UNIVERSITY OF TRENTO - Italy

DETECTING INCONSISTENCIES BETWEEN ORGANIZATIONAL REQUIREMENTS AND SECURITY REQUIREMENTS WITH STS-ML

Elda Paja, Fabiano Dalpiaz, Paolo Giorgini

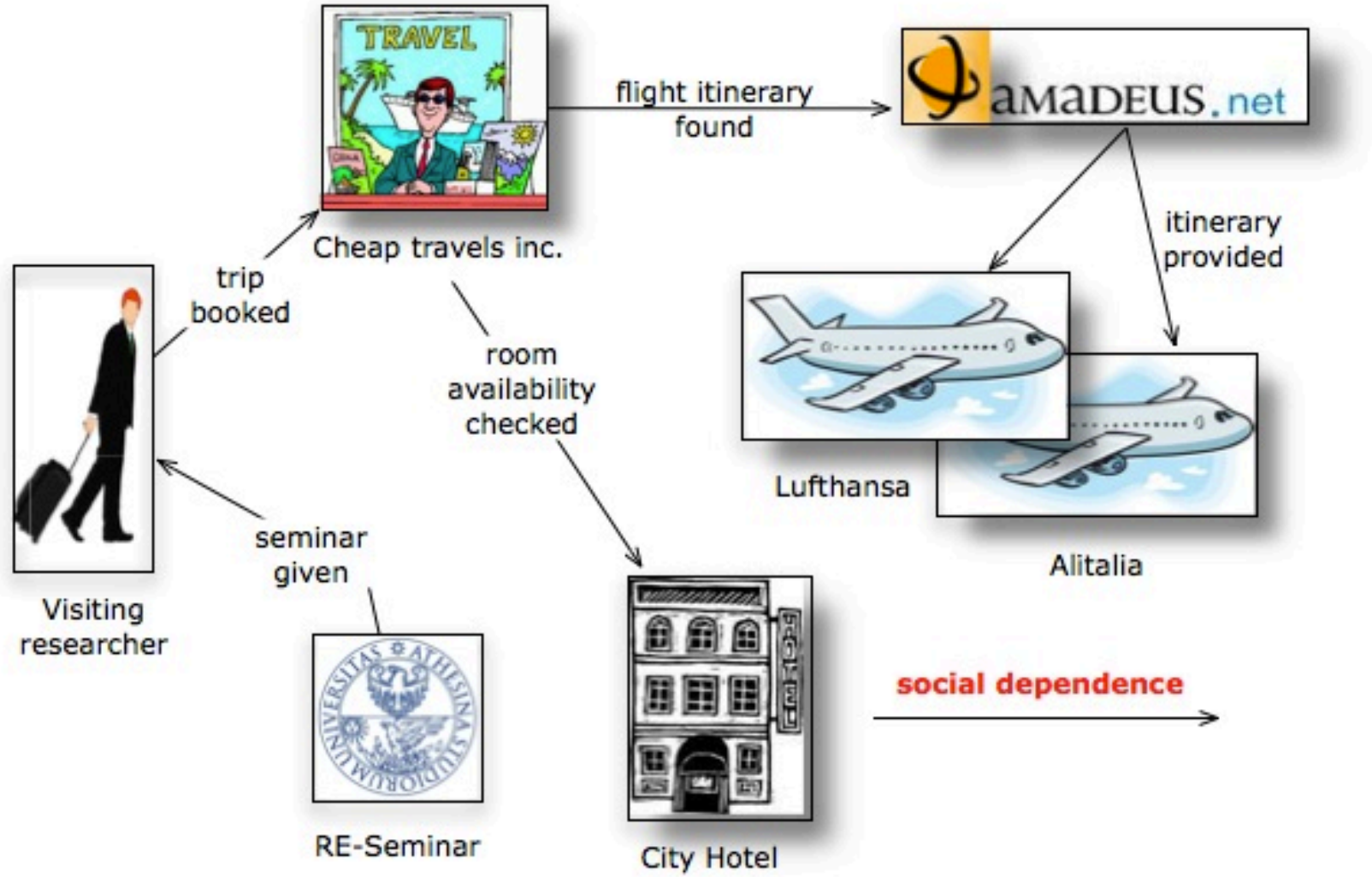
Re-Seminar

October 25th 2012

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



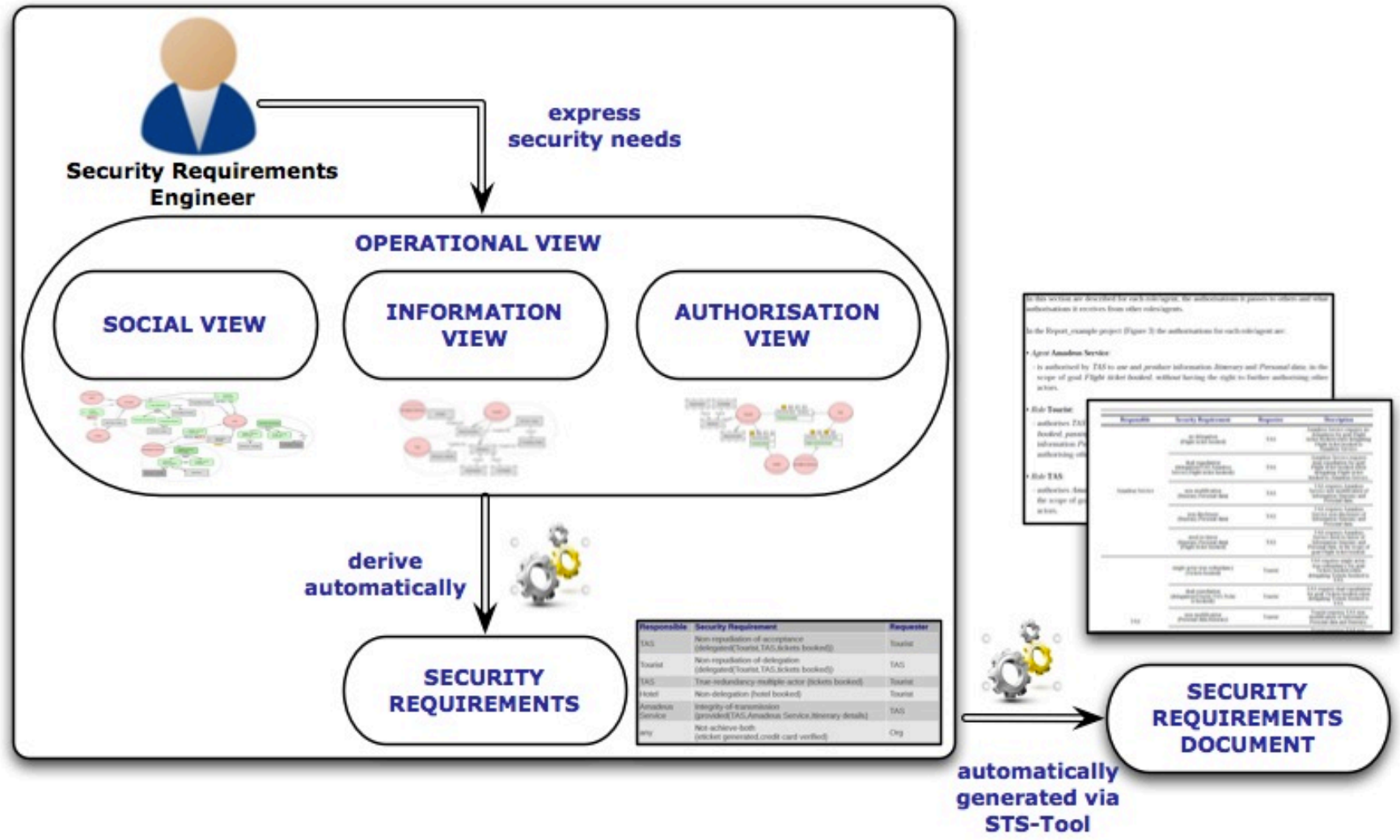
The Security Problem in STSs

- 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

Our Approach: STS-ml

- Role – and goal – oriented requirements modelling language
- Security requirements as social contracts that constrain interactions
- Allow actors to express constraints (security needs) over interactions
  - Social dependence
    - E.g.: visiting researcher depends on the cheap travel inc. to book the hotel and flight tickets
  - 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
