# Socio-Technical Security Modelling Language

Elda Paja

ANIKETOS

12 December 2012

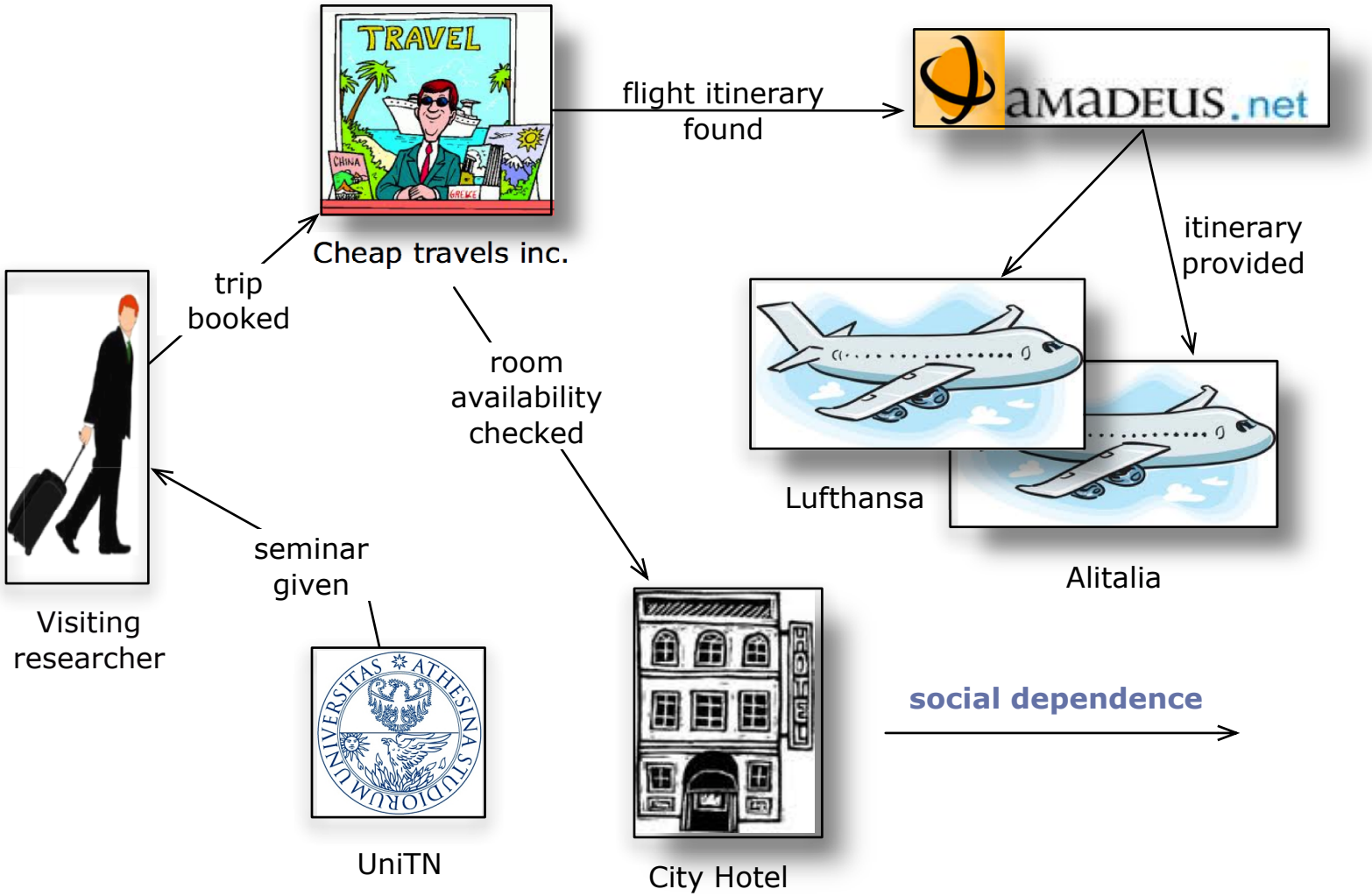
SEVENTH FRAMEWORK PROGRAMME

# Socio-Technical Systems (STS)

- An interplay of different subsystems
  - Not only technical, but also humans and organisations
  - Each subsystem is autonomous
  - Defined in terms of interaction among subsystems
    - Each subsystem needs to socially rely on others to fulfill its objectives

- Examples include
  smart homes, e-commerce sites, eHealth systems, etc.

# An example of STS



# The Security Problem in STS

- Interaction is everywhere!
  - Technical Systems – Technical Systems
  - Technical Systems – Social Actors
  - Social Actors – Social Actors
- Social aspects are a main concern
  - Decentralized setting: no controlling authority
  - Autonomy: security cannot be enforced
- Key idea: social contracts to constraint interaction
  - Social dependence
  - Information exchange

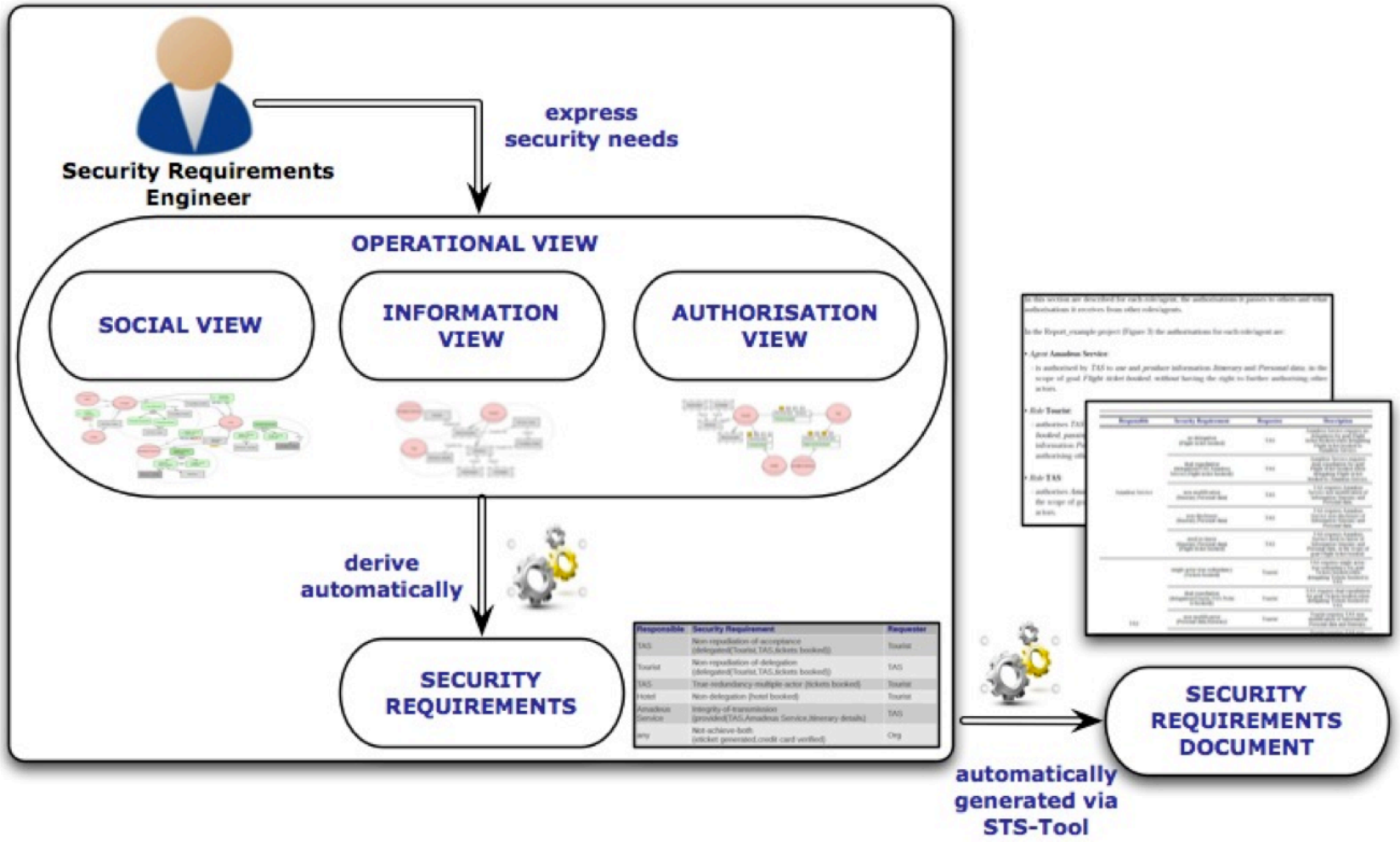
# Socio-Technical Security Modelling Language (STS-ml)


