out of window<br>Cognitive actions - read strip information, validate visually, recognise aircraft and match with strip, recognise when it is appropriate to give line up clearance, formulate aircraft line up clearance sequence, understand current airspace, runway and capacity situation |
| **2. Runway ATCo calls Pilot and gives line up clearance**<br>Resources - strip, radio, headset<br>Physical actions - touch strip, flick radio transmission switch, look aircraft, runway and holding point, move to look out of window<br>Communication - talk to pilot, issue clearance, provide information<br>Cognitive actions - read strip information, validate visually, |

Actions

**Fig. 2.** Part of the DMAN Human Activity Model.

# System Modelling: use of i*

- System modeling in RESCUE includes 3 analyses and 3 different models:
  - Context diagrams, possible boundaries of (start of i* modelling: actors)
    - Automation
    - Changed work system
    - Other systems that are influenced
    - Systems that are not included, but interacts with the new system
  - Strategic Dependency Model (SD-Model)
    - RESCUE: challenge the developers in asking questions on the system boundaries by stating these as dependencies
    - 15 actors, 46 dependencies
  - Strategic Rational Model (SR Model)
- Give input to scenarios and requirement management

# System Modelling: i*

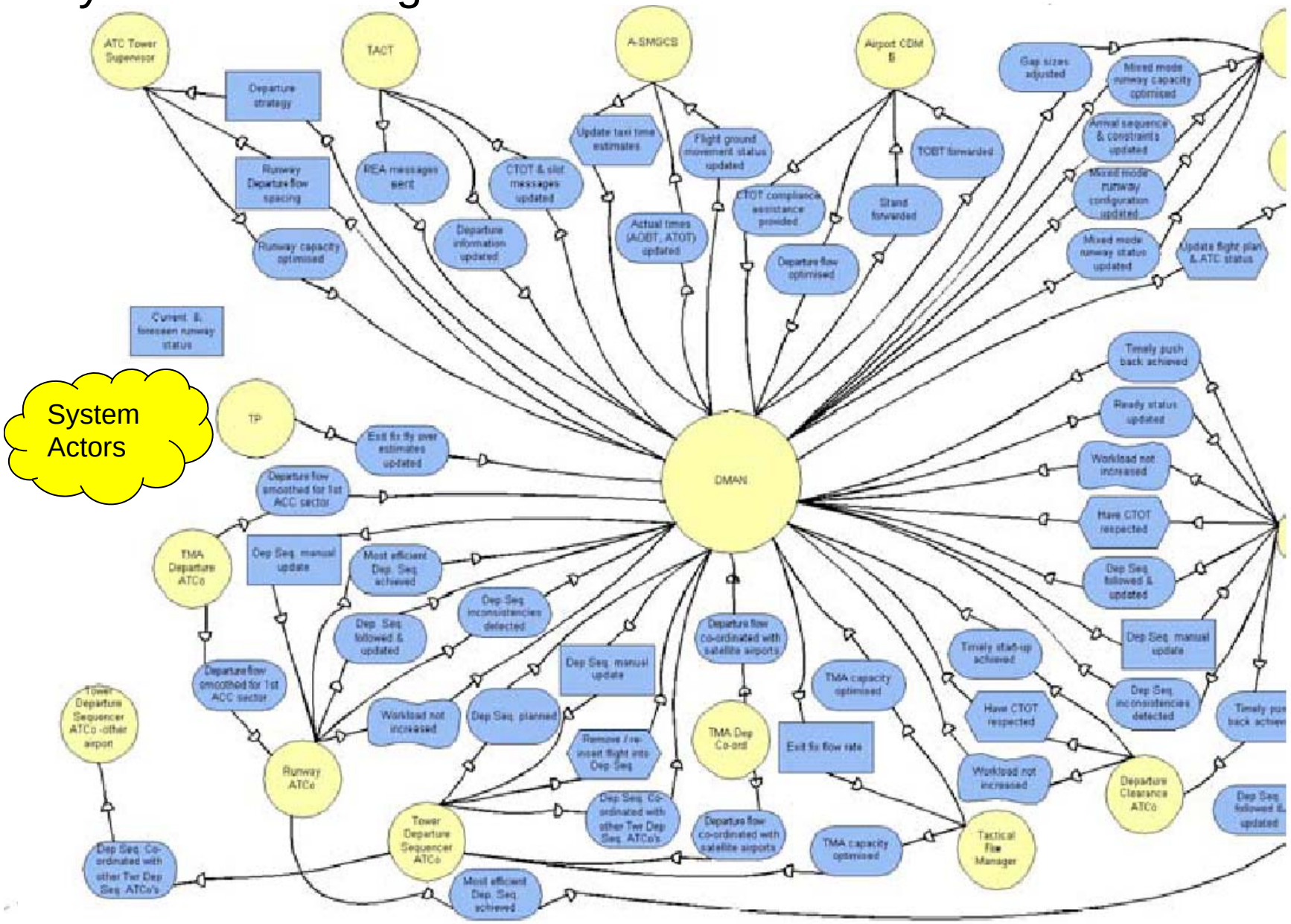
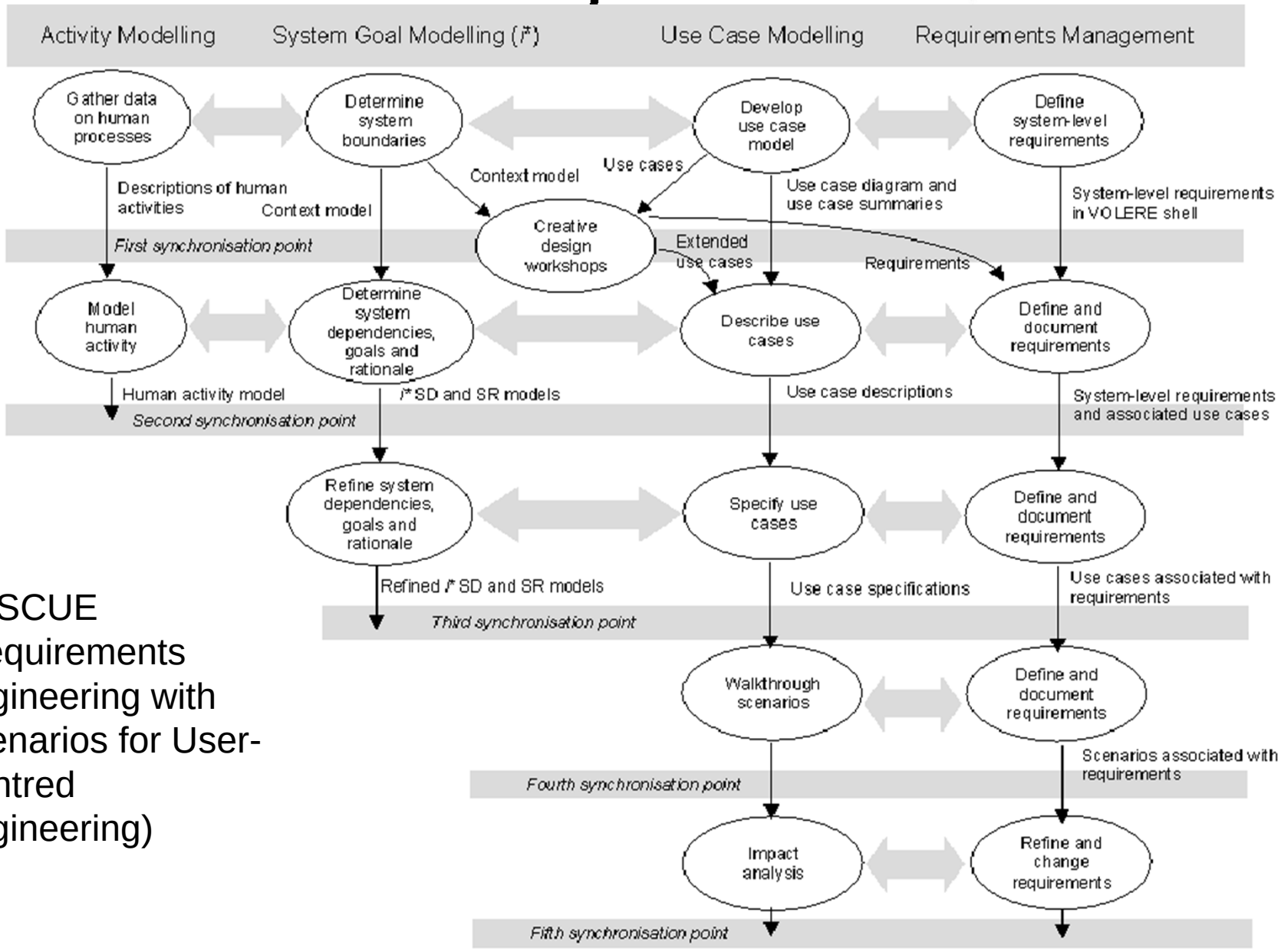



