# *Security analysis in AOSE*

Paolo Bresciani

Automated Reasoning Systems division
ITC-irst
(institute for scientific and technological research)

Trento (Italy)

# Security analysis: an early requirements need

- Security requirements are usually defined in terms of the existence of properties such as:
  - Confidentiality, Authentication, Integrity, Access Control, Non-repudiation, Availability
- They are obtained by analysing the attitude of the organization towards security and after studying the *security policy* of the organisation

→ The security issues should be addressed since the analysis of the organizational setting (*early requirements*)

# Security in Software Engineering

- Currently, in SE the definition of security requirements is usually considered after the design of the system (Devanbu&Stubblebine-2000)
- This means that security enforcement mechanisms have to be fitted into a pre-existing design
- SE considers security requirements, as *Non-Functional Requirements*
- Security Requirements represent
  - Quality characteristics
  - Constraints under which the system must operate

# Security Concepts

- Security Features
  - Security Features the system-to-be must have (e.g. Privacy)
- Protection Objectives
  - Principles that contribute towards the security features (e.g. Access Control)
- Security Mechanisms
  - Mechanisms to achieve the protection objectives (e.g. Authentication)

# Dealing with Security in AOSE

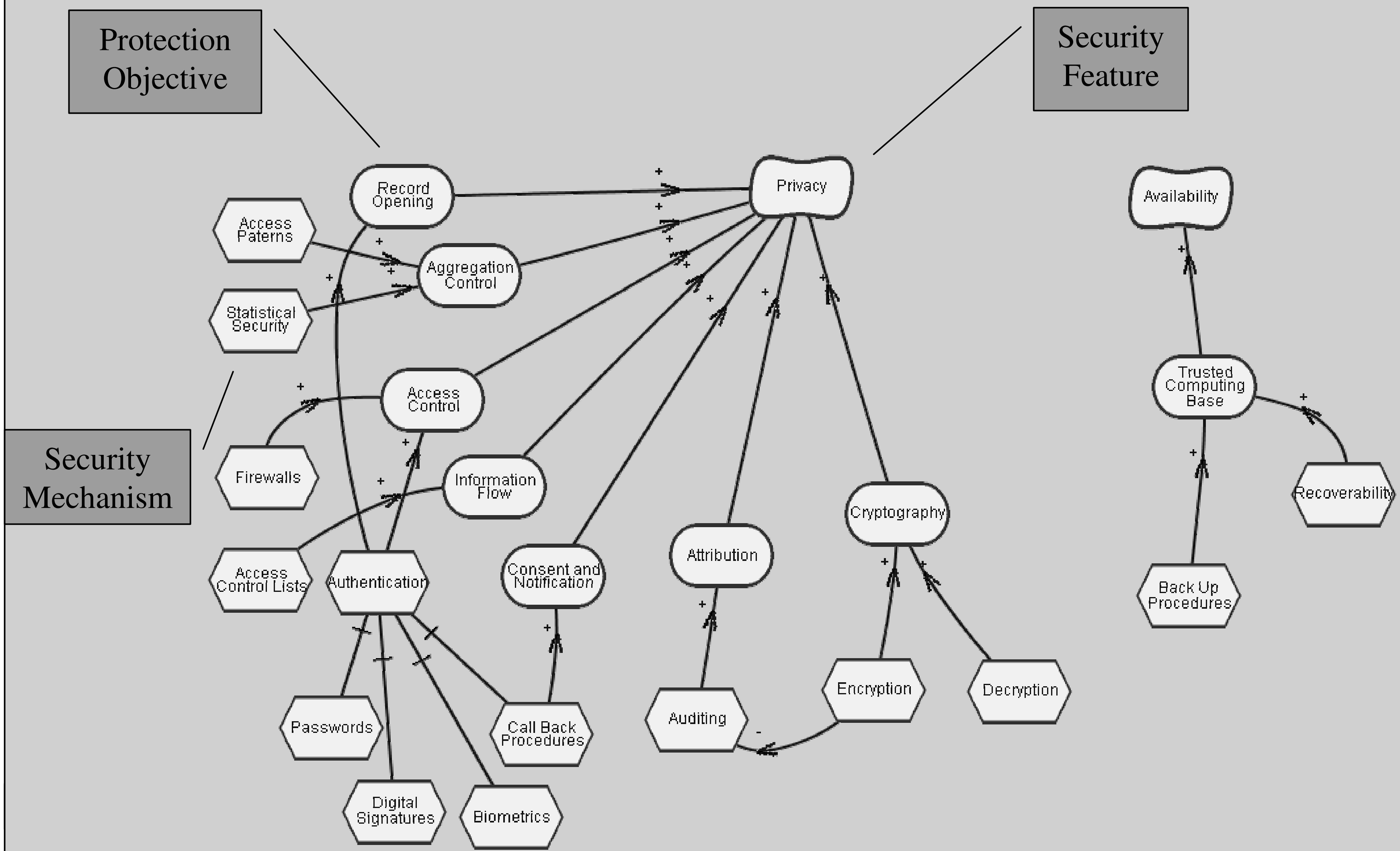
An example:

