Course Requirements

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

Software Engineering for Dependable Systems

Tomas Bures

bures@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Organisation

- Seminar – every other week
  - 4 teaching hours each
- Tomáš Bureš
  - bures@d3s.mff.cuni.cz
- Course website
  - https://d3s.mff.cuni.cz/teaching/nswi054/

Requirements for passing the course

- Students have to subscribe to the course and the group in the Student Information System

- If you cannot attend, let me know (by email) in advance

Department of Distributed and Dependable Systems

Requirements for passing the course

- 80% attendance
- Completed homework/report
  - To be prepared by the next meeting
- These will cover the topics covered during the course:
  - Requirement modeling and specification
  - Test specification
  - Various design models

Department of Distributed and Dependable Systems

Resources

- Ian Sommerville: Software Engineering (10th edition)
  - http://iansommerville.com/software-engineering-book/

- Online resources indicated in each lecture

Introduction

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

Software Engineering for Dependable Systems

Tomas Bures

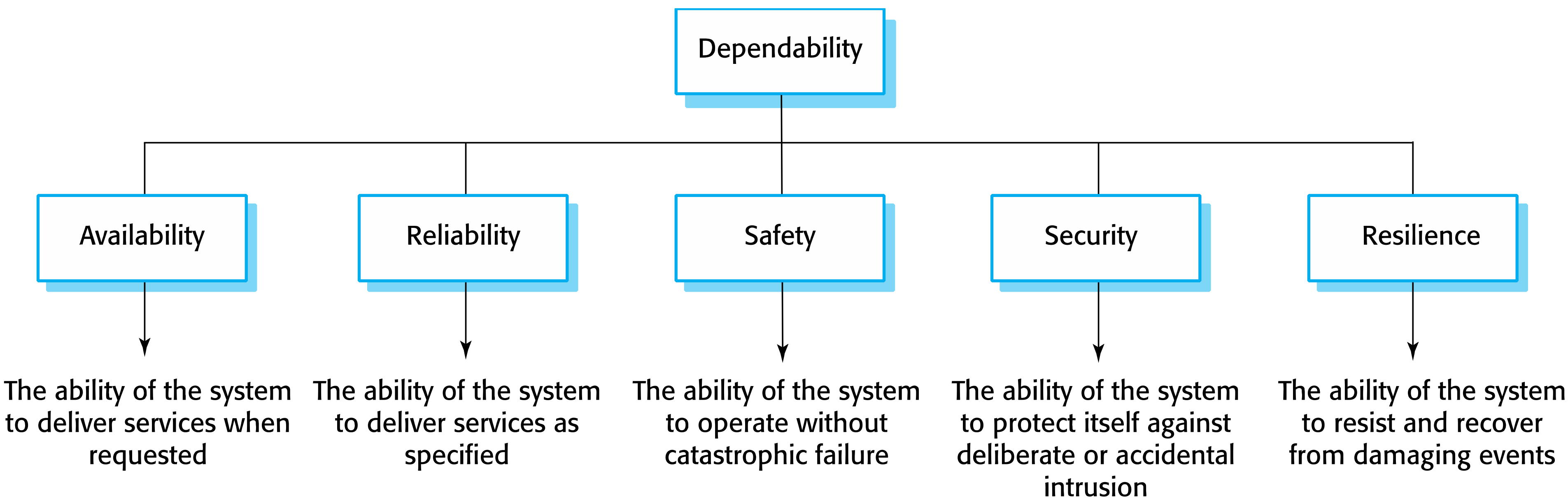
bures@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Dependability

"The ability to provide services that can defensibly be trusted within a time-period"



Department of Distributed and Dependable Systems

Principal properties

- Availability
  - The probability that the system will be up and running and able to deliver useful services to users.
- Reliability
  - The probability that the system will correctly deliver services as expected by users.
- Safety
  - A judgment of how likely it is that the system will cause damage to people or its environment.
- Security
  - A judgment of how likely it is that the system can resist accidental or deliberate intrusions.
- Resilience
  - A judgment of how well a system can maintain the continuity of its critical services in the presence of disruptive events such as equipment failure and cyberattacks.