**Fig. 3.** Part of the SD Model for DMAN.

# Use cases modelling

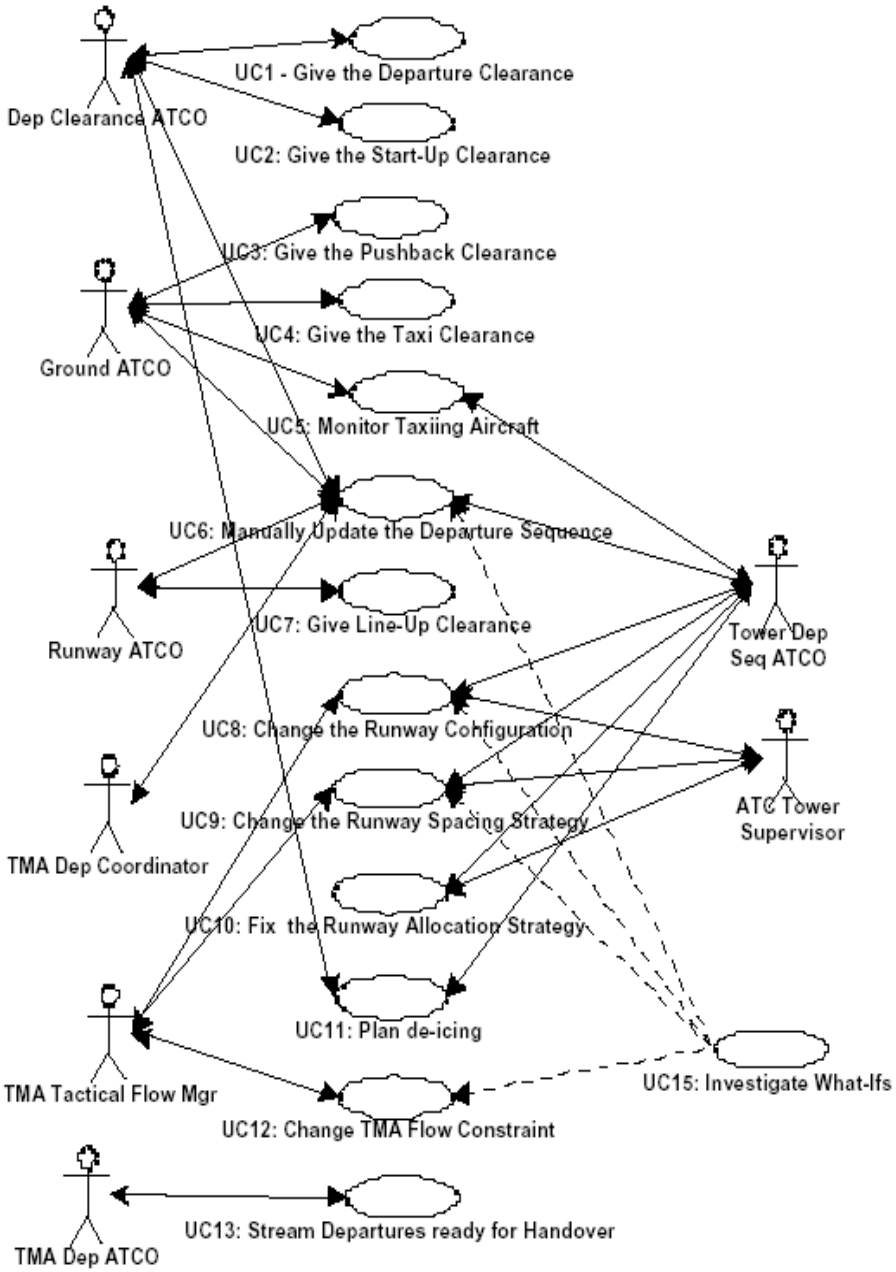
- Create use case diagrams (using input from the context diagrams).
- Write use case scenarios – descriptions using output from the i* SR models.
- Generate and walk through scenarios
  - To acquire stakeholder requirements that are complete, precise and testable.