TROPOS

# Mapping Security Concepts into Tropos Concepts

- Security Features ➔ *secure Softgoals*
  - Security Features the system-to-be must have (e.g. Privacy)
- Protection Objectives ➔ *secure Hardgoals*
  - Principles that contribute towards the security features (e.g. Access Control)
- Security Mechanisms ➔ *secure Tasks*
  - Mechanisms to achieve the protection objectives (e.g. Authentication)

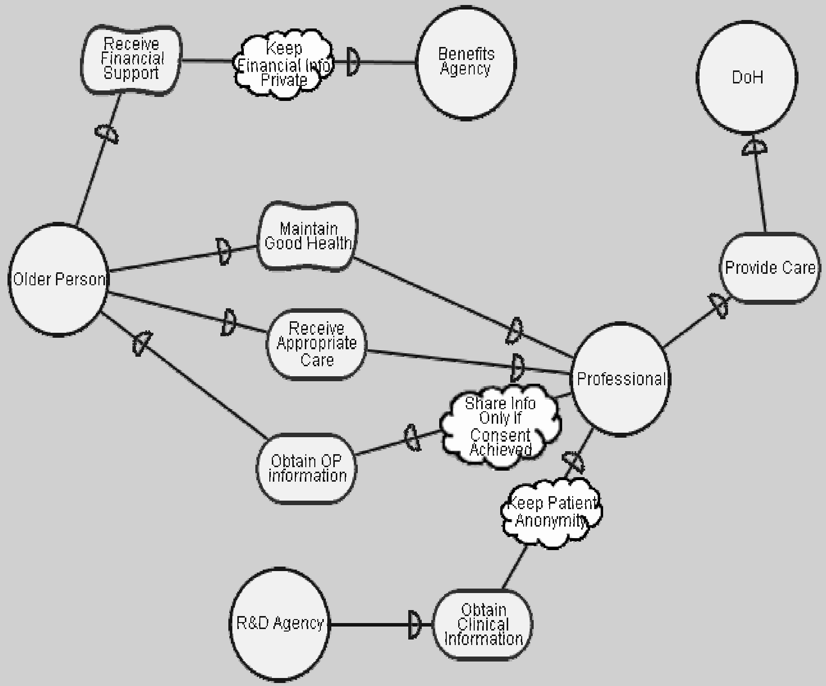
# Security Diagram Example



# Intentional Dependencies in Tropos

***Intentional Dependencies*** are an essential ingredient in Tropos.

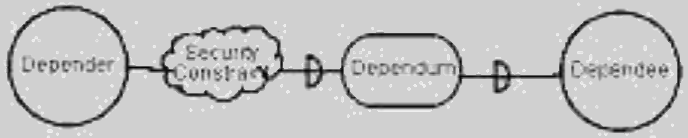
Security issues can be added to intentional dependecies as ***constranints***



# Constraints & Secure Dependencies

***Constraints*** are imposed by the actors for the dependency to be valid.

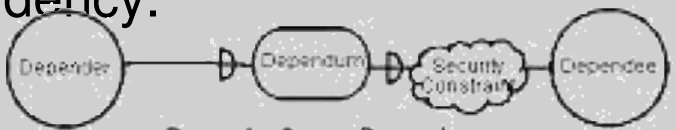
- Depender → Secure Dependency




Dependee Secure Dependency

  - depender depends on dependee, and depender introduces security constraints for the dependency.