- Role – and goal – oriented requirements modelling language

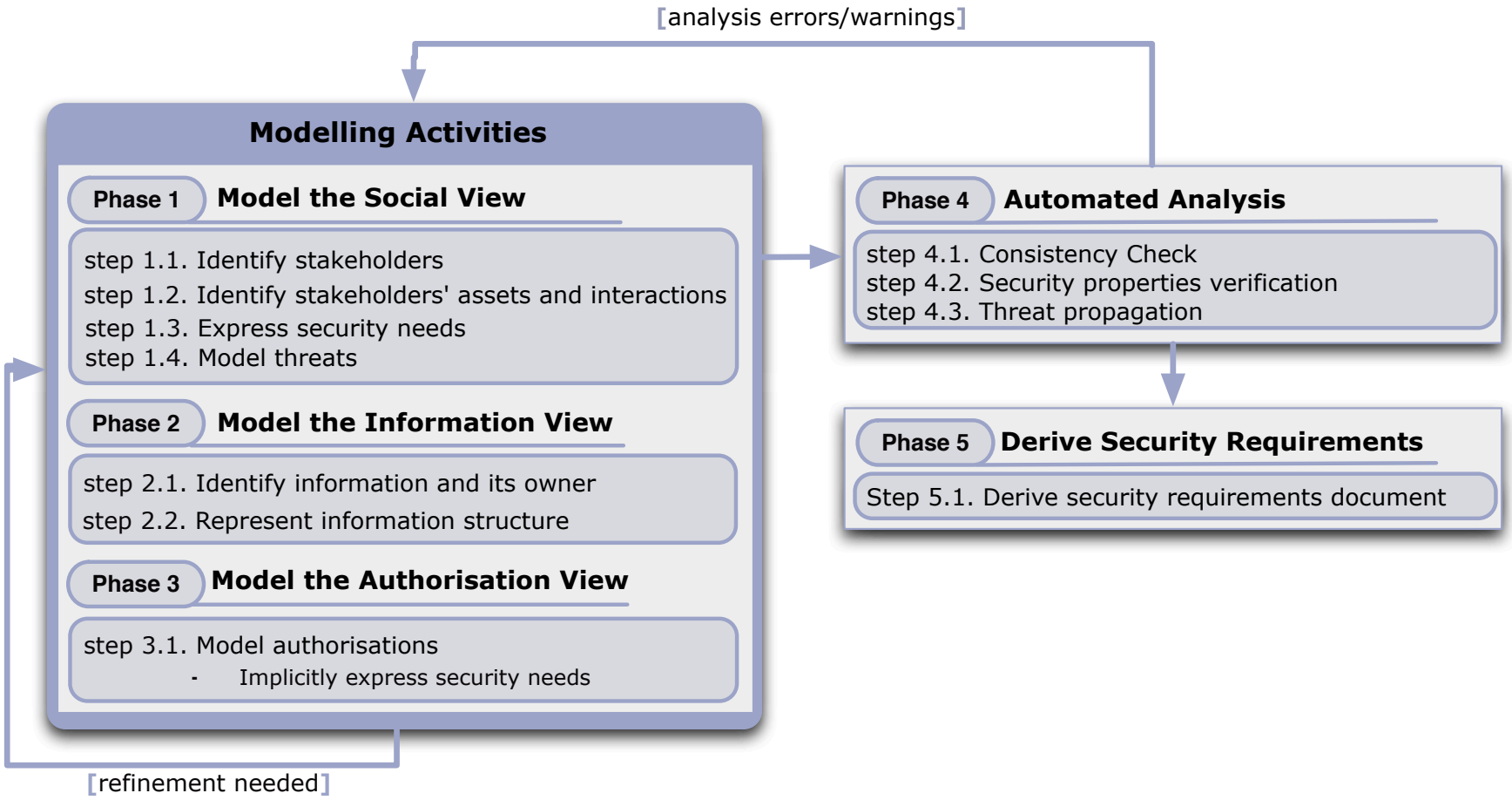
- Models are built diagrammatically
  - Graphical concepts and relations are used to create the models
  - Multiple views, each focusing on a specific perspective

- Allow actors to express constraints (security needs) over interactions
  - Social dependence (goal delegation)
    - E.g.: visiting researcher depends on the cheap travel inc. to book the hotel and flight tickets and he requires it not to deny having accepted the delegation
  - Documents exchange
    - E.g.: visiting researcher wants the cheap travel inc. to use his personal data information strictly to book the hotel and flight tickets, but not for any other purposes

# STS-ml: outline

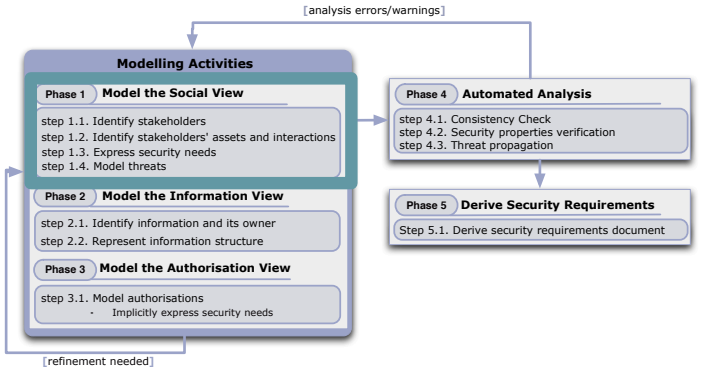


# The STS-ml method

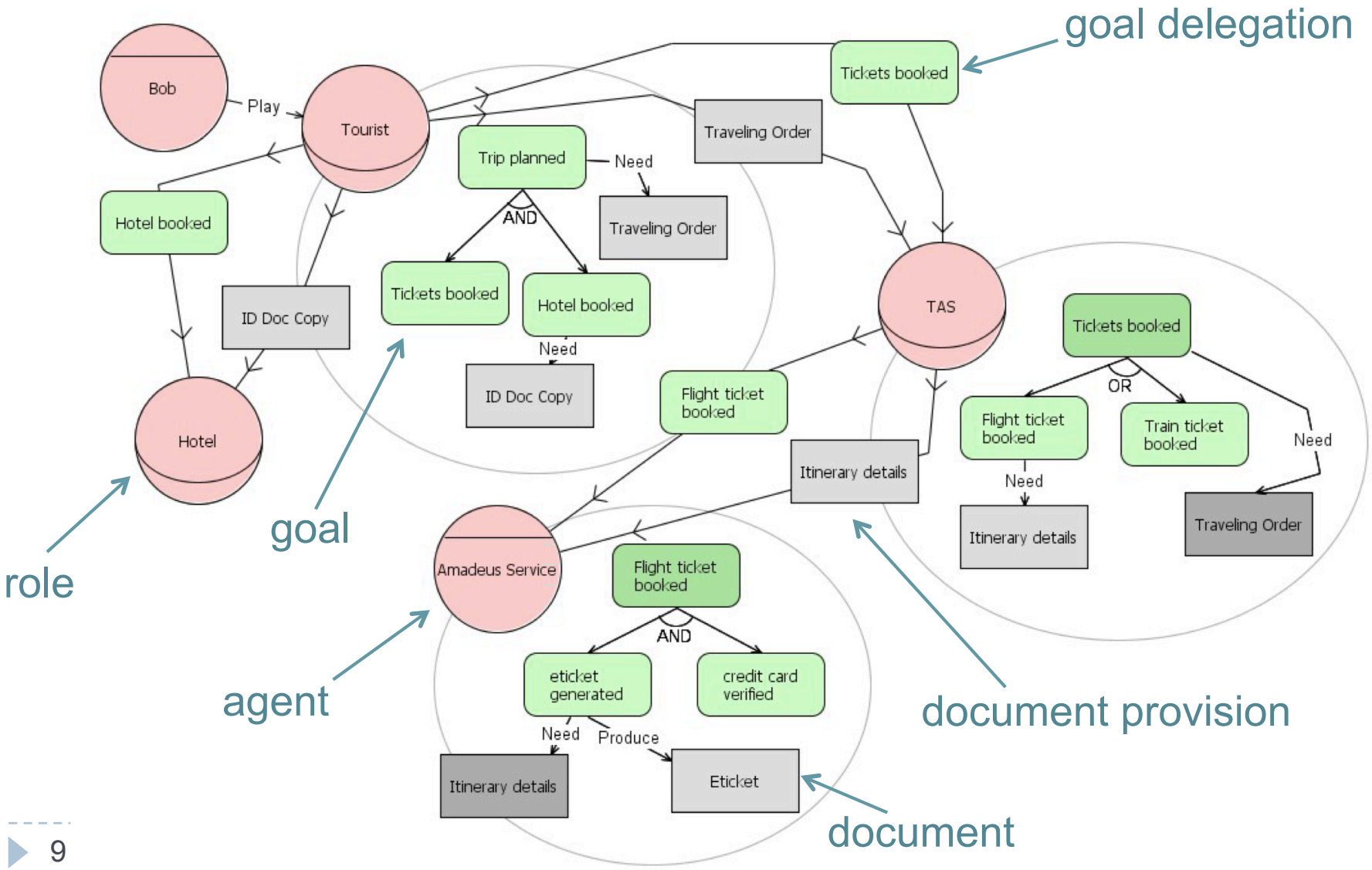


# Phase 1. Modelling the Social View

- Step 1.1 Identify stakeholders
  - Agents and roles
- Step 1.2 Identify assets and interactions
  - Assets: goals, documents
  - Interactions: goal delegations and document provisions
- Step 1.3 Express security needs
  - Express expectations concerning security over interactions
    - Elicited from the stakeholders
- Step 1.4 Model threats
  - Represents events threatening assets

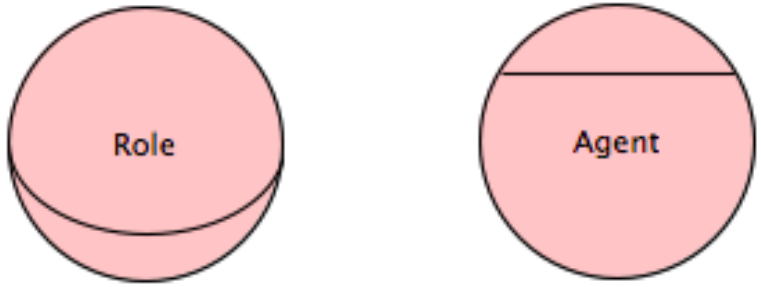


# Social view: an example



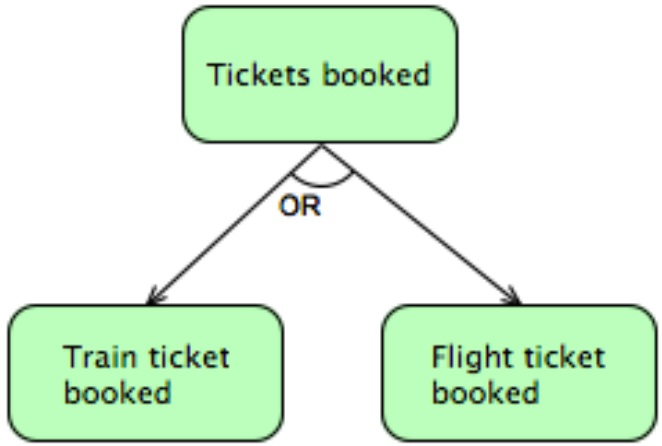
# Step 1.1. Identify Stakeholders

- Elicit roles and agents
  - Role is an abstract characterization of the behavior of an active entity within some context
    - Most participants are unknown at design time
    - e.g., Tourist, Travel Agency Service, Hotel, …
- Agents play (adopt) roles at runtime, and they can change the roles they play
  - e.g., Bob, Fabiano, CheapTravels Inc.
  - Some agents are known, e.g., Amadeus Flight Service