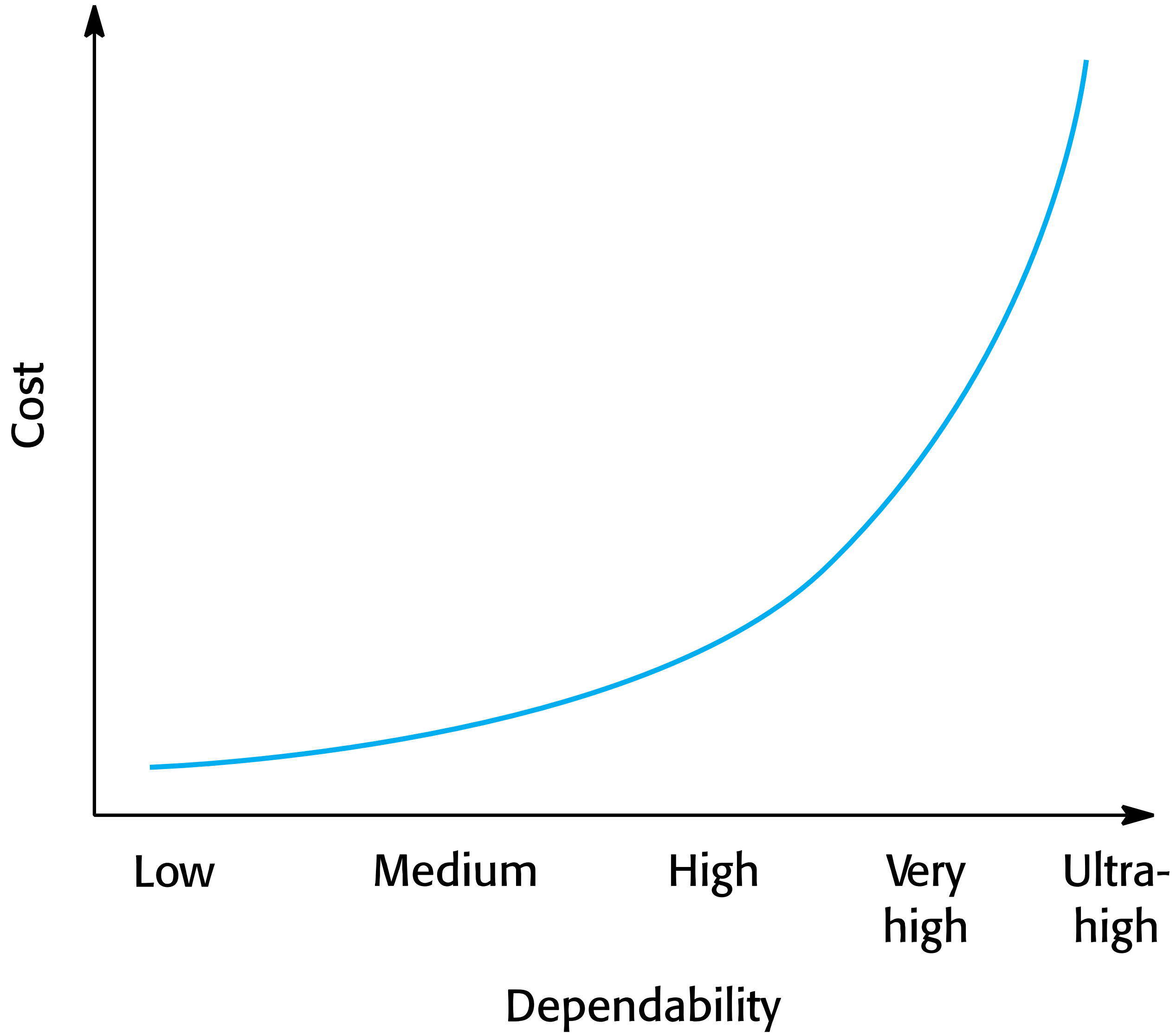
Related system properties

- Repairability
  - Reflects the extent to which the system can be repaired in the event of a failure
- Maintainability
  - Reflects the extent to which the system can be adapted to new requirements;
- Error tolerance
  - Reflects the extent to which user input errors can be avoided and tolerated.

Causes of failure

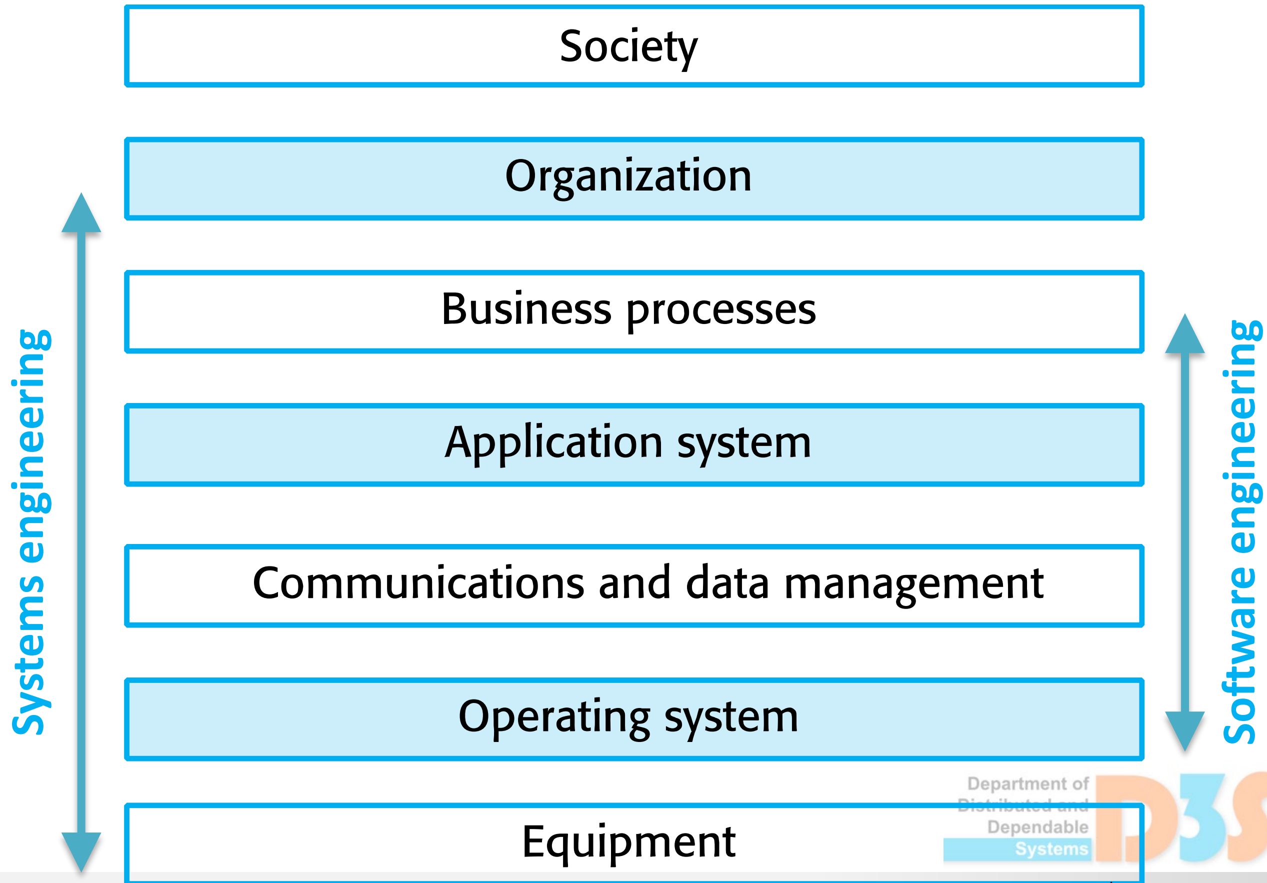
- Hardware failure
  - Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life.
- Software failure
  - Software fails due to errors in its specification, design or implementation.
- Operational failure
  - Human operators make mistakes. Now perhaps the largest single cause of system failures in socio-technical systems.