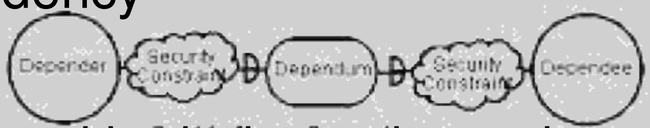
- Dependee → Secure Dependency




Depender Secure Dependency

  - depender depends on dependee, and dependee introduces security constraints for the dependency

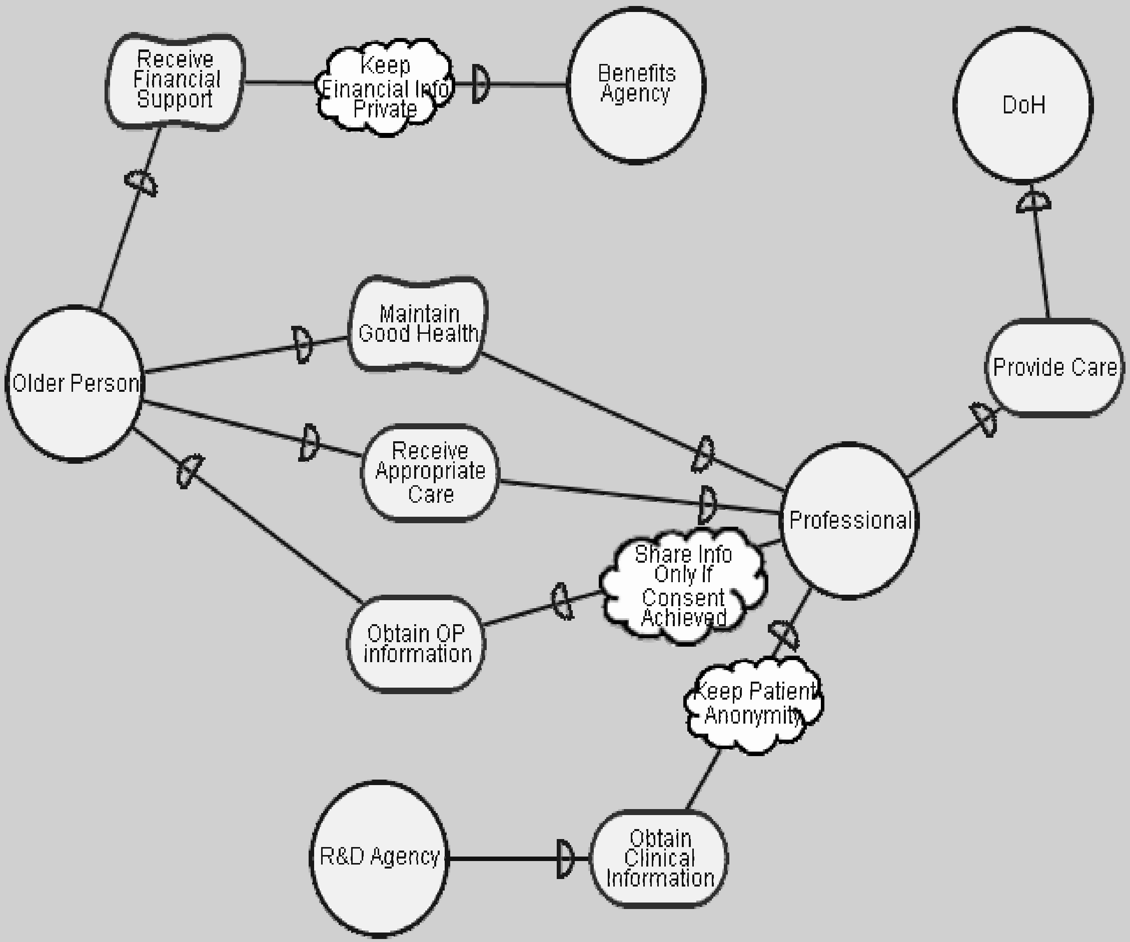