# The RESCUE process




RESCUE (Requirements Engineering with Scenarios for User-Centred Engineering)

# Use Cases


Dep Clearance ATCO
UC1 - Give the Departure Clearance
UC2: Give the Start-Up Clearance
Ground ATCO
UC3: Give the Pushback Clearance
UC4: Give the Taxi Clearance
UC5: Monitor Taxiing Aircraft
UC6: Manually Update the Departure Sequence
Runway ATCO
UC7: Give Line-Up Clearance
Tower Dep Seq ATCO
UC8: Change the Runway Configuration
ATC Tower Supervisor
TMA Dep Coordinator
UC9: Change the Runway Spacing Strategy
UC10: Fix the Runway Allocation Strategy
UC11: Plan de-icing
TMA Tactical Flow Mgr
UC15: Investigate What-Ifs
UC12: Change TMA Flow Constraint
TMA Dep ATCO
UC13: Stream Departures ready for Handover

# Use Case Descriptions

| | UC9 Change the Runway Spacing Strategy |
|---|---|
| Date | 16 October 2003 |
| Source | Stage 1 Document |
| Actors | ATC Tower Supervisor, Tower Departure Sequencer ATCO, TMA Departure Co-ordinator, AMAN, Tower Departure Sequencer ATCO (other airports), ATC Tower Supervisor (other airports), Tactical Flow Manager. |
| Problem State-ment (now) | Integrate runway spacing strategy into departure planning process |
| Triggering Event | An imbalance between arrival and departure delay is predicted |
| Assumptions | We assume that the runway spacing is defined by the number of take off and landing per hour for each runway. |
| Successful End States | A time to implement new runway spacing is agreed by ATC Tower Supervisor and TMA Departure Coordinator. The DMAN departure sequence takes into account the change in runway allocation. |
| Unsuccessful End States | DMAN departure plan does not allow for runway spacing change at the correct time. |
| Normal Course | 1. The ATC Tower Supervisor looks at the predicted arrival delays in AMAN |
| | 2. The ATC Tower Supervisor looks at the predicted departure delays in DMAN |
| | 3. The ATC Tower Supervisor considers the predicted arrival and departure delays |
| | 4. The ATC Tower Supervisor decides that a different spacing strategy would be preferable. |
| | 5. Abstract Use Case 14(the ATC Tower Supervisor performs "What-Ifs" with DMAN) |
| | 6. The ATC Tower Supervisor contacts the TMA Departure Coordinator by telephone |
| | 7. The ATC Tower Supervisor states the proposed new runway spacing strategy. |
| | 8. The TMA Departure Coordinator agrees the new spacing strategy. |
| | 9. AMAN re-plans arrivals taking into account the new runway configuration |
| | 10. AMAN notifies DMAN that the re-planning of arrivals is complete |
| | 11. CALL UC6 (DMAN Updates the Departure Sequence) for this airport. |

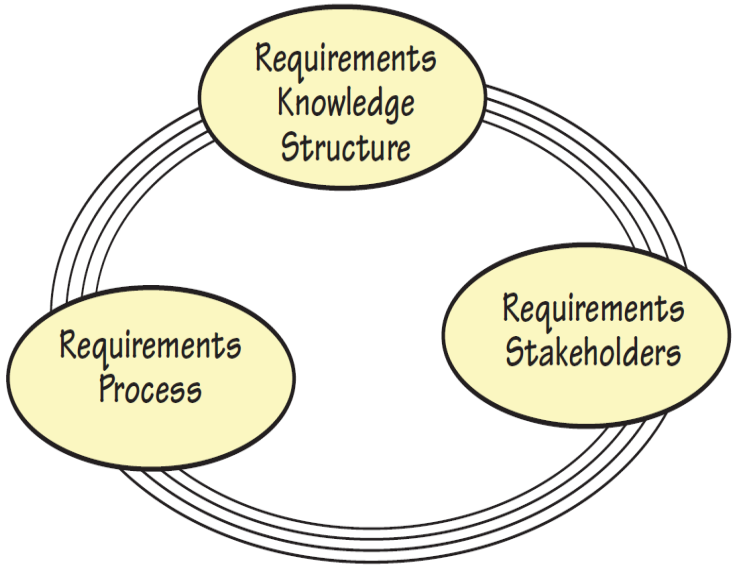
**Fig. 4.** Part of a draft DMAN use case description for UC9: Change the runway spacing strategy.