- Models are built diagrammatically
  - Multiple views, each focusing on a specific perspective

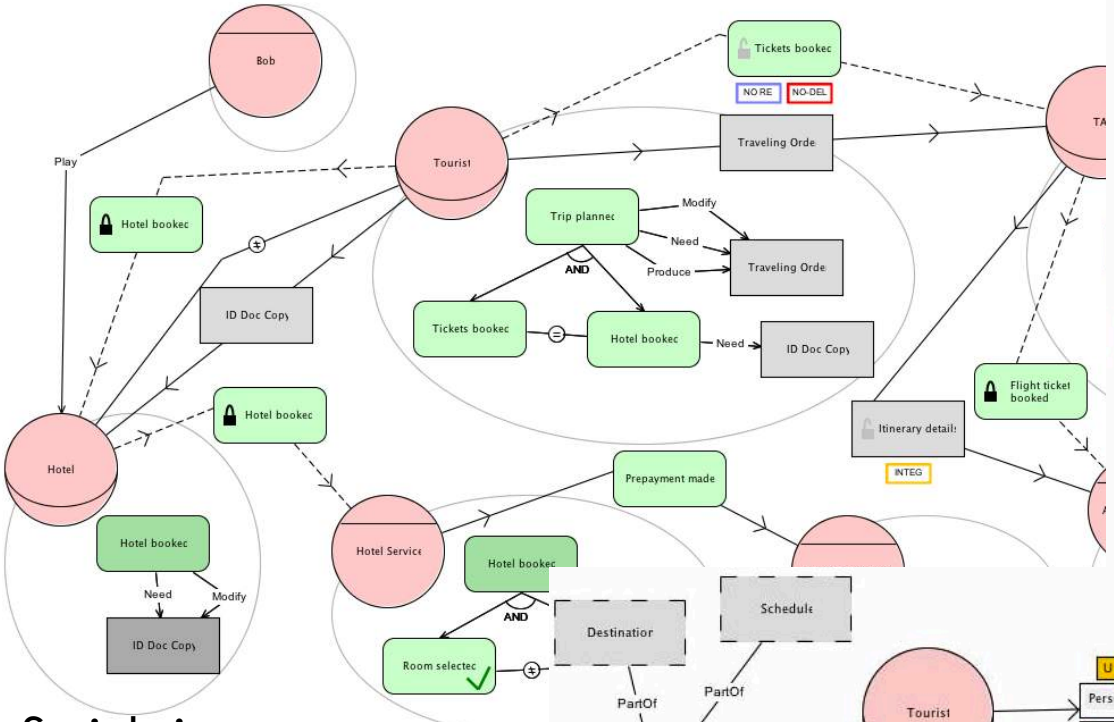
STS-ml: outline



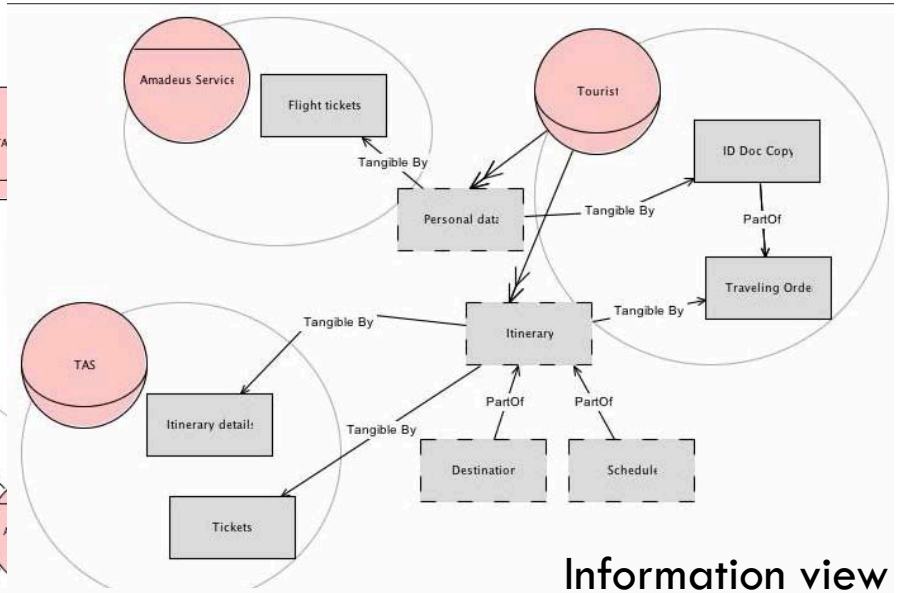
Formal Framework

- A framework to detect inconsistencies
  - Inconsistencies not trivial to find
  - Scalability is an issue
- Formal language to support automated reasoning about the expressed security needs
- Formally Defined
  - Security needs supported by STS-ml
  - The derived security requirements (in terms of social commitments)
    - Are the security needs violated in the modelled STS?
    - Key question: Is the specification consistent?
- Built on top of DLV
  - Define transformation rules from STS-ml concepts and relations into Datalog predicates
  - Define propagation rules