- Double Secure Dependency



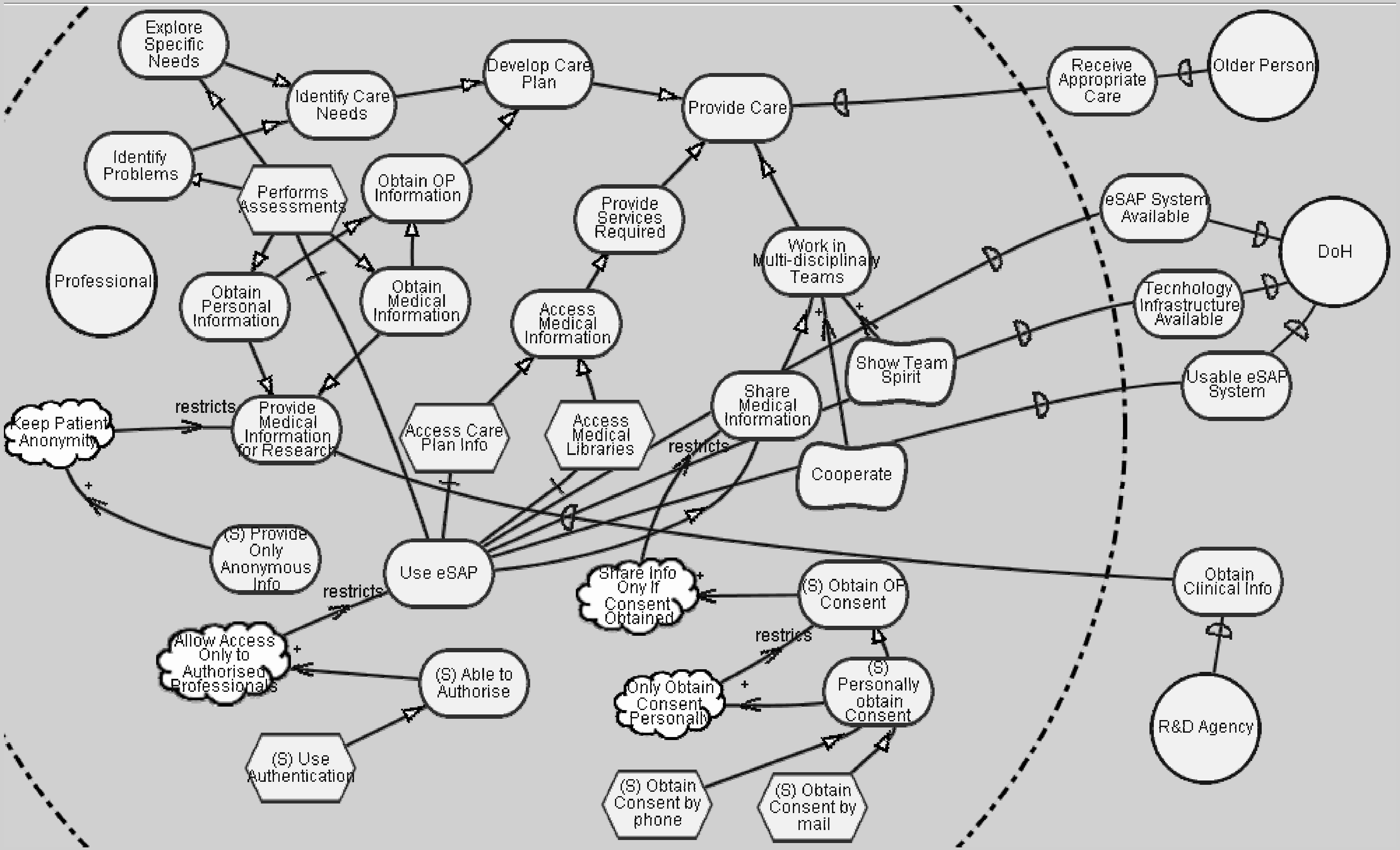

Double Secure Dependency

  - depender depends on dependee and both depender and dependee introduce security constraints for the dependency