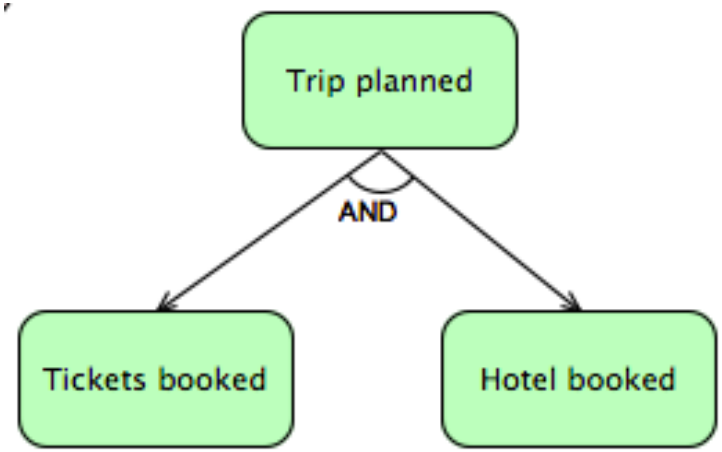


# Step 1.2. Identify assets and interactions

- A goal is a state of affairs that an actor intends to achieve
  - e.g., trip planned, flight tickets booked
  - Used to capture motivations and responsibilities of actors
- Goal can be decomposed (refined)




Or-decomposition



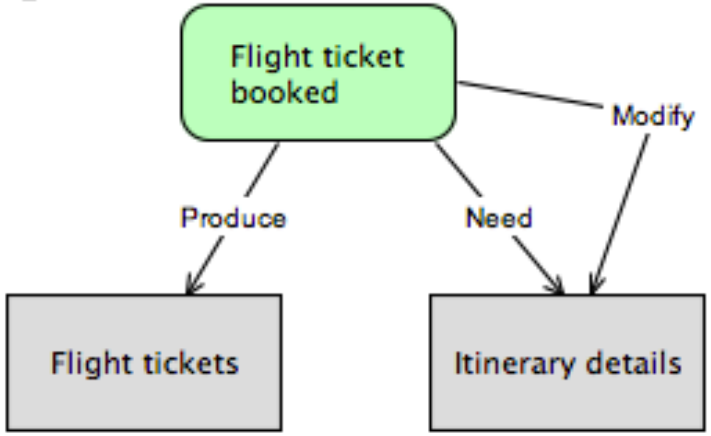

And-decomposition

# Step 1.2. Identify assets and interactions

- Goal delegation
  - A Delegator actor delegates the fulfillment of a goal (delegatum) to a different actor (delegatee)
    - Lack of capability or transfer of responsibility
  - e.g., Tourist is not capable of booking the tickets on his own, he depends on a Travel Agency Service to achieve this goal
  - In STS-ml, only leaf goals can be delegated

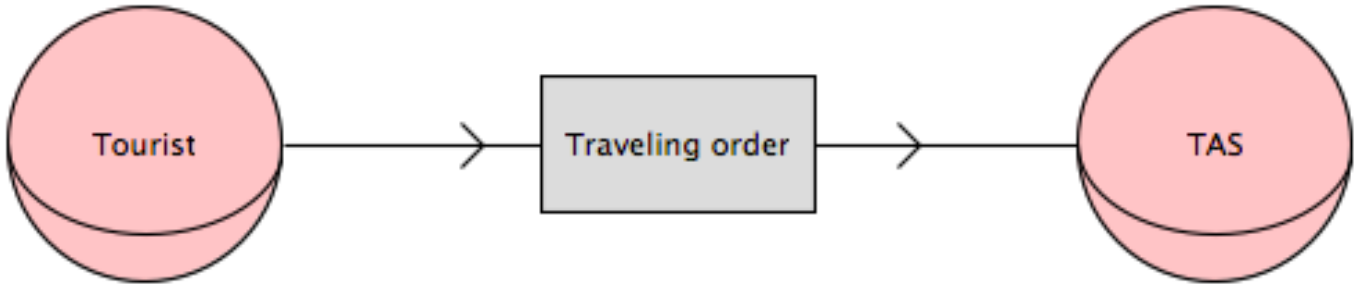
# Step 1.2. Identify assets and interactions

- A document represents an exchangeable entity which may contain some information
  - Actors possess or manipulate documents to achieve their goals
- Goal-document relationships
  - An actor may need one or more documents to fulfill a goal
  - An actor may produce documents while fulfilling a goal
  - An actor may modify a document while fulfilling a goal

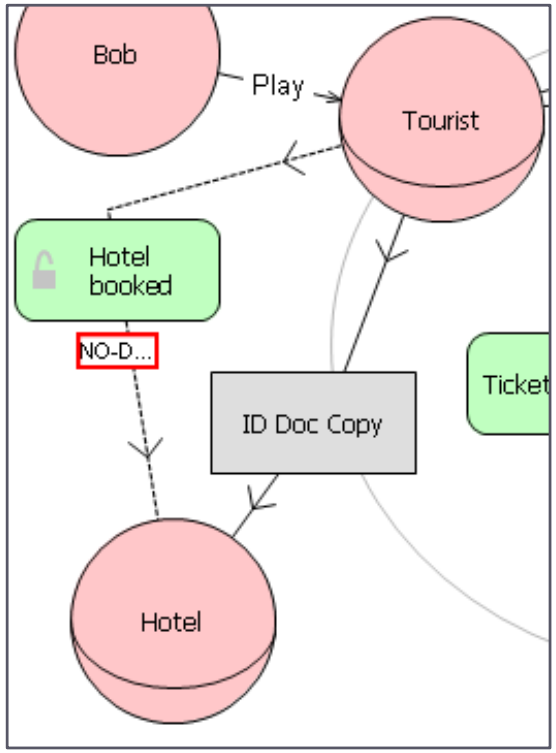


# Step 1.2. Identify assets and interactions

- Document provision
  - Captures exchange of documents between a provider actor and a providee actor
  - Provider: an actor that possesses the document
  - Providee: an actor that might need documents to achieve its goals

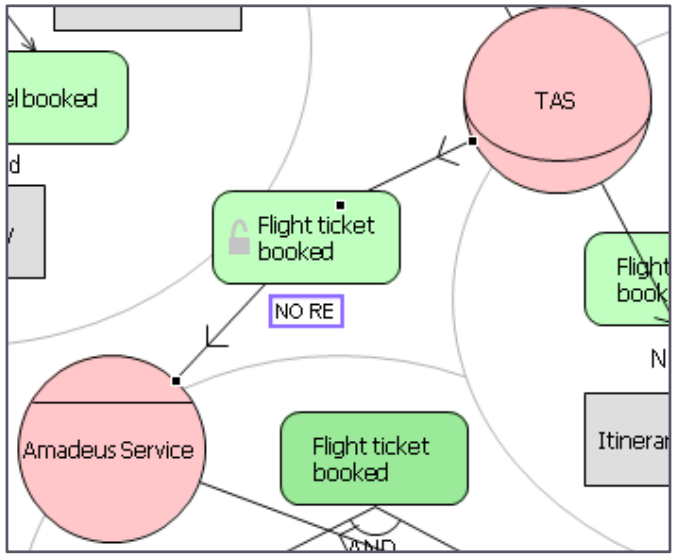


# Step 1.3. Express security needs




## Non-delegation

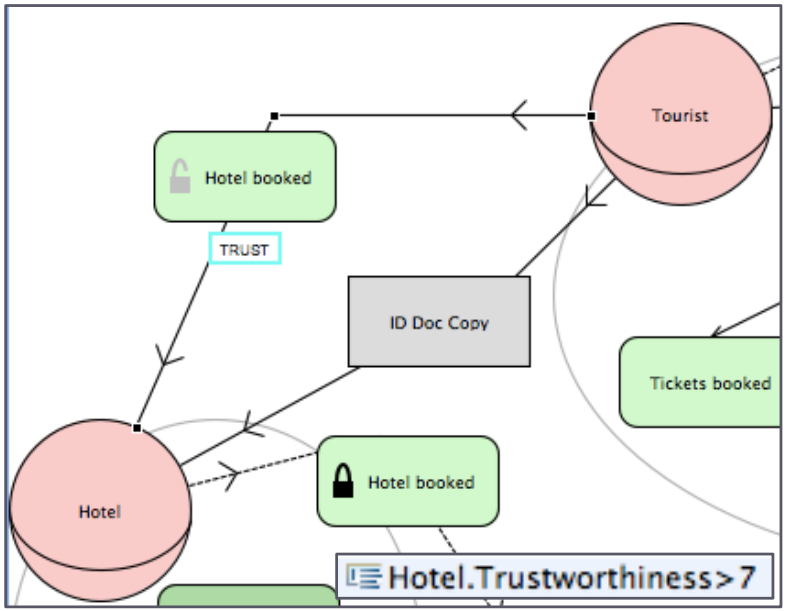
The re-delegation of the fulfilment of a goal is forbidden