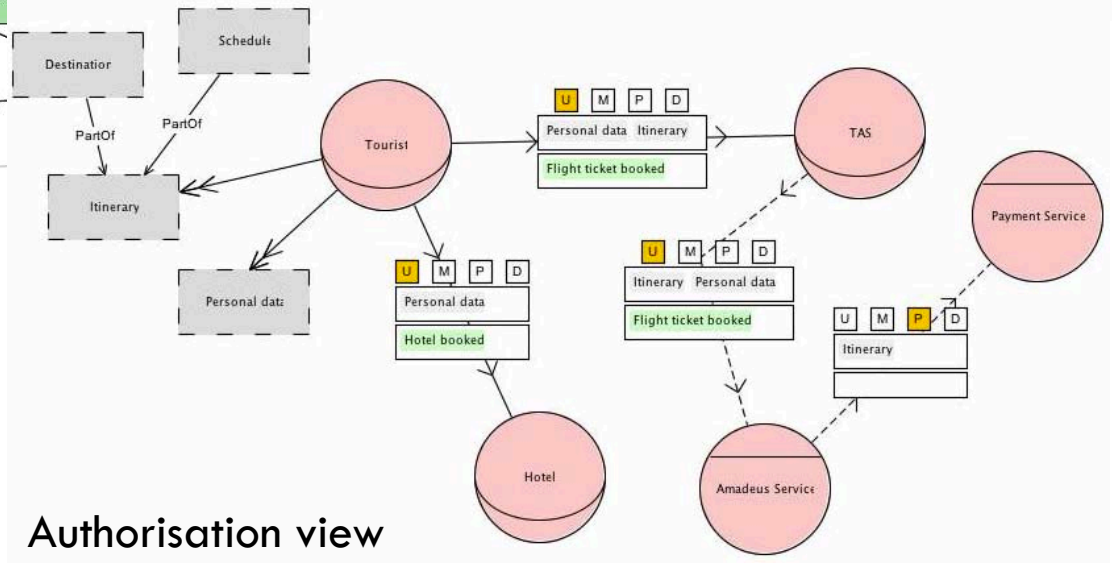
Modelling with STS-ml



Social view



Information view

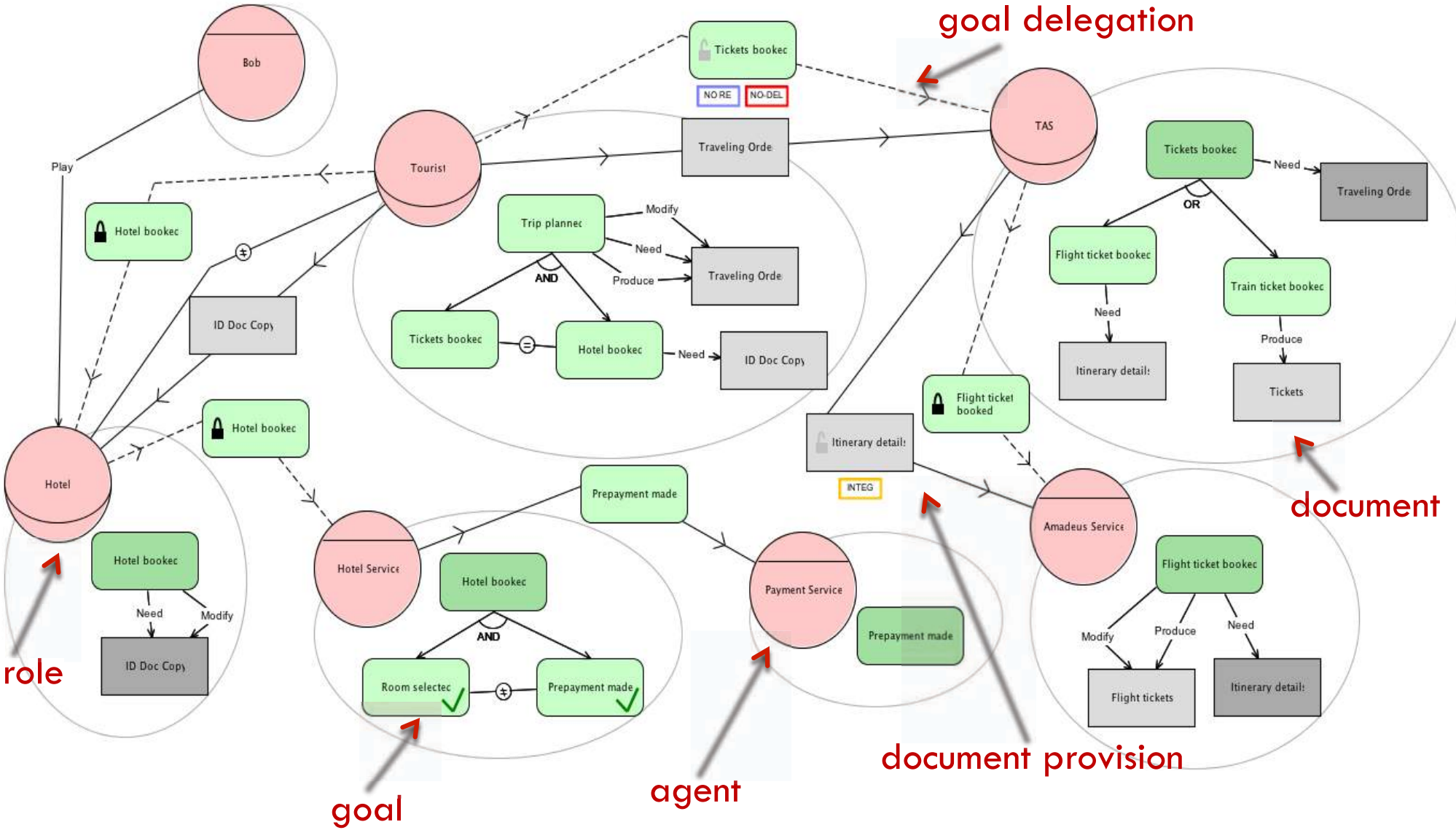


Authorisation view

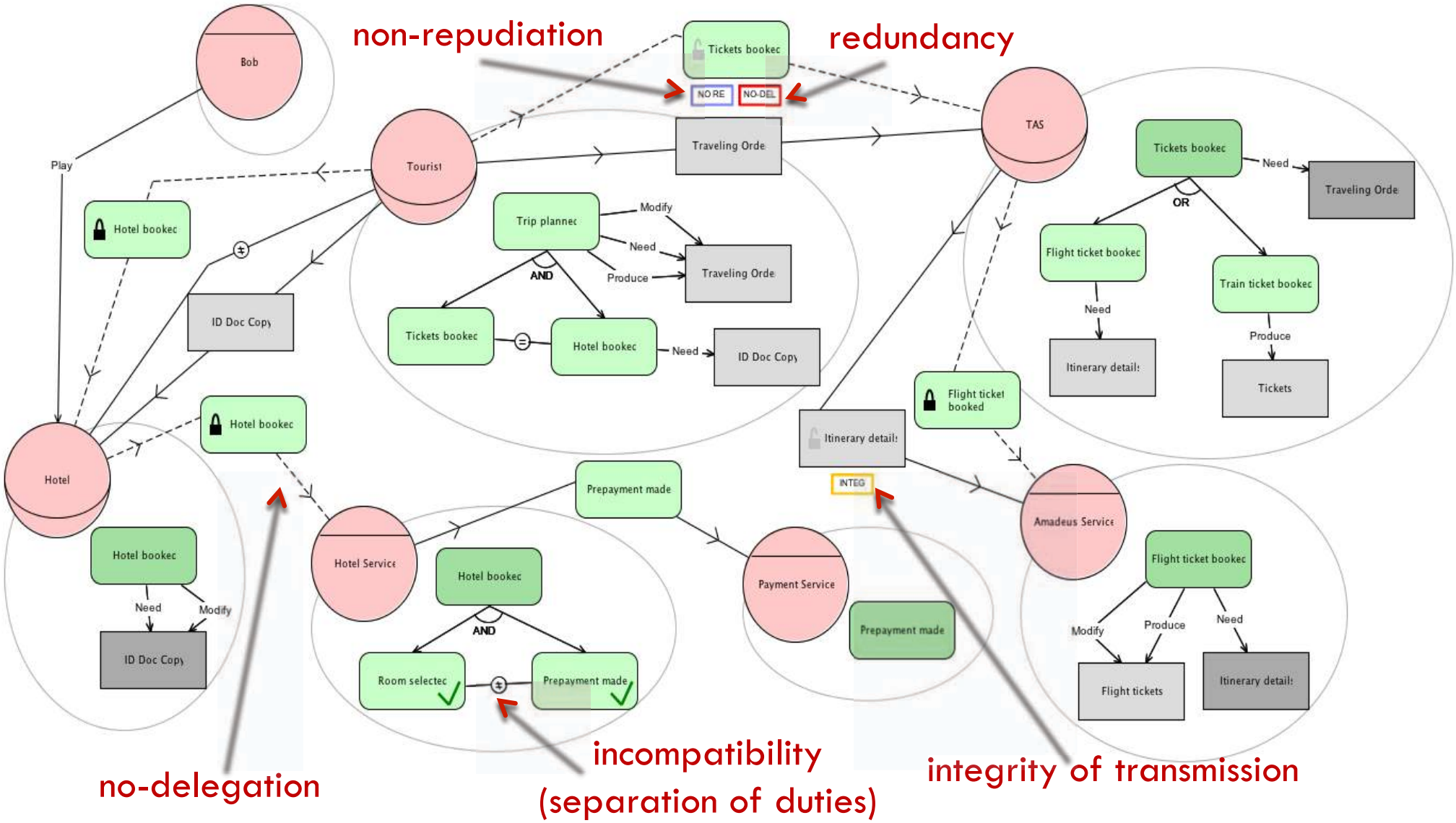
3 Views:

Different perspectives of the same model!