Cost/dependability curve



Sociotechnical systems stack

- Systems perspective is essential for dependability
- Necessary to understand how the layers influence the software
- … and how faults and failures propagate and multiply



Layers in the STS stack

- Equipment
  - Hardware devices, some of which may be computers. Most devices will include an embedded system of some kind.
- Operating system
  - Provides a set of common facilities for higher levels in the system.
- Communications and data management
  - Middleware that provides access to remote systems and databases.
- Application systems
  - Specific functionality to meet some organization requirements.
- Business processes
  - A set of processes involving people and computer systems that support the activities of the business.
- Organizations
  - Higher level strategic business activities that affect the operation of the system.
- Society
  - Laws, regulation and culture that affect the operation of the system.

Dependable processes

- Software process to produce dependable software
- Explicitly defined process
  - Data are collected during the process to show that the development team has followed the process
- Repeatable process
  - Does not rely on individual interpretation and judgement. Can be repeated across projects with different team members. Particularly important for critical systems which are developed over long period of time and team may change a lot.

Process characteristic	Description
Auditable	The process should be understandable by people apart from process participants, who can check that process standards are being followed and make suggestions for process improvement.
Diverse	The process should include redundant and diverse verification and validation activities.
Documentable	The process should have a defined process model that sets out the activities in the process and the documentation that is to be produced during these activities.
Robust	The process should be able to recover from failures of individual process activities.
Standardized	A comprehensive set of software development standards covering software production and documentation should be available.

Dependable processes

- Activites
  - Requirements review
  - Requirements management
  - Formal specification
  - System modeling
  - Design and program inspections
  - Static analysis
  - Test planning and management

Formal methods and dependability

- Formal methods – mathematical approaches to verification of software
- Typically based on formal models of software behavior
- Effective for discovering or avoiding:
  - Specification and design errors and omissions
  - Inconsistencies between a specification and a program

- More at:
  - NSWI132 (Program Analysis and Code Verification)
    - Summer term – 2/2 Zk+Z
    - Lecture: Wed 10:40 S1
    - Lab: Wed 12:20 SU1
  - NTIN043 (Formal Foundations of Software Engineering)
    - Winter term – 2/2 Zk+Z
  - NSWI101 (System Behavior Models and Verification)
    - Winter term – 2/2 Zk+Z

Dependability in timing

- Dependable systems are often real-time and embedded
- Requires special design and reasoning to give guarantees about timing

- More at NSWE001 (Embedded and Real-time Systems)

Department of Distributed and Dependable Systems

Why Dependability?

Arianne 5

- Exploded on June 4, 1996
  - only 39 seconds after launch
  - loss of about US$ 370 million
- A 64-bit float was truncated to 16-bit integer in a “non-critical software component”
- This caused unhandled hardware exception
- The erroneous component (a method) was inherited/reused from Ariane 4 and had no practical use in Ariane 5

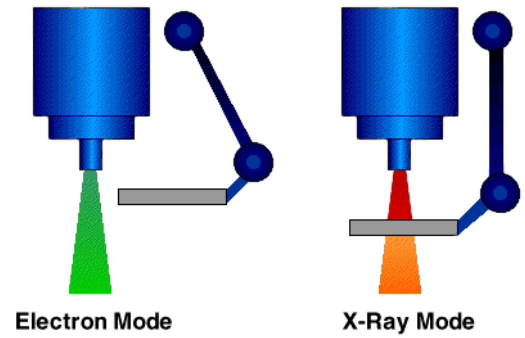
Department of Distributed and Dependable Systems

Patriot – Failure at Dhahran

- February 25, 1991, an Iraqi Scud hit the barracks in Dhahran killing 28 soldiers
- The area was protected by Patriot aerial interceptor missiles
- Due to drift of system's internal
  - by one third of a second in 100 hours
  - amounted to miss distance of 600 meters
  - The system detected the missile but due to the time skew, it disregarded it as spurious

Therac-25

- Computer controller radiation therapy machine
- 6 accidents 1985-1987
  - three people died as the direct consequence of radiation burns
- Race condition as the primary cause
- Other causes included
  - Poor design, no review of the software
  - Bad man-machine interface
  - Overconfidence in the software
  - Not understanding safety
    - The software was in use previously, but different hardware design covered its flaws