## Non-repudiation

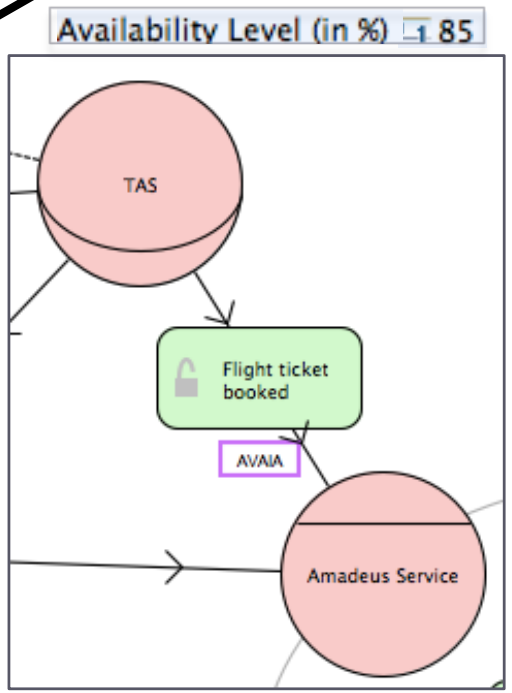
- The delegator cannot repudiate he delegated
- The delegatee cannot repudiate he accepted the delegation

# Step 1.3. Express security needs




## Min trustworthiness level

The delegation of the goal will take place only if the delegatee has a min required trustworthiness level




## Availability

The delegatee should ensure a min availability level for the delegated goal

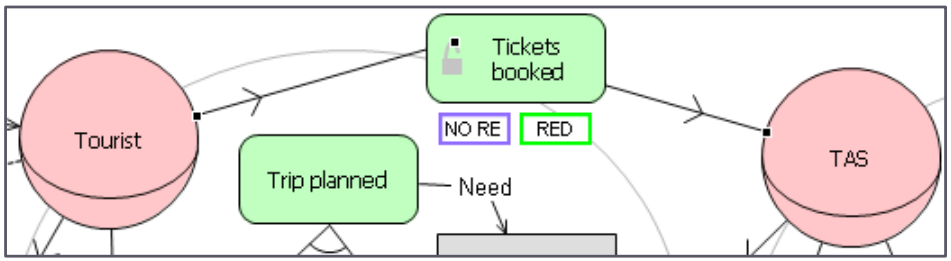
# Step 1.3. Expressing security needs

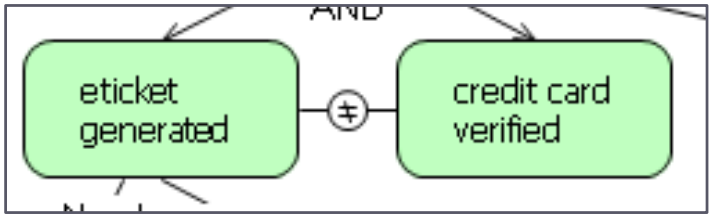
## Redundancy

- Alternative ways of achieving a goal
- Different redundancy types
  - True and Fallback
  - Single and Multi Actor

## Combine/ Incompatible

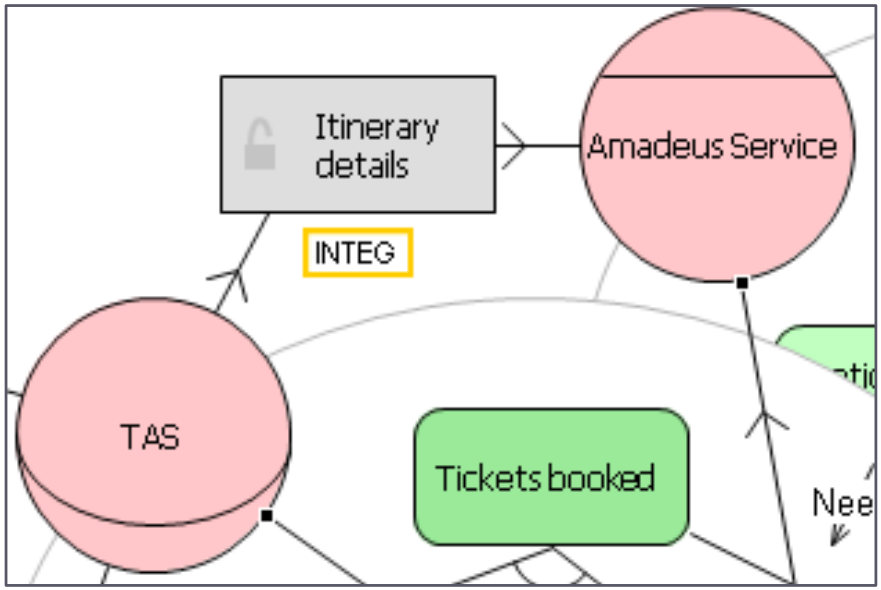
- Two goals shall be achieved by different (the same) actors
- Two roles are incompatible, i.e., cannot be played by the same agent



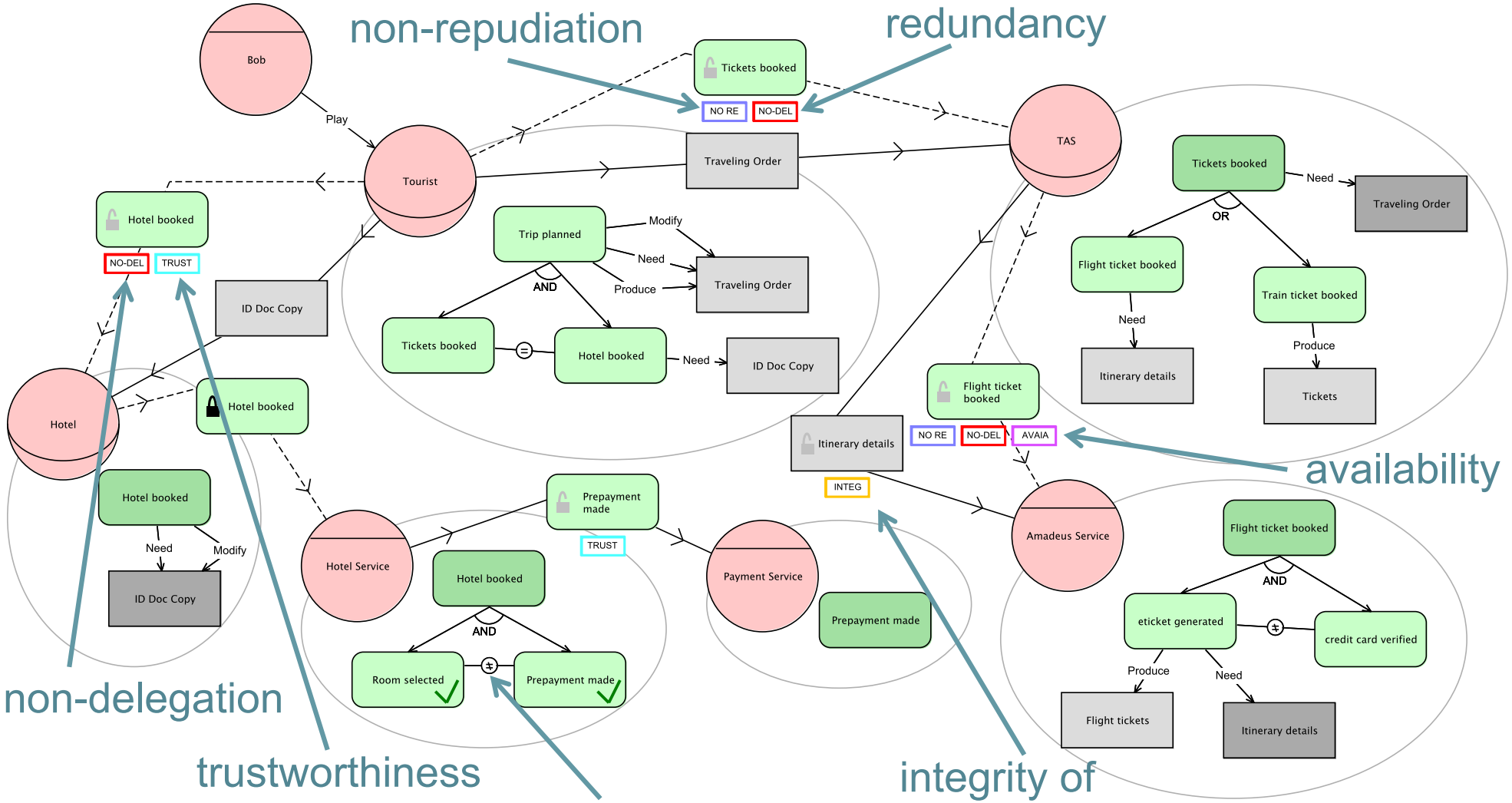



# Step 1.3. Expressing security needs

- Integrity of transmission
  - The document shall not be altered during the transmission from the sender to the receiver



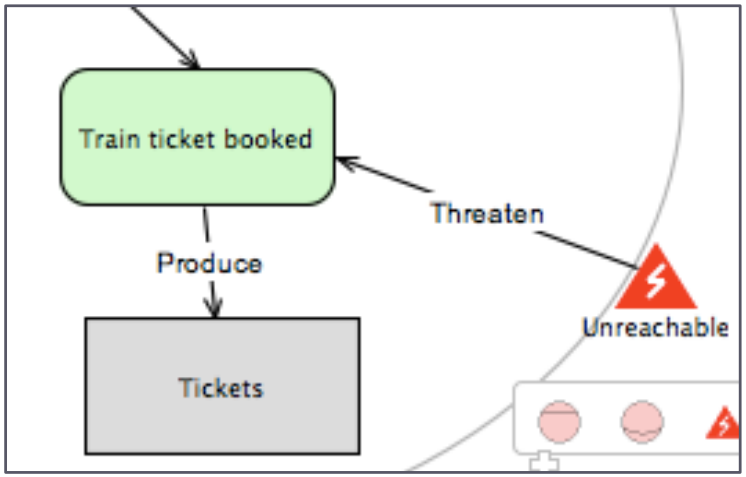
# Social view: expressing security needs