# Requirements management

- Collect and structure (textual) requirements
- Use VOLERE template
  - Ensure that requirements are categorized and is testable based on its category
- Requirement can be connected to
  - The whole system
  - A use case
  - A step in a use-case

# Volere: Basic Idea

The *Volere requirements techniques were developed to answer the need for a* common language for discovering requirements and connecting them to solutions.
Considers RE as a socio-technical discipline.



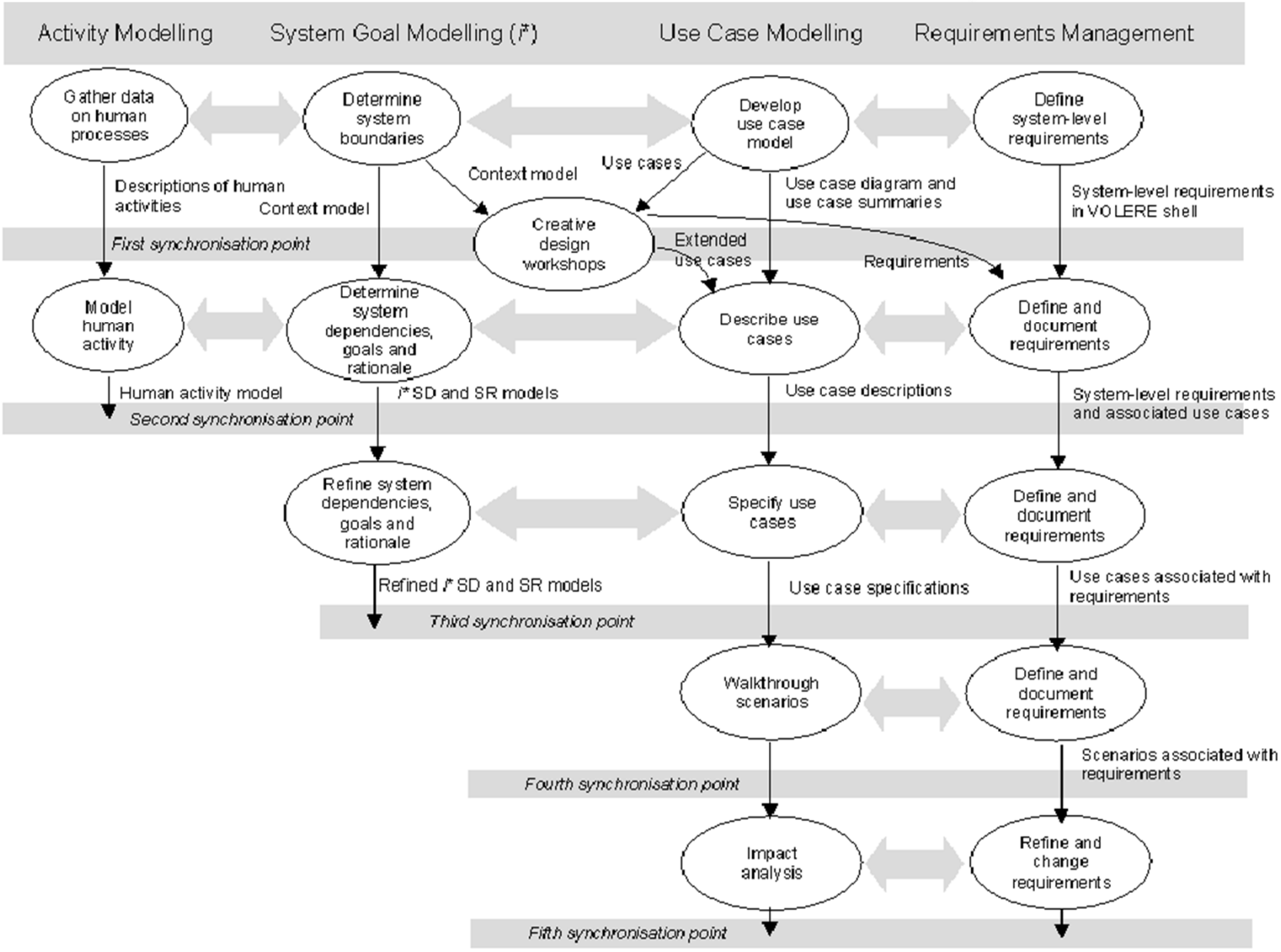

There are three groups of interconnected components.
- The Requirements Knowledge Structure is concerned with how different items of requirements knowledge relate to each other and how to trace requirements from one level to another.
- The Requirements Process is concerned with procedures and activities for how to discover, populate and disseminate the requirements knowledge.
- The Requirements Stakeholders are the input for determining how much of the knowledge structure needs to be populated for each particular project. The project team uses knowledge of the stakeholders to determine the order in which the work needs to be done and the appropriate level of detail.