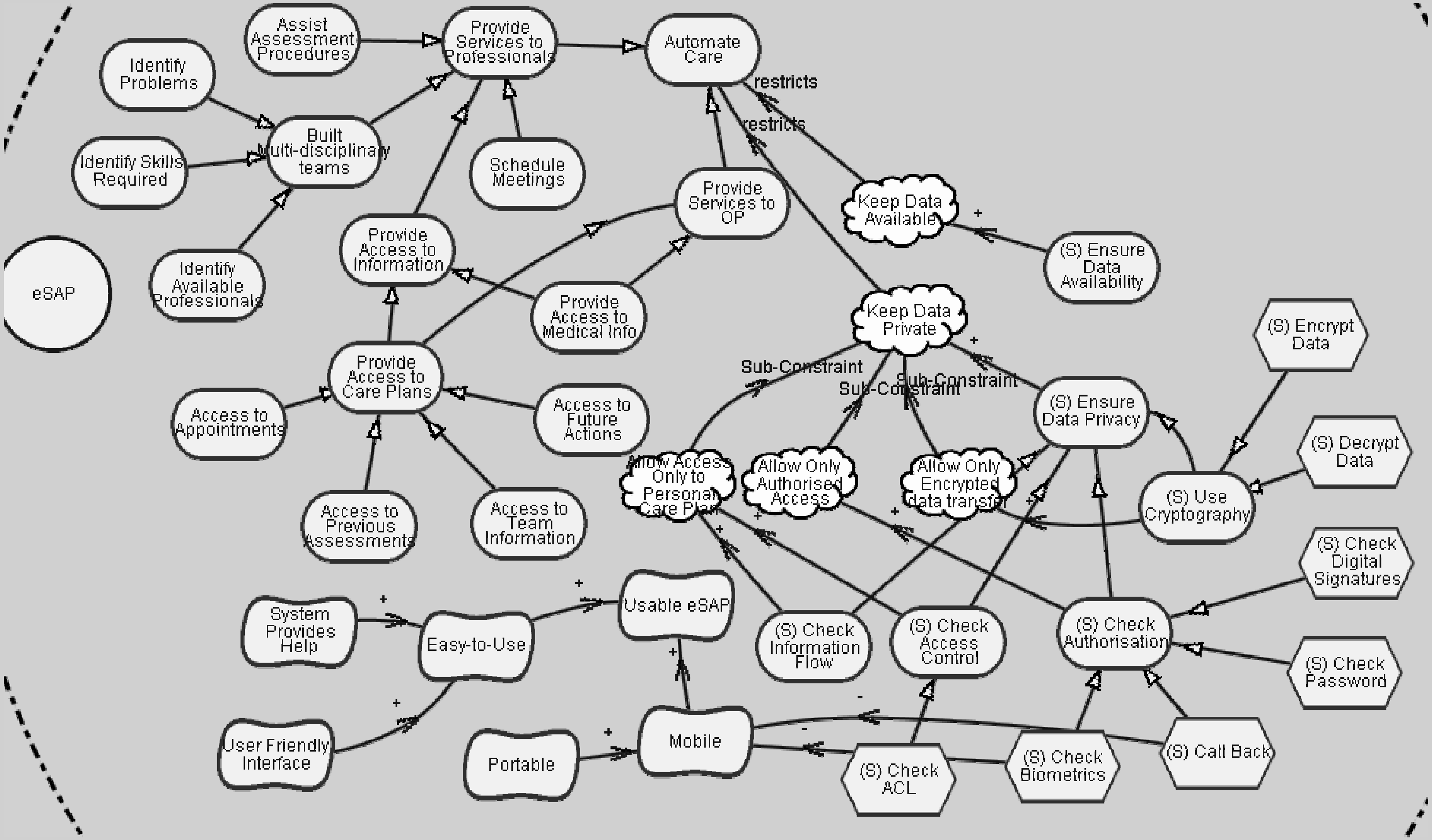
# Constraints Example



# Professional's Goals Analysis



# eSAP System Goals Analysis



# Conclusions

- Security should be be dealt with already during RE phases
- It’s not easy to fit security notions in RE in a simple way: this may be one more reason because security features are considered only afterwards
- Similar considerations may be applied to other instances of QoS analysis
- How to deal with them in general? (and in a simple way)