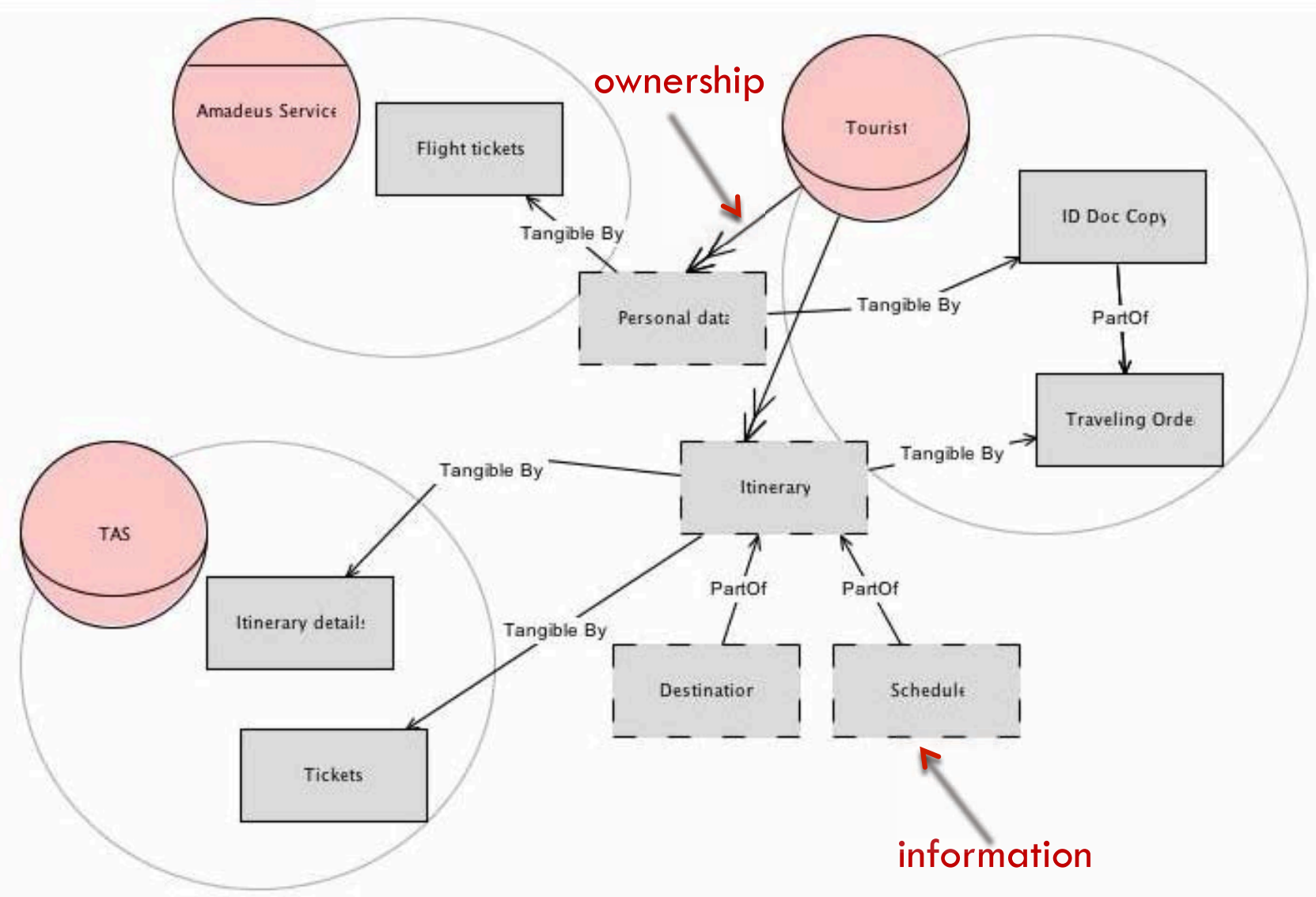
Social view: an example



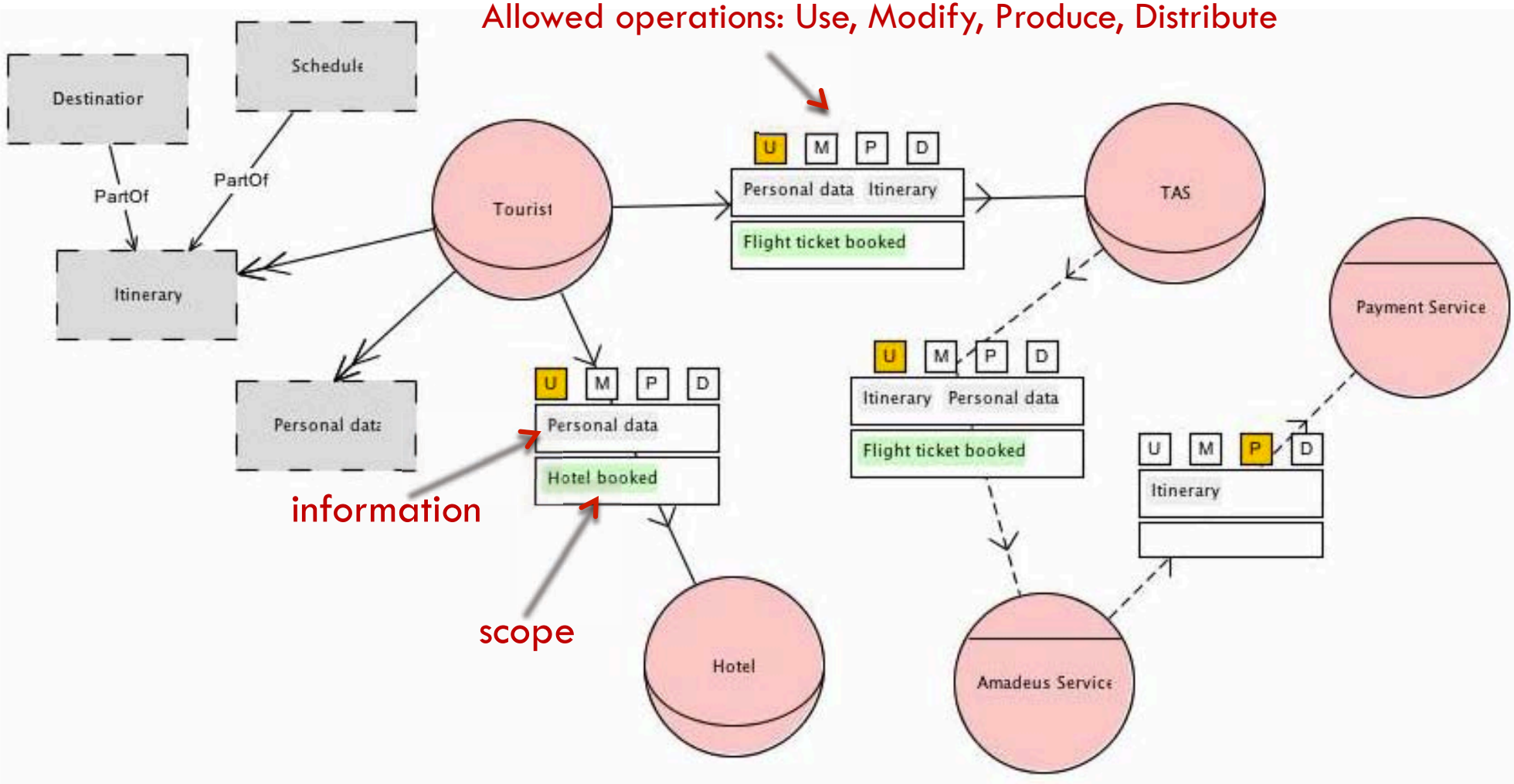
Social view: an example



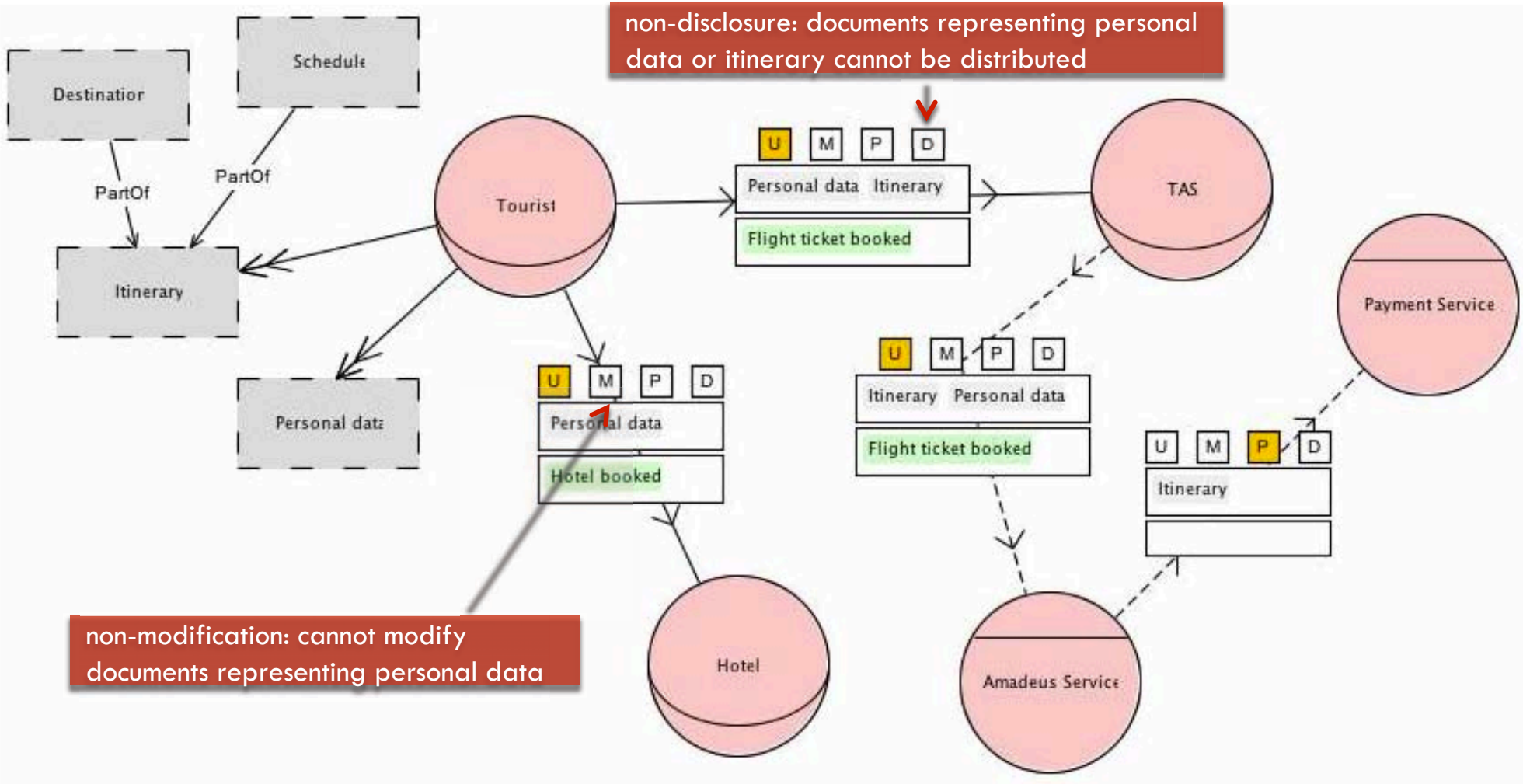
Information view: an example



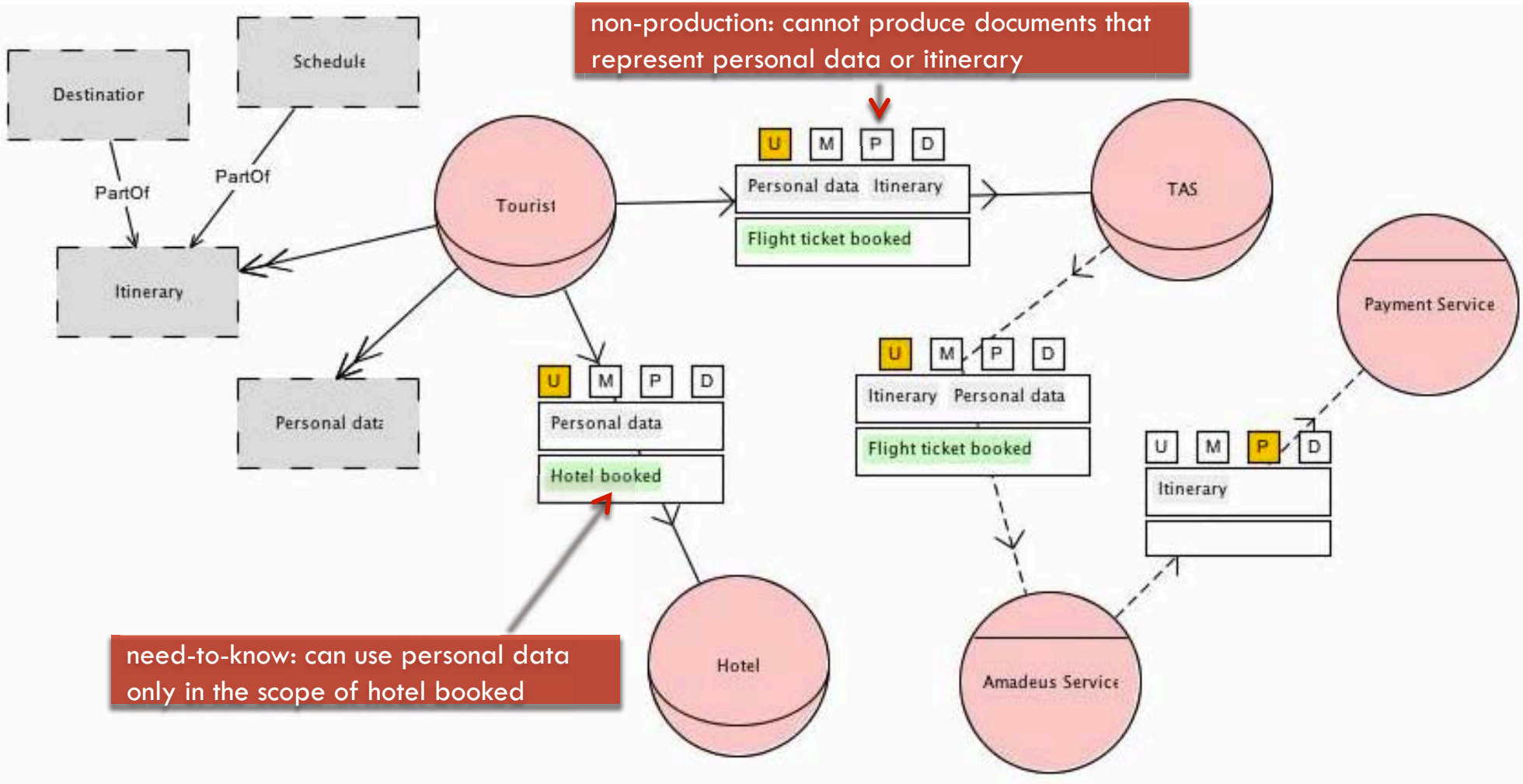
Authorisation view: an example



Authorisation view: expressing security needs



Authorisation view: expressing security needs



Supported security needs

- non-repudiation (3 types): non-repudiation of delegation, of acceptance, of delegation and acceptance;
  no-delegation;
  redundancy (4 types): fallback redundancy single, fallback redundancy multi, true redundancy single, true redundancy multi;
  integrity of transmission

- non-usage, non-modification, non-production, non-disclosure, need-to-know

- separation of duties, binding of duties

Requirements specification via commitments

- In STS-ml
  - Security requirements constrain interactions in contractual terms
  - These contracts are expressed as social commitments

- Social commitment: a promise with contractual validity
  - made by a debtor actor to a creditor actor
  - that a state of affairs will be brought about [consequent]
  - (optional) provided that another state of affairs holds [antecedent]

- E.g.: *C(Elda, RE-seminar-group, seminar scheduled, talk given)*

Commitments as requirements

- Commitments can express requirements
  - Social commitments represent the constraints the actors shall comply with while interacting
    - For each security need expressed from one actor to the other, a commitment is expected on the opposite direction to comply with the security need

- Security requirements via commitments
  - Debtor actor = Responsible
  - Creditor actor = Requester
  - Antecedent = Precondition
  - Consequent = Security requirement

Derived 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
TAS	need-to-know(personal data, trip planned, u)	Tourist
Hotel	non-disclosure(personal data)	Tourist
Amadeus FS	non-modification(personal data ☐ itinerary)	TAS
TAS	non-production(personal data ☐ itinerary)	Tourist
Any	not-achieve-both(room selected, prepayment made)	Org