*Ref: http://www.systemsguild.com*

# Model synchronization

- 5 Synchronization points
  1. Boundaries point
  2. Work allocation point (actors and dependencies)
  3. Generation of scenarios point (goals and tasks identified)
  4. Coverage point (All scenarios gone through, with stakeholders)
  5. Consequences point (all impacts of the system is analysed)
- Use a combined meta-model
  - Show the mapping between different concepts in different modeling language (i*, UML/use cases, human activity)
- Checklists
  - Describe expected connections between models

# The RESCUE process - revisited



# RESCUE Concept Meta-model

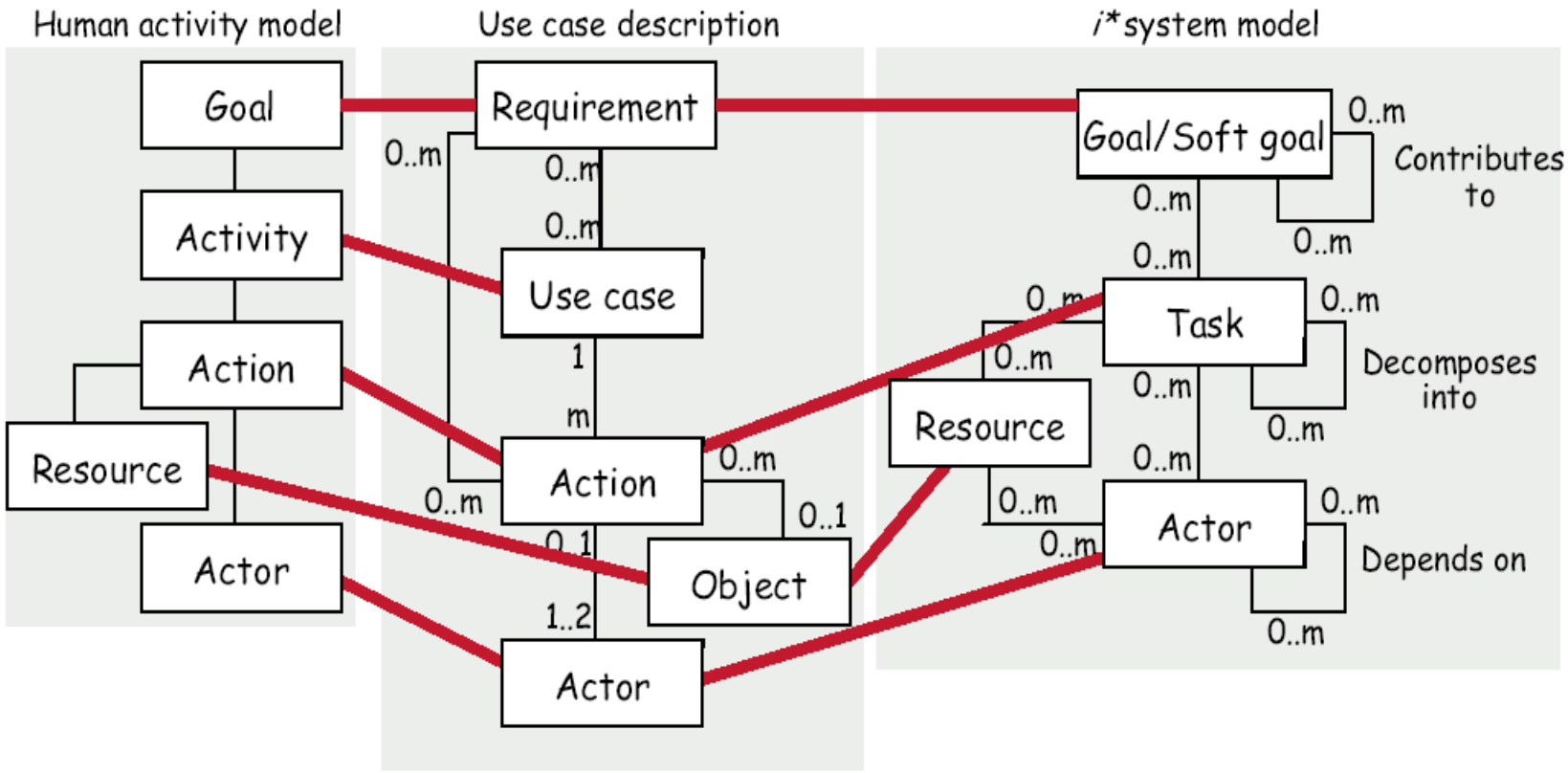



**Fig. 5.** RESCUE concept meta-model as a UML class diagram showing mappings between constructs in the 3 model types.

# Checklists – Stage 1

| | |
|---|---|
| Check 1.1 | Every major human activity (e.g. applying resolutions) should correspond to one or more use cases in the use case model. |
| Check 1.2 | Every actor identified in human activity modelling is a candidate actor for the context model. |
| Check 1.3 | Every adjacent actor (at levels 2, 3 or 4 of the context model) that communicates directly with the technical system (level 1 in the context model) should appear as an actor in the use case diagram. |
| Check 1.4 | The system boundary in the use case diagram should be the same as the boundary between levels 1 and 2 in the context model. |
| Check 1.5 | Services and functions related to use cases in the use case model should map to system level requirements, i.e. high-level functional and non-functional requirements, in the requirement database. |

Data about human activities and the extended context model are used to check the completeness and correctness of the use case model.

# Discussion

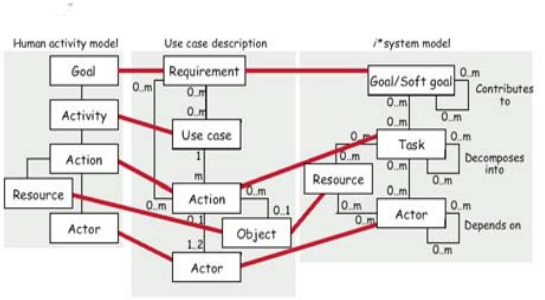



Fig. 5. RESCUE concept meta-model as a UML class diagram showing mappings between constructs in the 3 model types.

- Synchronisation points and checklists can help identify a number of different types of issues.
- Can you think of some?
  - Example issues:
    - Dependencies that were missing in the SR model compared to the SD model.