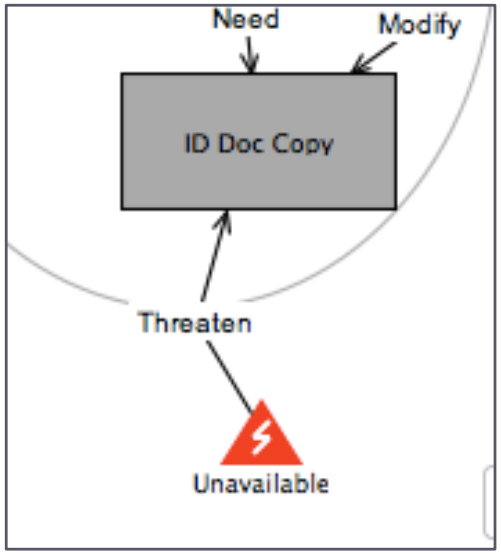


# Step 1.4. Model threats

Represent event threatening assets

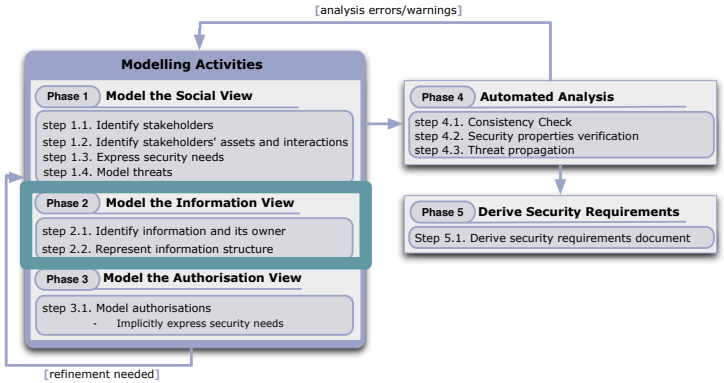
- Over goals
  - Goal cannot be reached
- Over documents
  - Document becomes unavailable



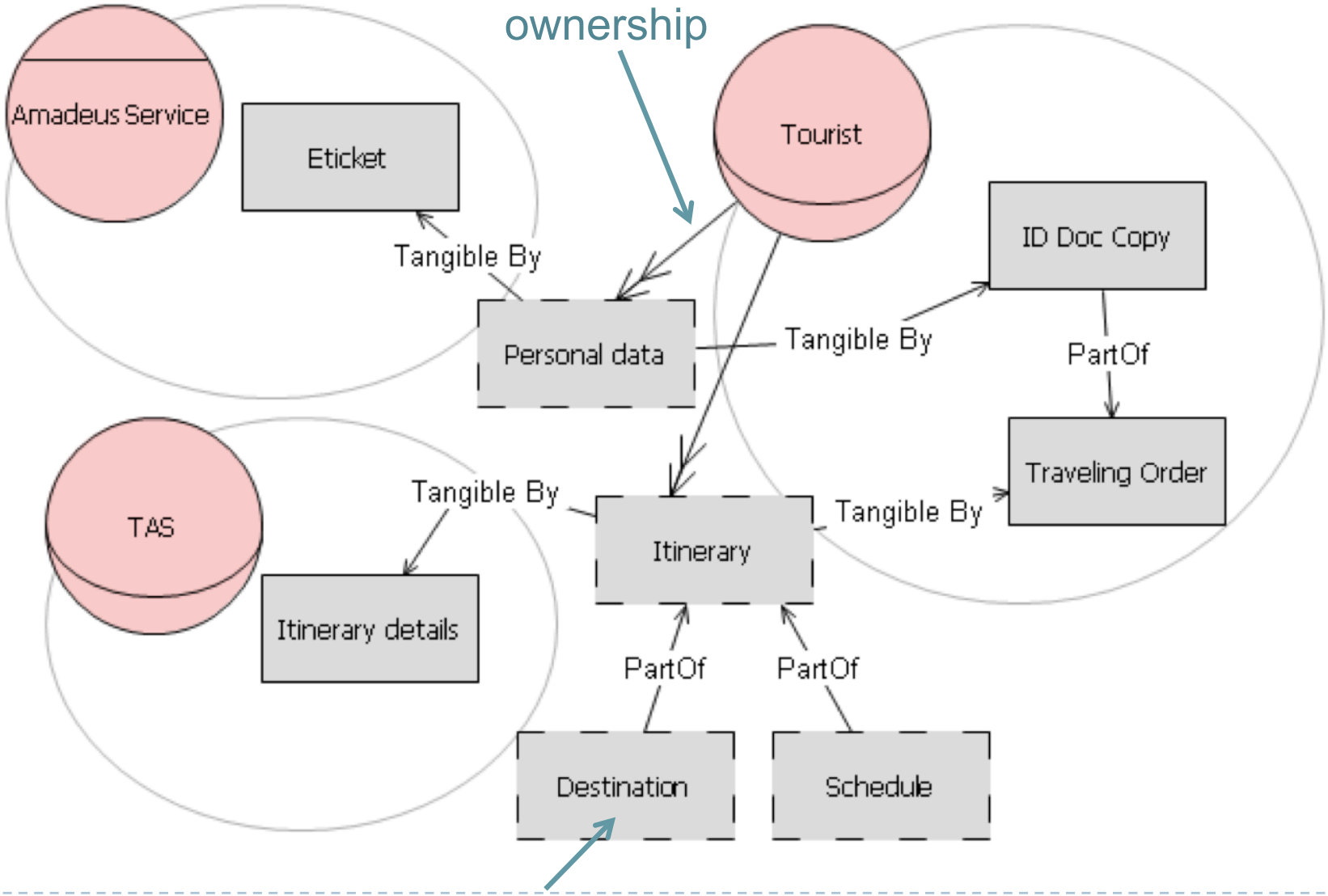



# Phase 2. Modelling the Information View

- Step 2.1 Identify information and its owner
  - Confidentiality requirements are concerned with protecting the disclosure and usage of information
  - Documents represent information
  - Represent the owners of different information
- Step 2.2 Represent information structure
  - Tangible By: information ⟶ document
  - Part Of: Info (doc) ⟶ Info (doc)