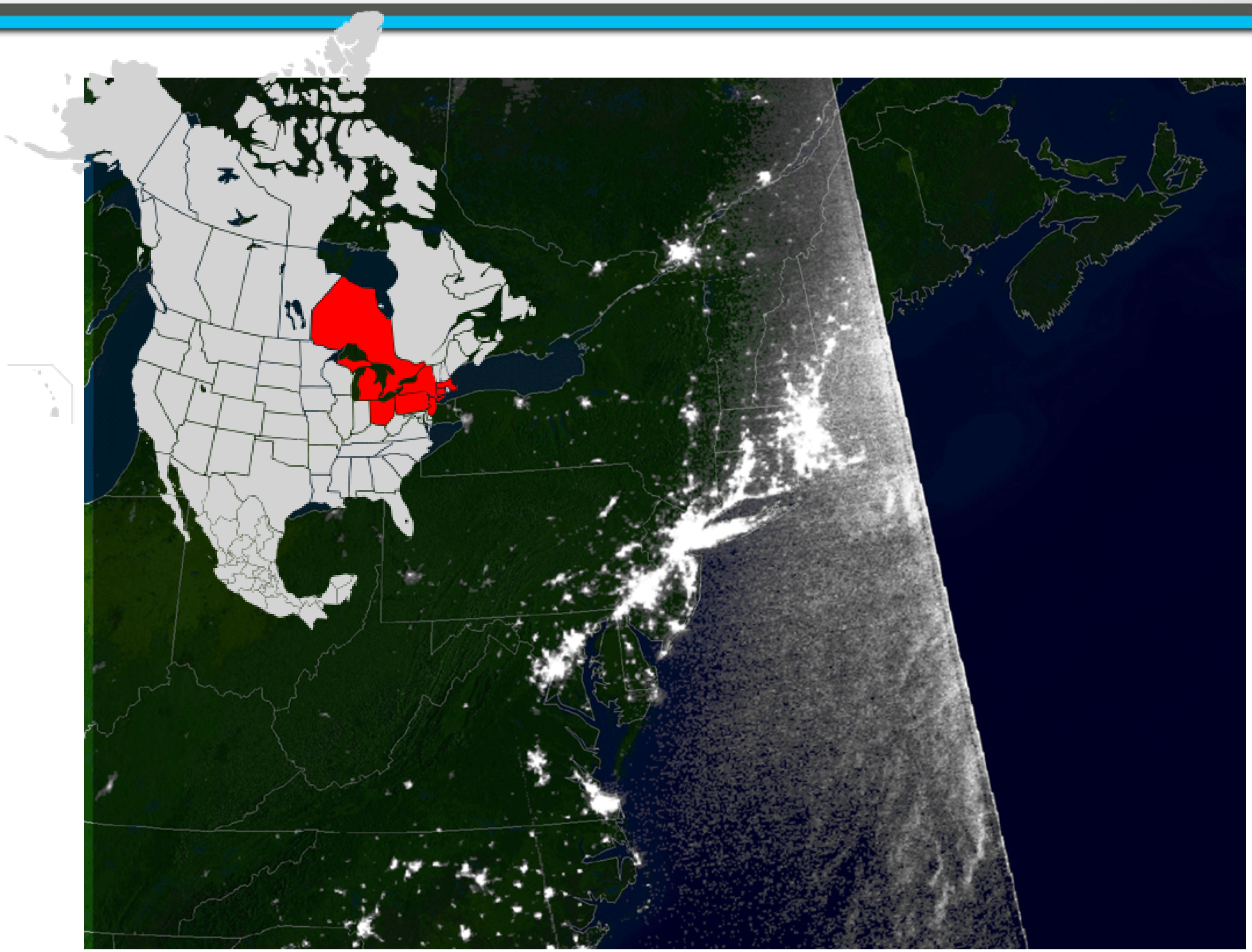

```
PATIENT NAME    : JOHN DOE
TREATMENT MODE : FIX  BEAM TYPE: X    ENERGY (MeV): 25

                      ACTUAL          PRESCRIBED
    UNIT RATE/MINUTE    0                 200
    MONITOR UNITS      50 50              200
    TIME (MIN)         0.27               1.00

GANTRY ROTATION (DEG)      0.0             0.0     VERIFIED
COLLIMATOR ROTATION (DEG) 349.2            359     VERIFIED
COLLIMATOR X (CM)          13.2            14.3    VERIFIED
COLLIMATOR Y (CM)          21.2            27.3    VERIFIED
WEDGE NUMBER                 1               1     VERIFIED
ACCESSORY NUMBER             0               0     VERIFIED

DATE   : 84-DEC-27   SYSTEM : BEAM READY    OP. MODE : TREAT AUTO
TIME   : 12:55: 8    TREAT  : TREAT PAUSE              X-RAY 173777
OPR ID : T25V02-R03  REASON : OPERATOR       COMMAND:
```

Total Blackout in 2003



2003 blackout in 8 states of USA and in Ontario (in total 55 mil. people affected), primary culprit was a SW problem (race condition) which delayed notifications

F22 Raptor Software Bug

2007 F22 Raptor software bug

navigation, communication and fuel systems crashed after crossing the international date line