Automated Analysis

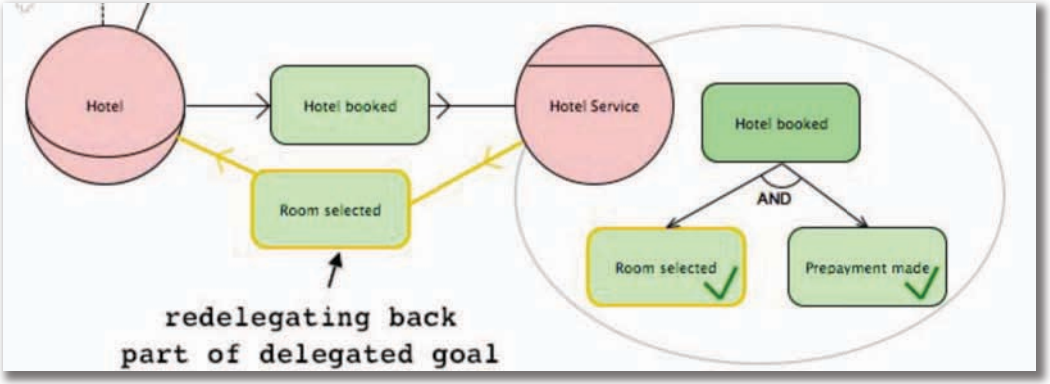
- Consistency Analysis
  - Does the model comply with the semantics of STS-ml?
  - E.g.: part-of cycles, contribution cycles

- Security Analysis
  - Do actors comply with the specified security needs?
    - Identify violations of security needs
  - E.g.: violation of no-delegation, non-usage, non-disclosure

Consistency Analysis

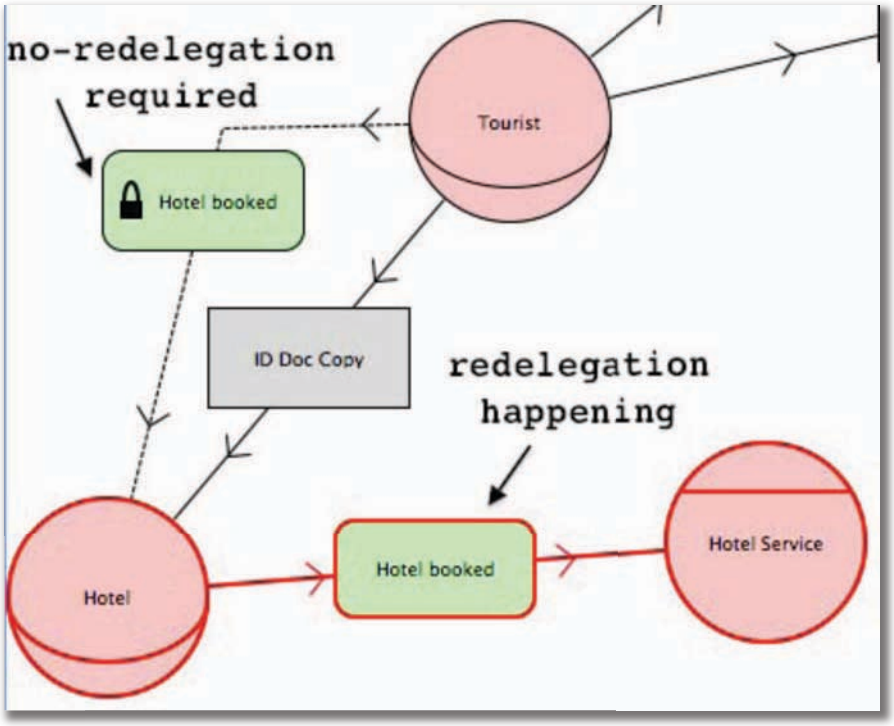
- Post-modelling checks
  - Give warnings or errors and visualize to designer
- Current checks
  - Single goal decompositions
  - Leaf goal delegation
  - Delegation cycles
  - Organisational constraints over goal trees
  - Part-of cycles
  - Contribution cycles
  - Ownership
    - Information without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