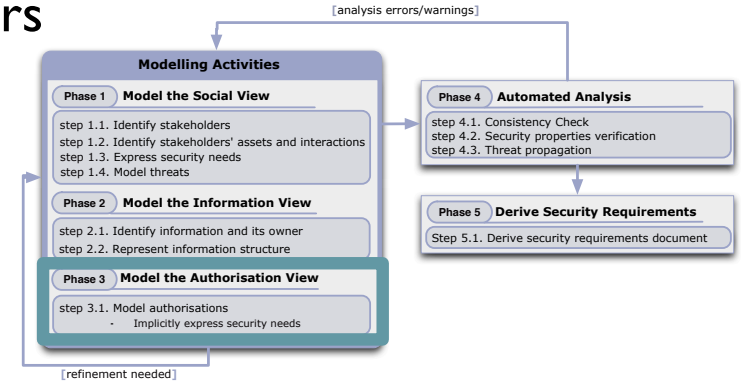


# Information view: an example

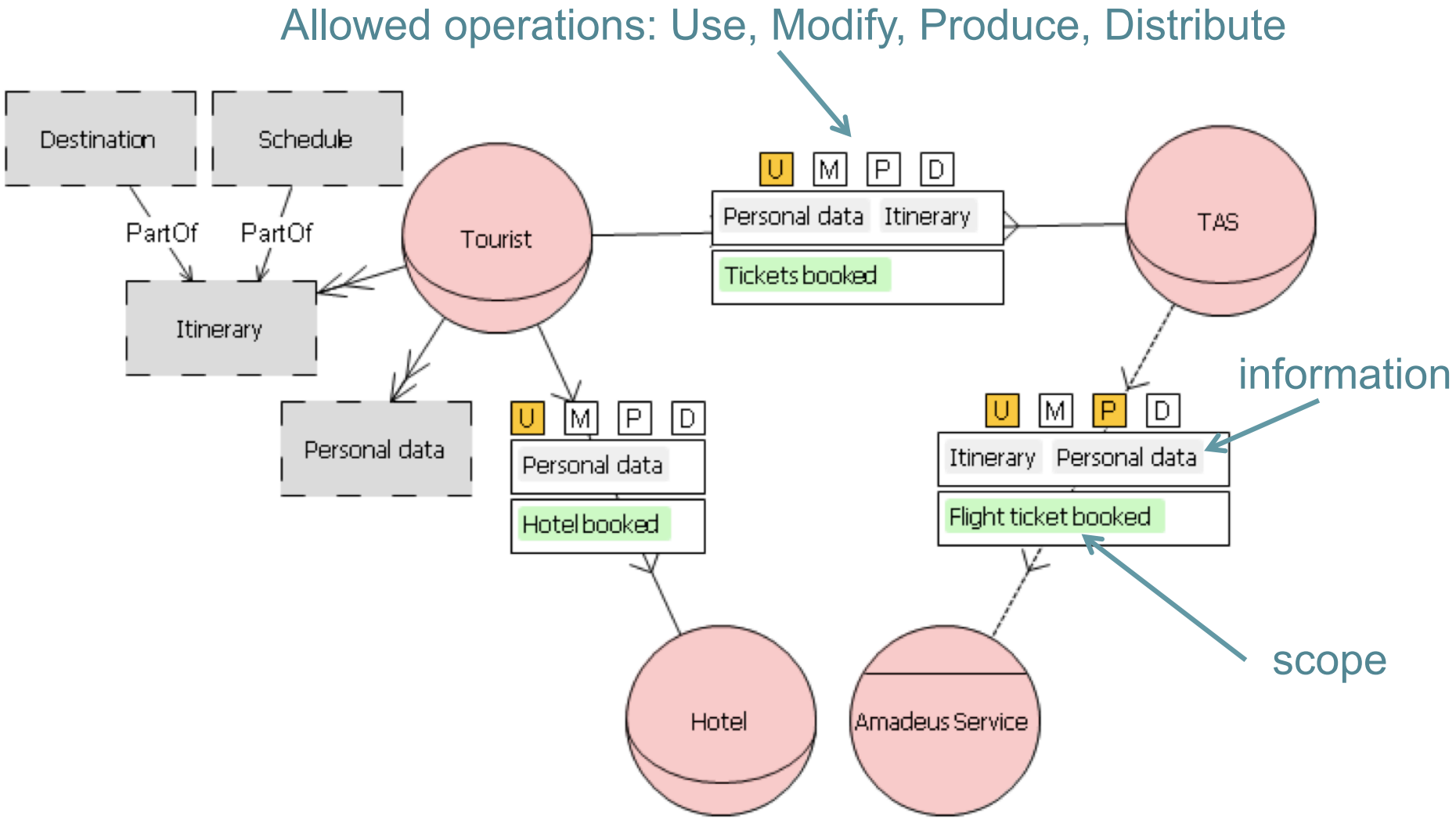


# Phase 3. Modelling the Authorisation View

- Step 3.1 Model authorisations
  - Transfer of rights/permissions between actors
- Authorisations about information, specifying
  - Scope of usage (a set of goals)
    - The customer permits the travel agency to use her personal data only to book the tickets
  - Allowed operations: use (read), modify, produce, distribute
  - Transferability
    - Further propagate rights to other actors



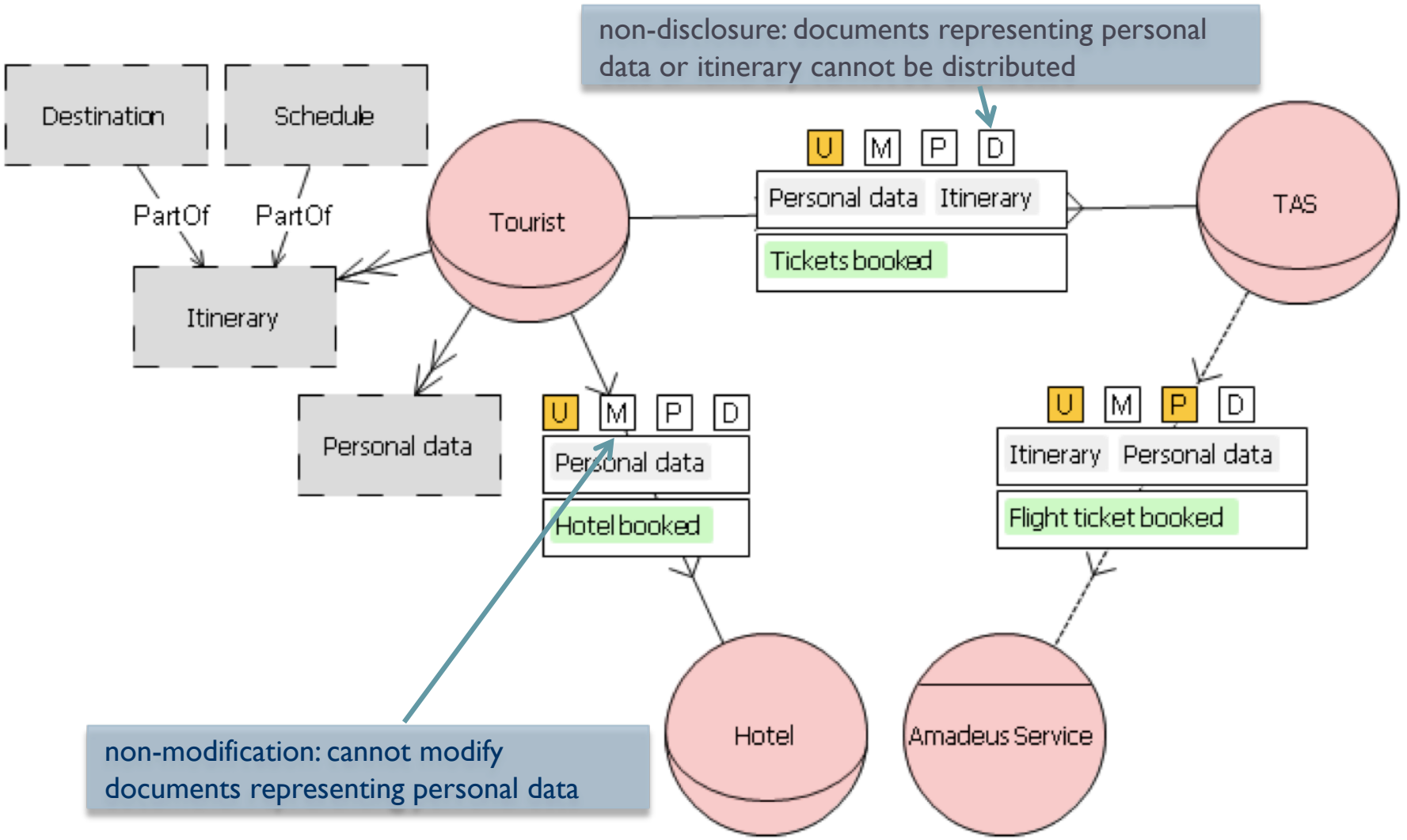
# Authorisation view: an example



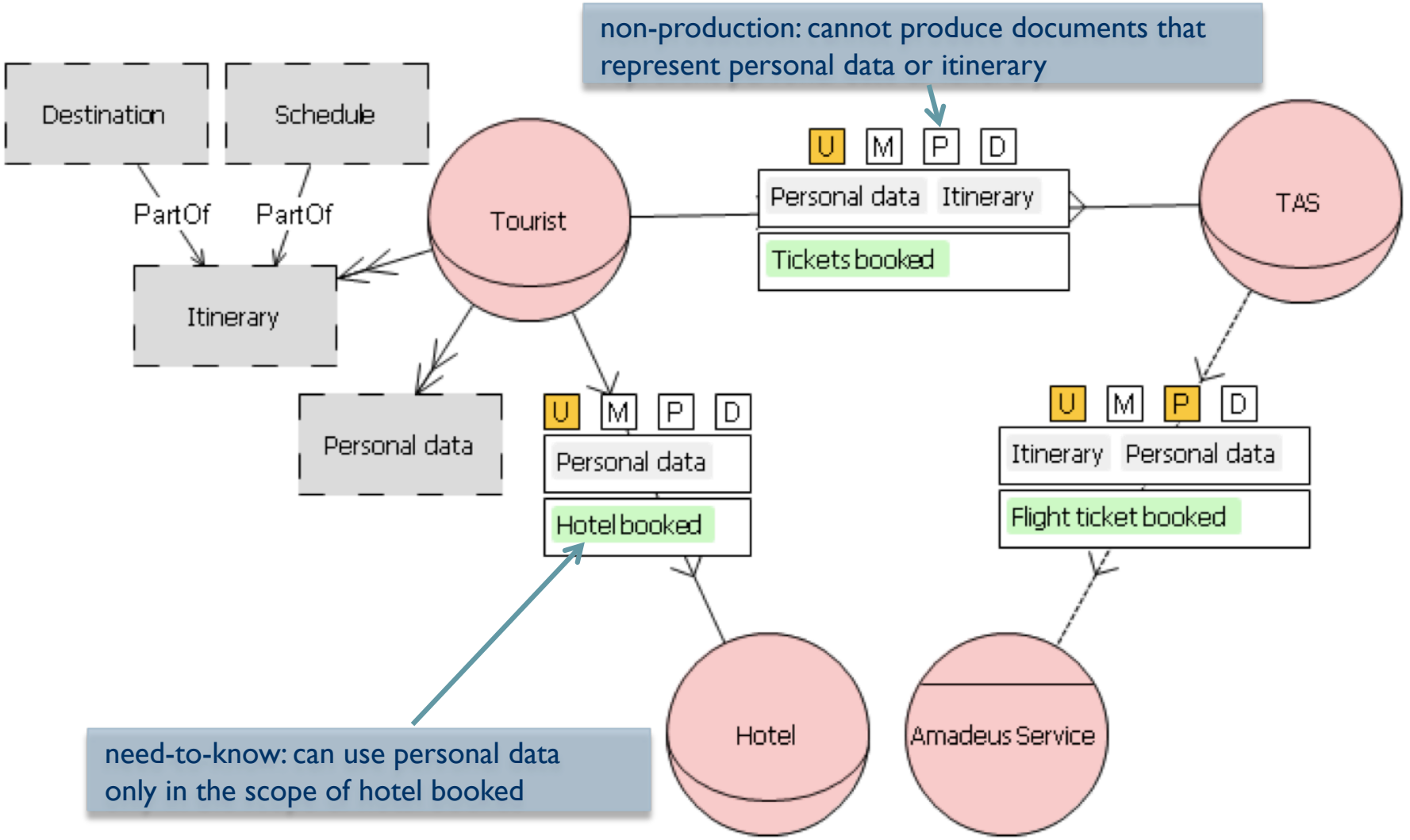
# Expressing security needs via authorisations

- Security needs about confidentiality are expressed by allowing only certain operations and limiting the scope
  - Need-to-know ← limiting the scope
  - Non-usage ← not allowing usage
  - Non-modification ← not allowing modification
  - Non-production ← not allowing production
  - Non-disclosure ← not allowing distribution