Jeep Cherokee Hack

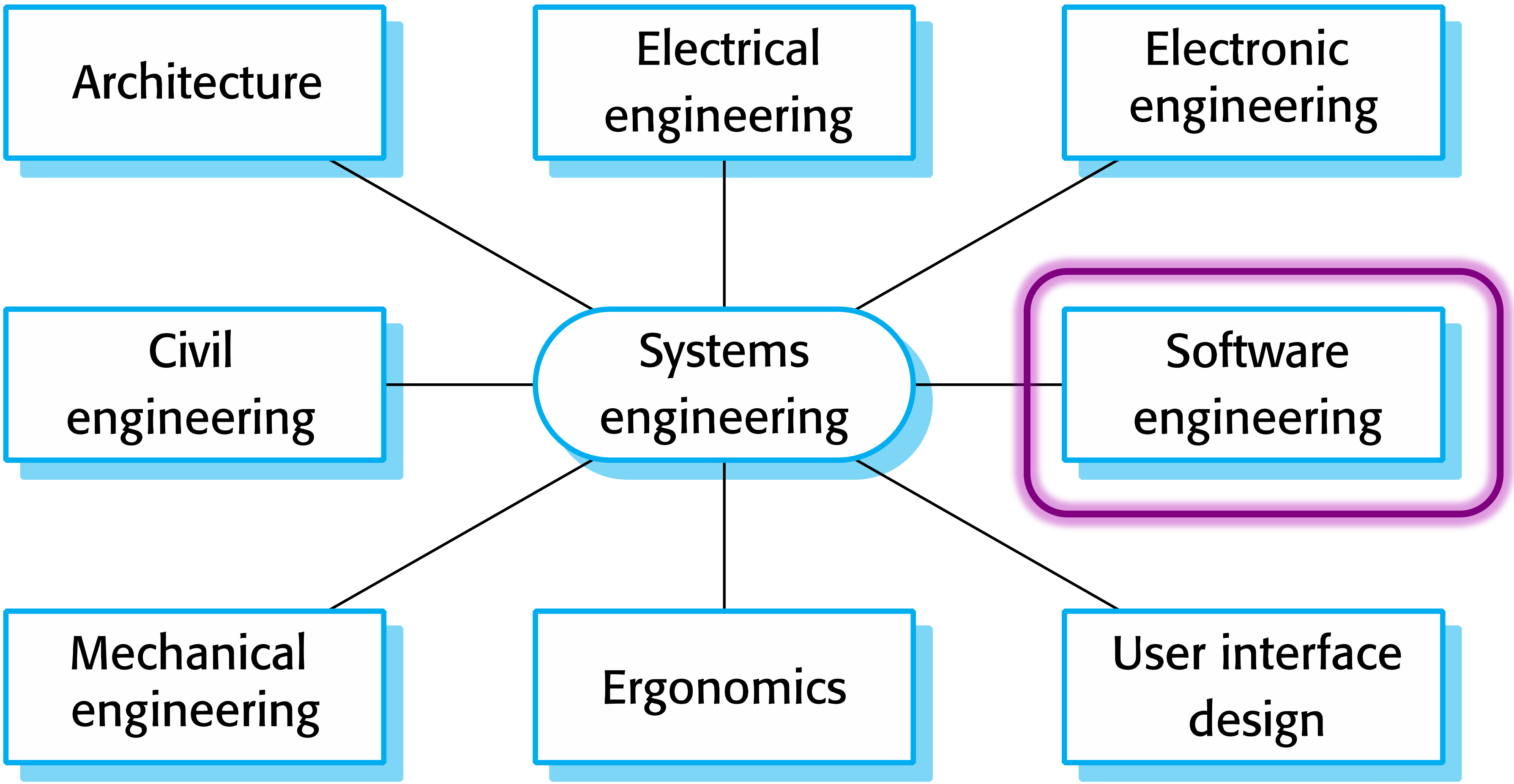
2014 Jeep Cherokee wirelessly hacked

security flaw in the infotainment system which allowed an attacker to control almost everything (including throttle, brakes and steering) over internet