    - Human activity model against i*: tasks undertaken by humans, described incorrectly, actors not represented in a use case.
    - Elements missing from use case descriptions (check against SR model)

# Experiences from the case

(Maiden et al.)

- Used RESCUE in connection to requirements specification of an Air Traffic Management system
- 9 month process!
- Synchronization points discovered a number of issues
- A number of these resulted in improvements of the requirements specification

- Conclusion
  - It is possible to use i* on large projects (with proper tool support, and a sensible method)
  - The RESCUE-method, with synchronization points, revealed many issues that might have been difficult to find otherwise
  - A lot to gain by combining several techniques

# Summary, from i* to other models (A01, A02, A03)

- i* is primarily for early RE/analysis
  - Actors, actor dependencies
  - Goal/task hierarchies
  - To understand the problem domain and the organization
  - Both for modeling as-is and to-be situations
- Can be connected to
  - 'Prior' informal models (Human Activity Model)
  - Later requirements specification (use cases/VOLERE requirements template)
  - Evaluation of design-alternatives (Use Case Maps)
- More experiences needed

# Discussion

- Different approaches for combining different perspectives
  1 Use many different languages in parallel without explicit tool-integration (early CASE-tools, Visio)
  2 Integrate existing languages (UML, EEML)
  3 Develop completely new approaches (i*)
  4 Develop framework to be able to create and adapt languages on an as-needed basis (e.g. METIS)

# Discussion, w.r.t assignment

- Can different (existing) modelling languages be used?
  - Which ones? And why do you choose them?
  - What are the strengths and weaknesses of each language?
  - Why would you use different languages?
- Would you try to integrate them or use them separately?
  - You could explain how you could connect them?
- Would you design your own language (meta-modelling)?

# Next Lecture

- Process Modelling
  - Functional perspective: SADT/ IDEF0, SeeMe
  - Combining Process and other aspects of organisations
  - Process Modelling in Metis