warning



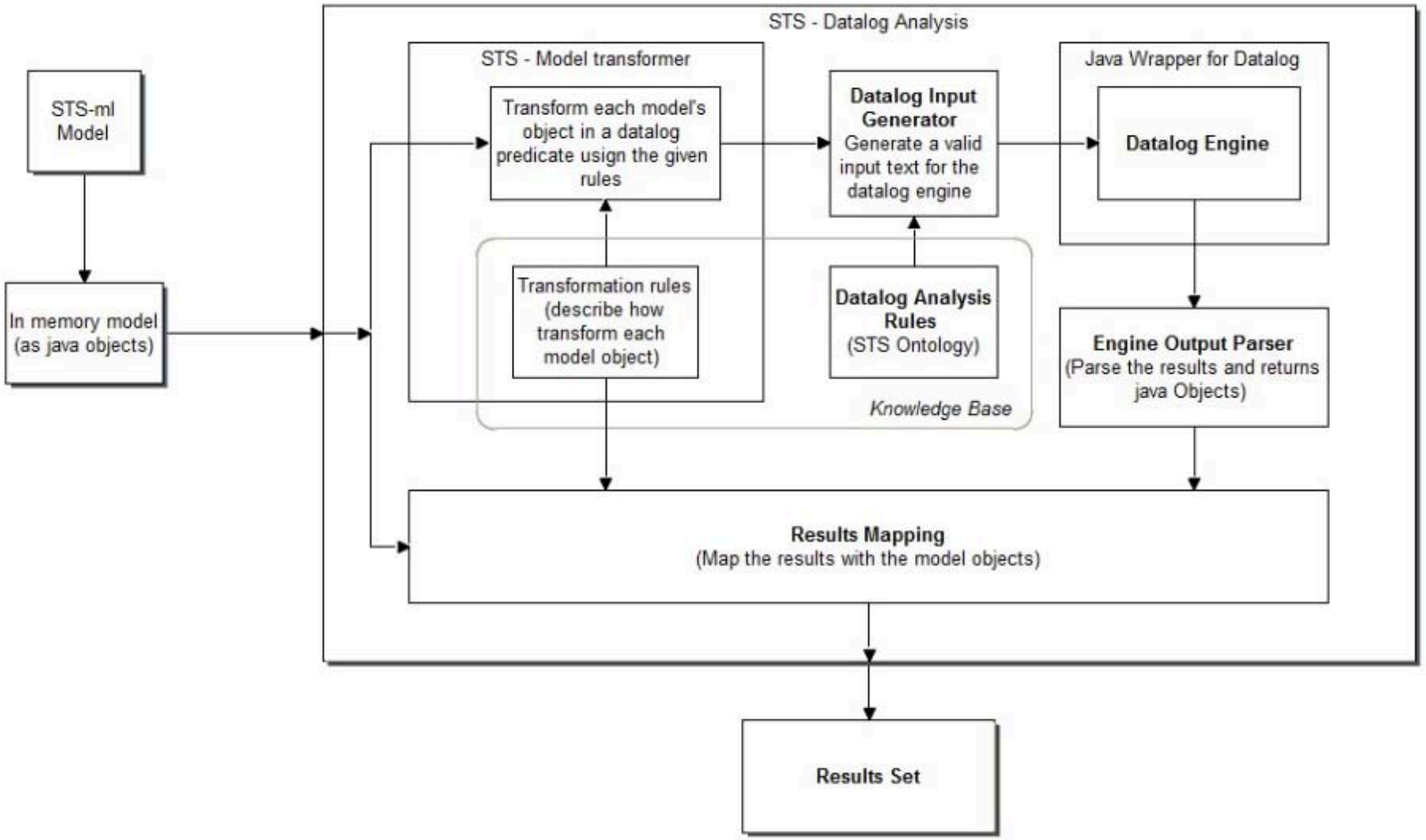
Security Analysis

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



error

Security Analysis

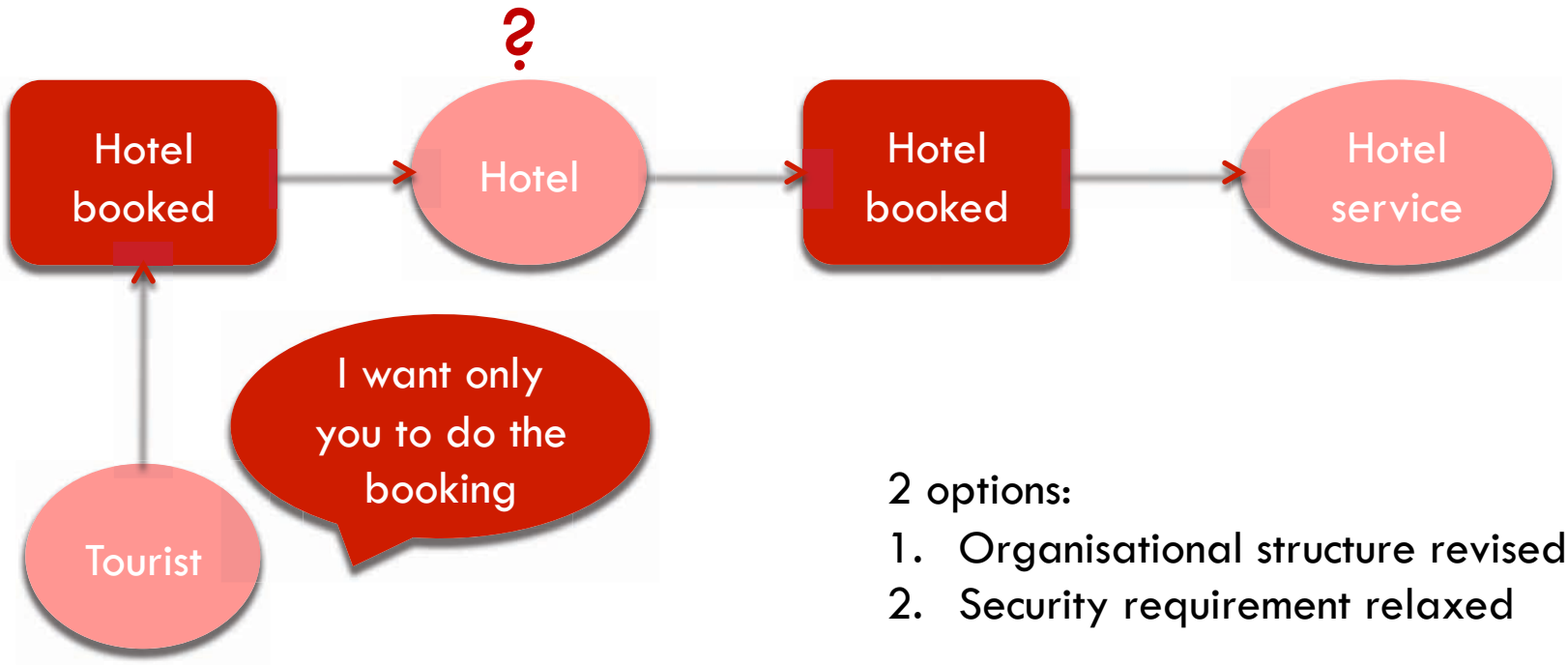


Identifying Organisational – Security Inconsistencies

- STS-ml supports performing a set of actions
  - Delegate
  - Use
  - Modify
  - Produce
  - Distribute
  - Provide
  - Authorise
- Security needs define what actions must not be performed
  - No-delegation
  - Non-usage
  - Non-modification
  - Non-production
  - Non-distribution
  - Non-transferrable authority

Identifying Inconsistencies: an example

- Organizational requirements – Security requirements Inconsistencies
  - Security requirements cannot be satisfied in the modeled organizational structure



2 options:

1. Organisational structure revised
2. Security requirement relaxed

Identifying inconsistencies: an example

- No-delegation

%define violation property, goal might be decomposed

```
violate_no_delegation(R2,R1,G,Gi) :- delegated(R1,R2,G), no_delegation(R1,R2,G,Gi), delegated(R2,_,Gi).
```

% expand no-delegation to the subgoals

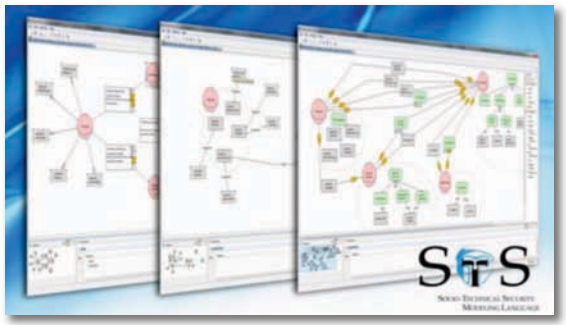
```
no_delegation(R1,R2,Gp,G) :- no_delegation(R1,R2,_,Gp), has(R2,G), isRefined(R2,Gp,G).
```

- Results

```
violate_no_delegation(Hotel,Tourist,hotel booked,hotel booked)
```

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)



Ongoing and Future Work

- Implement Analysis for detecting inconsistent security requirements

- Evaluation
  - 2 different case studies
    - Air Traffic Control Management
    - eGoverment

Special Thanks

STS Team

- Dr. Fabiano Dalpiaz
- Mauro Poggianella (Developer)
- Dr. Pierluigi Roberti
- Prof. Paolo Giorgini

ANIKETOS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos)

The end

Thank you!

Questions?

Contact: paja@disi.unitn.it