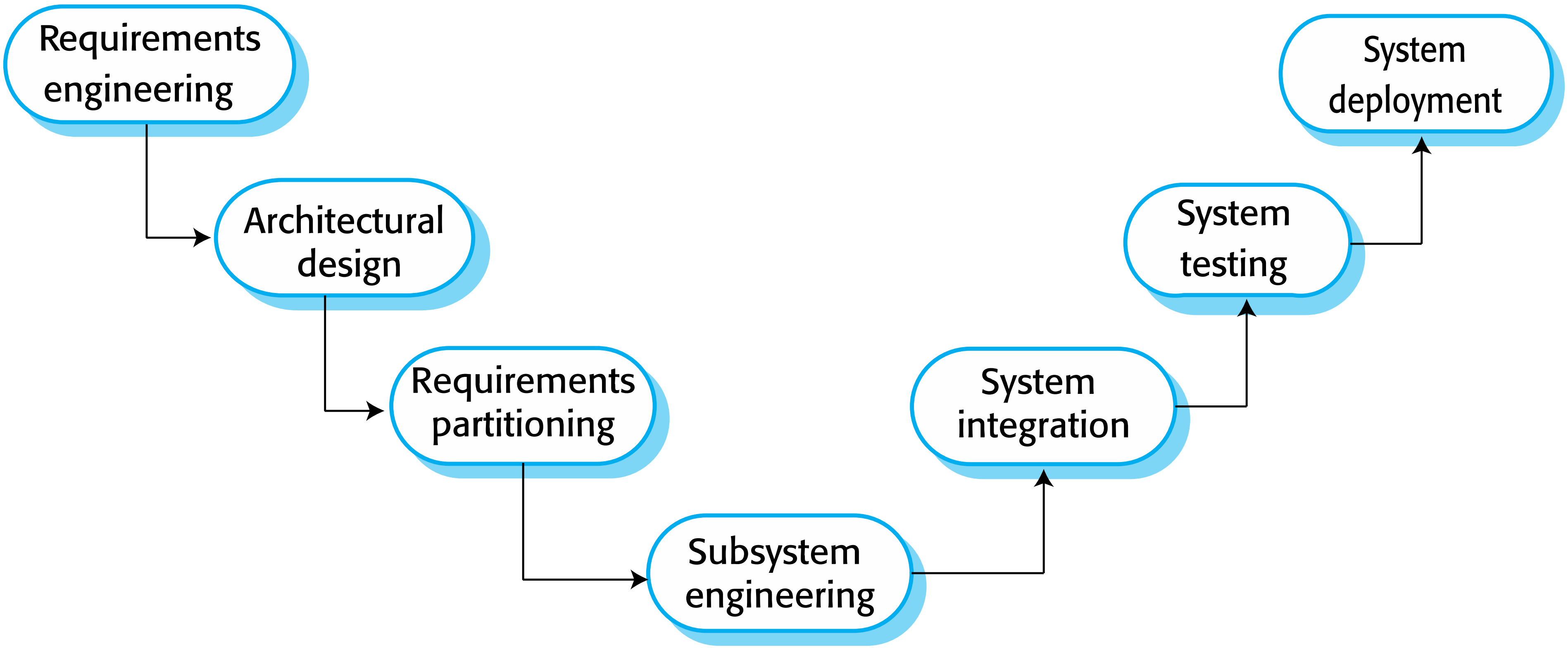
Systems Engineering

Department of
Distributed and
Dependable
Systems

Professional disciplines involved



Systems development



The system development process

- Requirements engineering
  - The process of refining, analysing and documenting the high-level and business requirements identified in the conceptual design
- Architectural design
  - Establishing the overall architecture of the system, identifying components and their relationships
- Requirements partitioning
  - Deciding which subsystems (identified in the system architecture) are responsible for implementing the system requirements

The system development process

- Subsystem engineering
  - Developing the software components of the system, configuring off-the-shelf hardware and software, defining the operational processes for the system and re-designing business processes
- System integration
  - Putting together system elements to create a new system
- System testing
  - The whole system is tested to discover problems
- System deployment
  - the process of making the system available to its users, transferring data from existing systems and establishing communications with other systems in the environment