# Security needs via authorisations

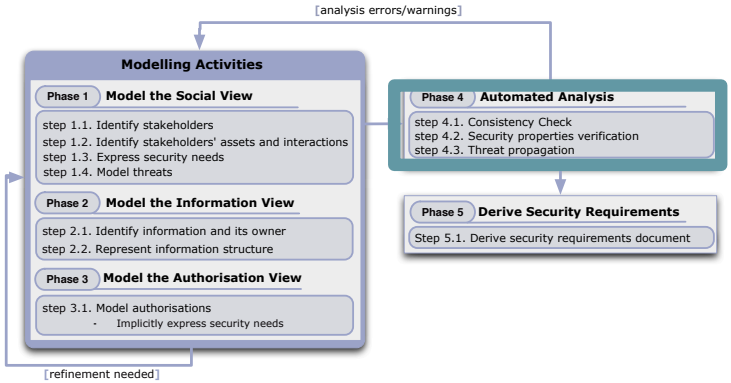


# Security needs via authorisations



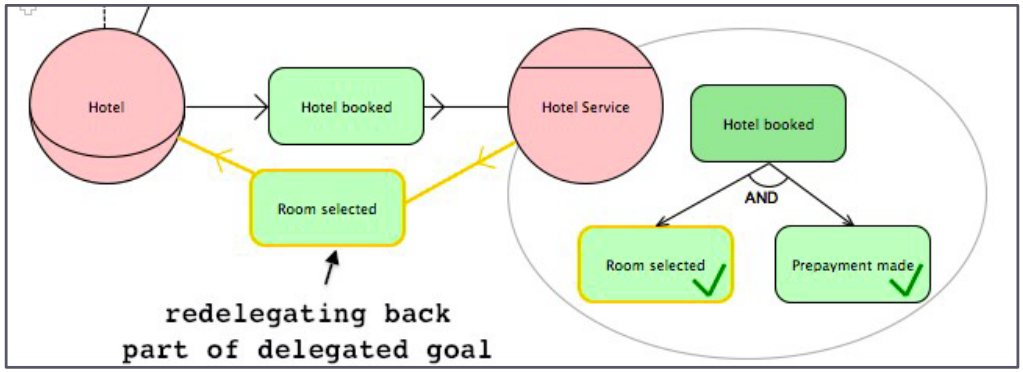
# Phase 4. Automated analysis

- Step 4.1 Consistency check
  - Does the model comply with the semantics of STS-ml?
  - E.g.: part-of cycles, contribution cycles
- Step 4.2 Security properties verification
  - Security requirements cannot be fulfilled in the modeled STS
  - E.g.: violation of no-delegation, non-usage, non-disclosure, separation of duty, …
- Step 4.3 Threat propagation
  - How does the specification of threats affect other assets



# Step 4.1. Consistency analysis

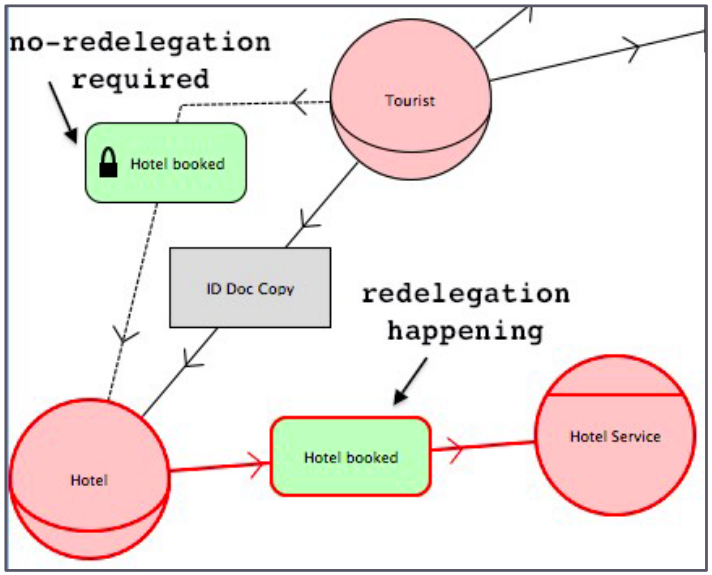
- Post-modelling checks
  - Give warnings or errors and visualize to designer
- Current checks
  - Single goal decompositions
  - Leaf goal delegation
  - Delegation cycles
  - Organisational constraints over goal trees
  - Part-of cycles
  - Contribution cycles
  - Ownership
    - Information without owner
  - Authorisations
    - Not empty, no duplicates




warning

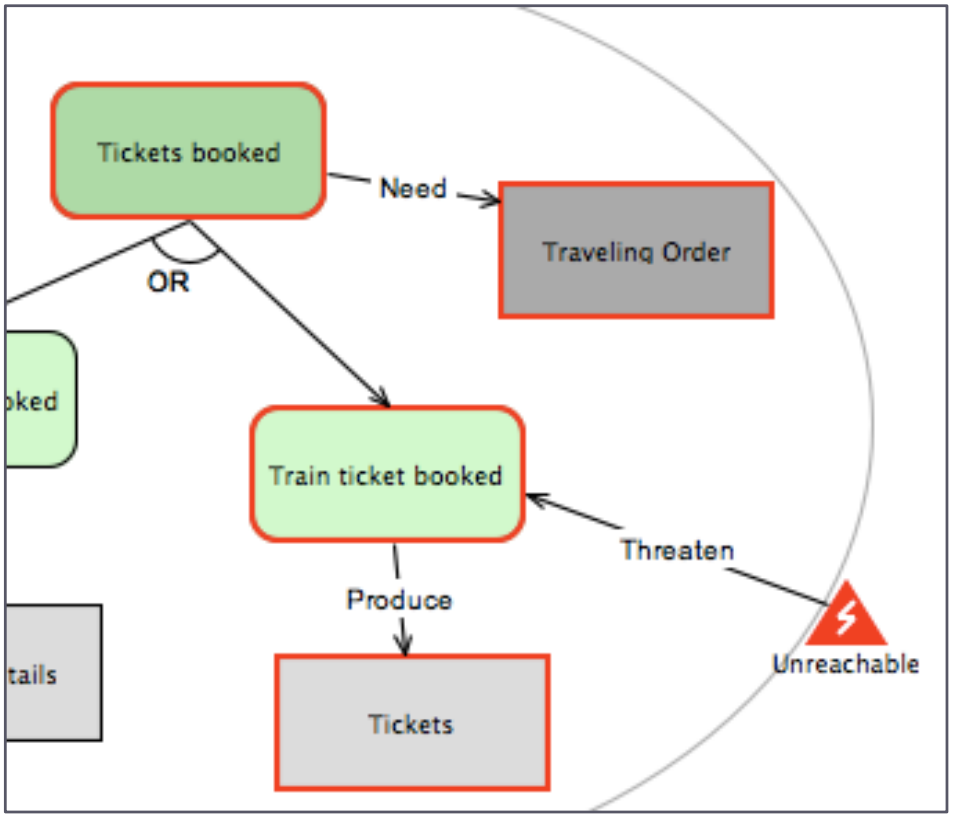
# Step 4.2. Security analysis

- It relies upon generating possible worlds
  - Identify and visualize possible problems
  - The engineer fixes the problem
  - Behind the scenes: formalization in disjunctive Datalog