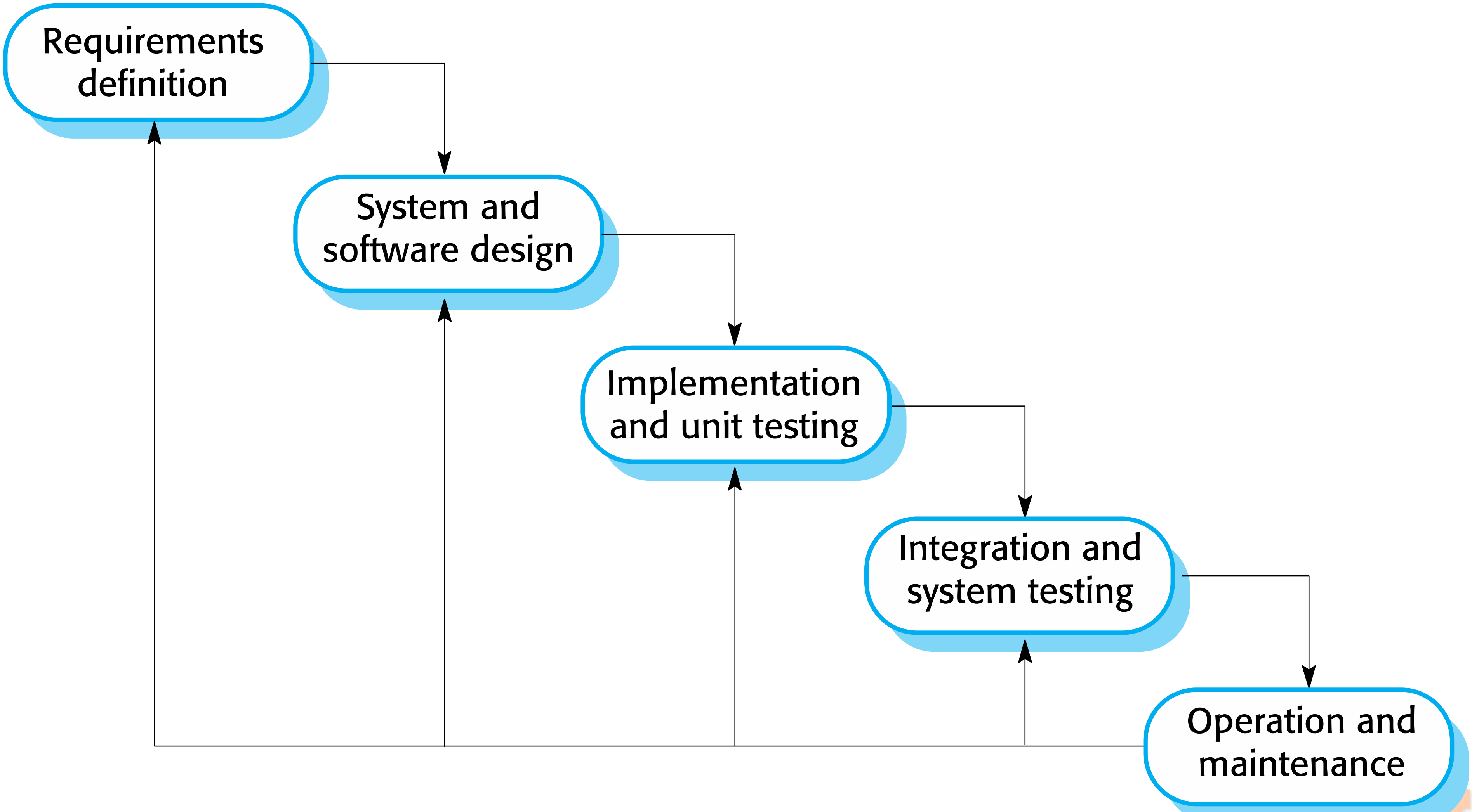
Software engineering

Software process

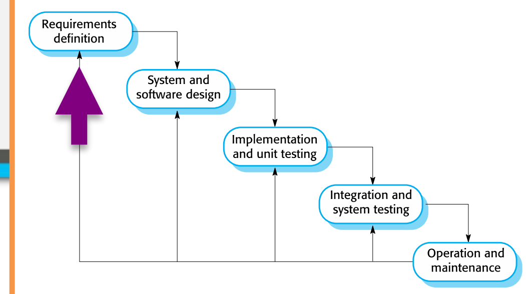
- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.

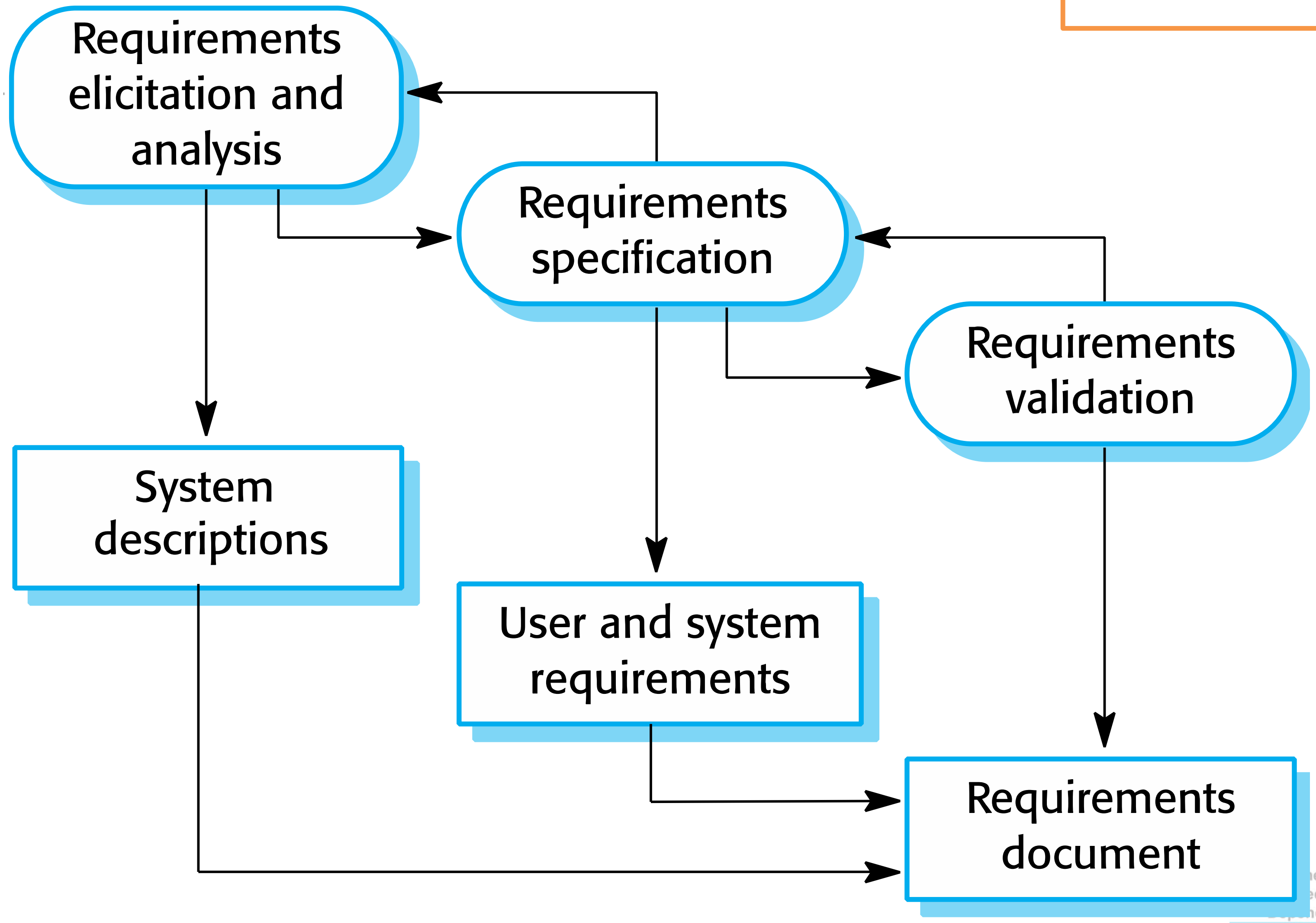
Department of Distributed and Dependable Systems