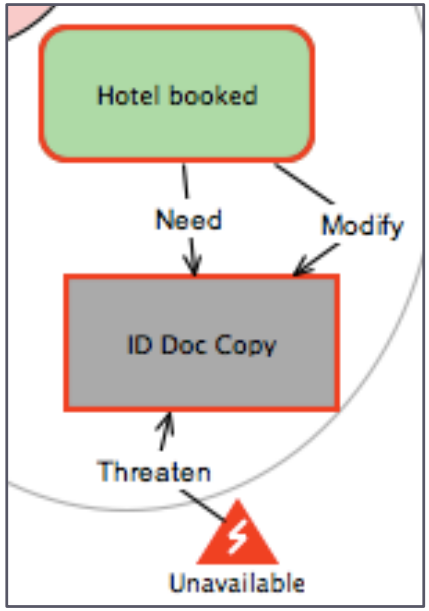

error

# Step 4.3. Threat propagation




**Threatened goal affects:**

- The document it produces
- The goal father tickets booked

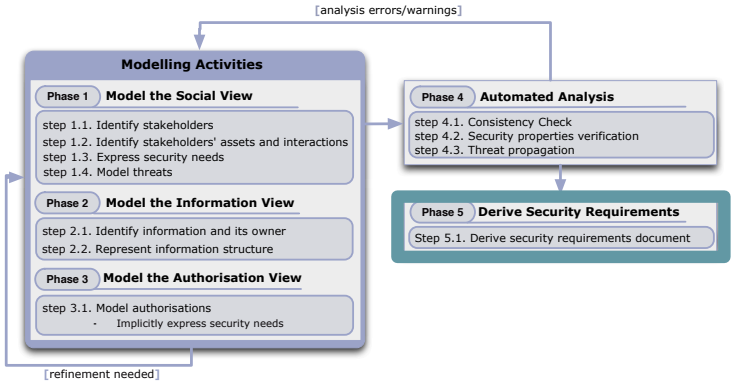



**Threatened document affects:**

- The goal that needs and modifies this document

# Phase 5. Derive security requirements

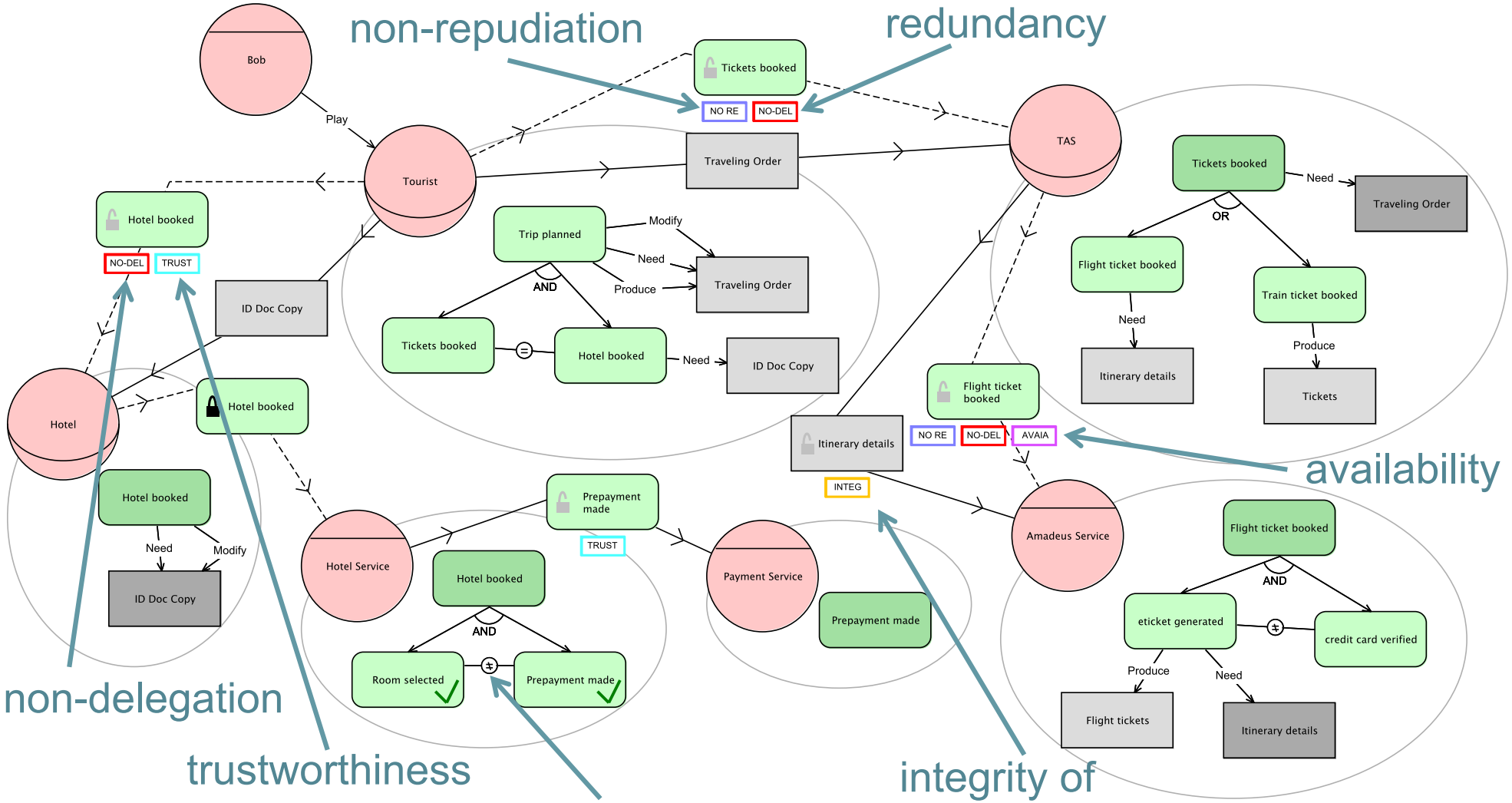
- Requirements models are useful for communication purposes with the stakeholders
- Requirements specifications tell designers what the system has to implement
  - In STS-ml, security requirements specifications are automatically derived from requirements models
  - Output: security requirements document



# Step 5.1. Derive security requirements

- In STS-ml
  - Security requirements constrain interactions in contractual terms
  - These contracts are expressed for each required security need
    - For each security need expressed from one actor to the other, a requirement is generated on the opposite direction to express compliance with the required security need
  - For each requirement
    - Requestor, Requirement, Responsible

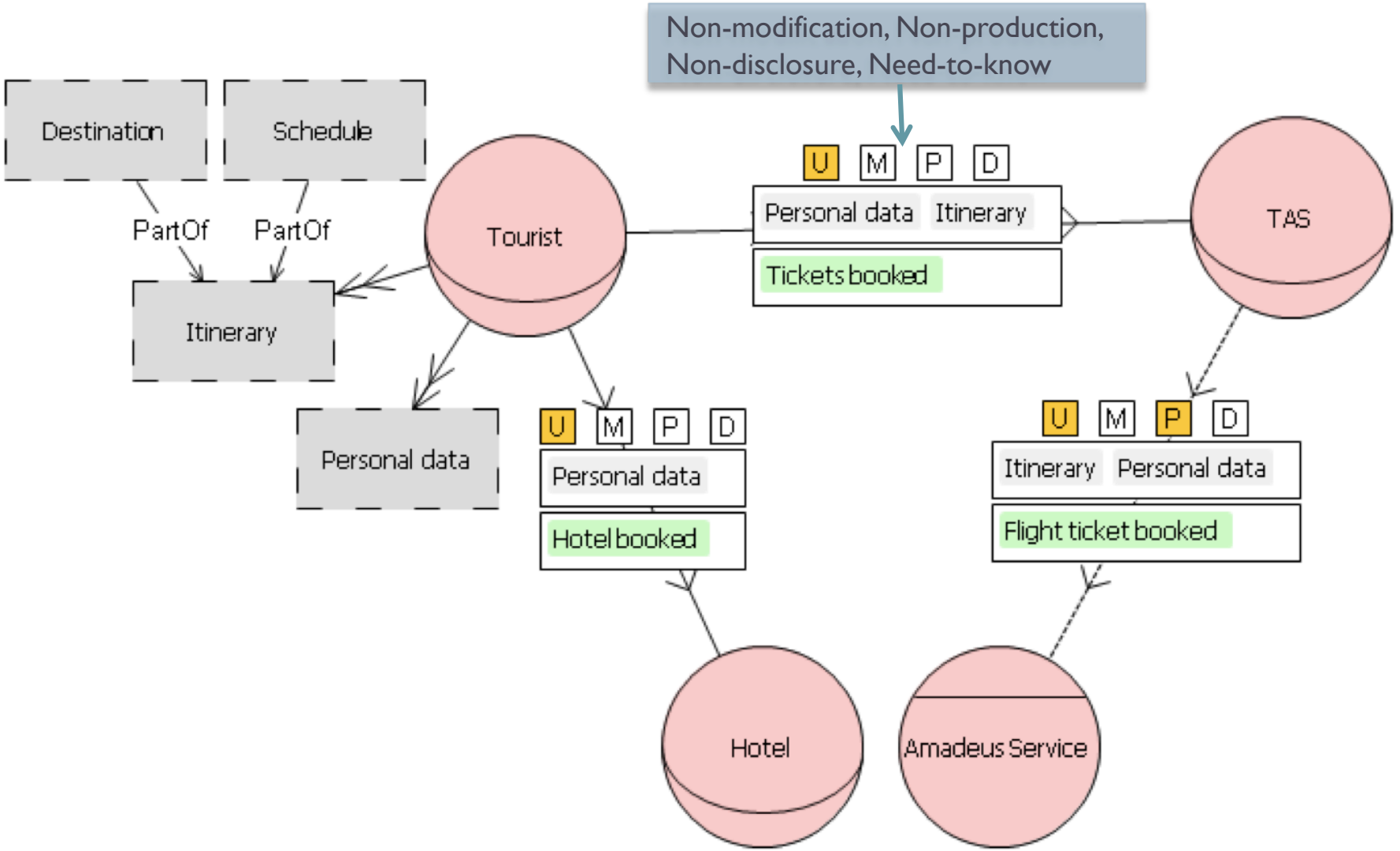
# Social view: expressing security needs



# Step 5.1. Derive security requirements

| Responsible | Security Requirement | Requester |
|---|---|---|
| TAS | non-repudiation-of-acceptance (delegated(Tourist,TAS,tickets booked)) | Tourist |
| Tourist | non-repudiation-of-delegation (delegated(Tourist,TAS,tickets booked)) | TAS |
| TAS | true-redundancy-multiple-actor(tickets booked) | Tourist |
| Hotel | no-delegation(hotel booked) | Tourist |
| Amadeus FS | integrity-of-transmission (provided(TAS,Amadeus Service,Itinerary details) | TAS |
| Any | not-achieve-both (eticket generated,credit card verified) | Org |
| Amadeus FS | availability(flight ticket booked, 85%) | TAS |
| Tourist | delegatedTo(Hotel, trustworthiness level 7) | - |

# Deriving security requirements: an example

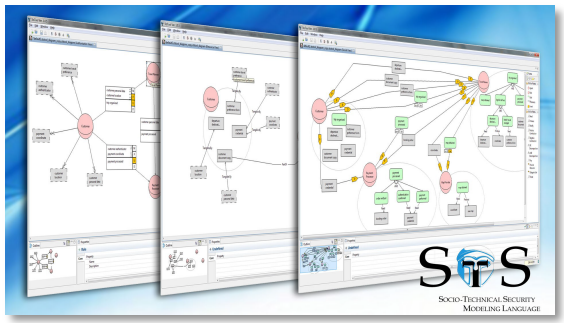


# Step 5.1. Derive security requirements

| Responsible | Security Requirement | Requester |
|---|---|---|
| TAS | need-to-know(personal data $\wedge$ itinerary, tickets booked) | Tourist |
| TAS | non-modification(personal data $\wedge$ itinerary | Tourist |
| TAS | non-production(personal data $\wedge$ itinerary) | Tourist |
| TAS | non-disclosure(personal data $\wedge$ itinerary) | Tourist |

# Tool Support: STS-Tool

- STS-Tool is the modelling and analysis support tool for STS-ml
  - Built on top of Eclipse
    - Standalone Eclipse RCP application
- Freely available for download: http://www.sts-tool.eu
- Derivation of security requirements
  - Report generation
- Multi-platform (Win, Linux, Mac)



# The End

paja@disi.unitn.it

- **Thank you!**


**12 December 2012**