The waterfall model

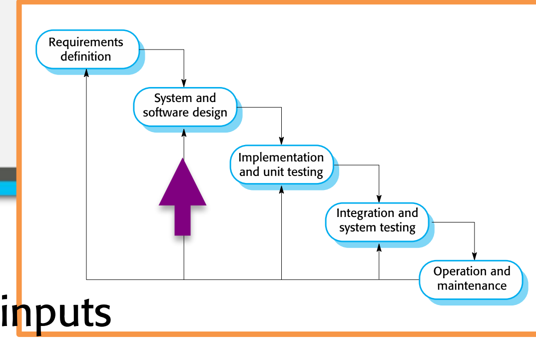


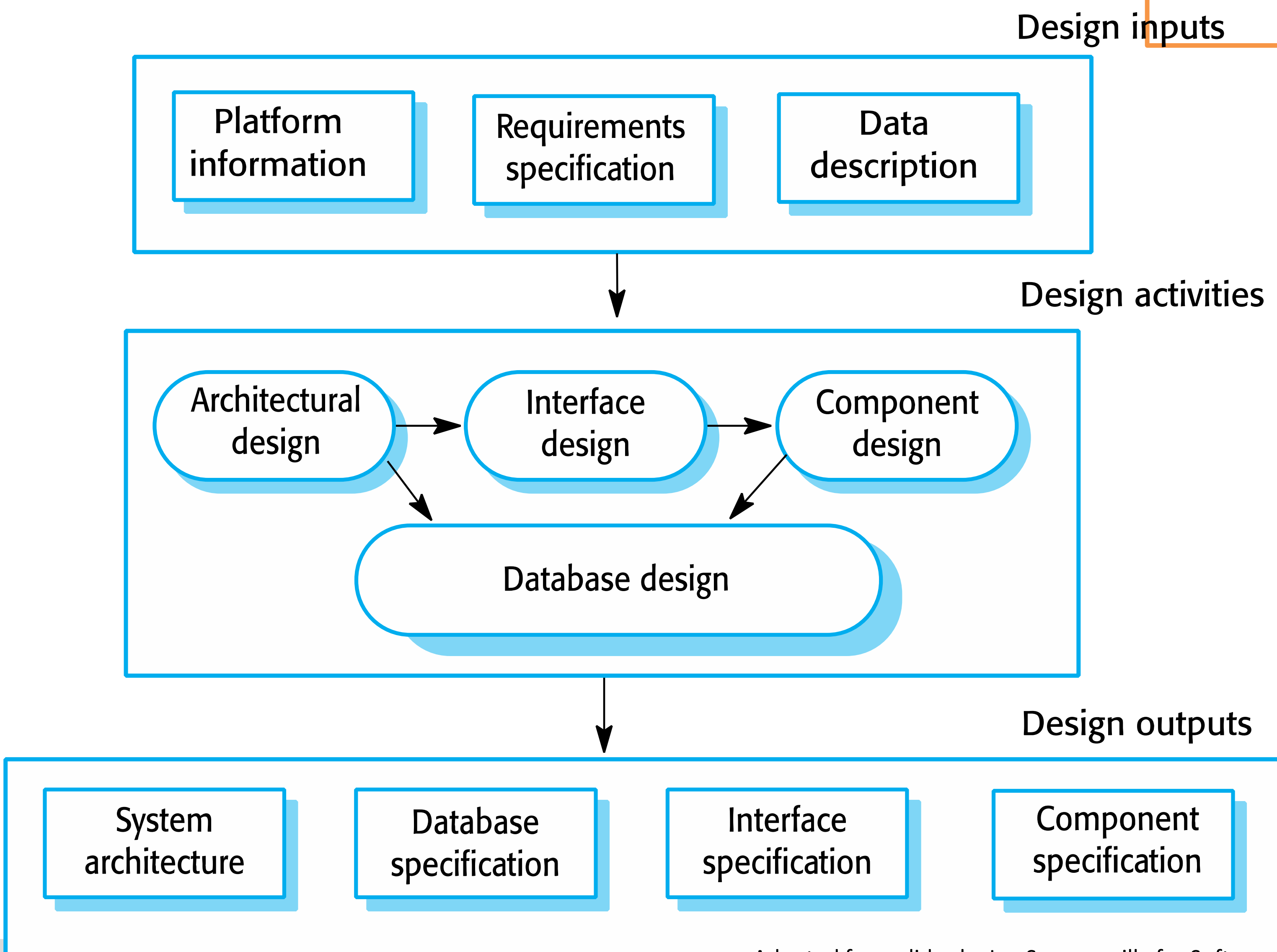
Requirements engineering process



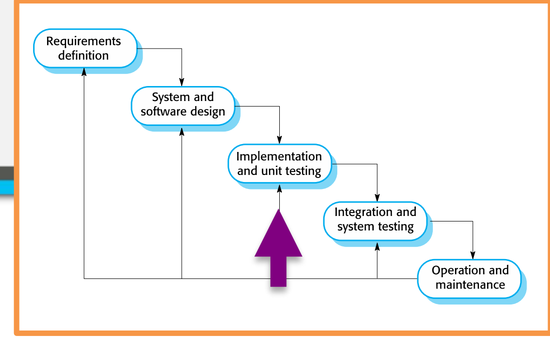


Software design process





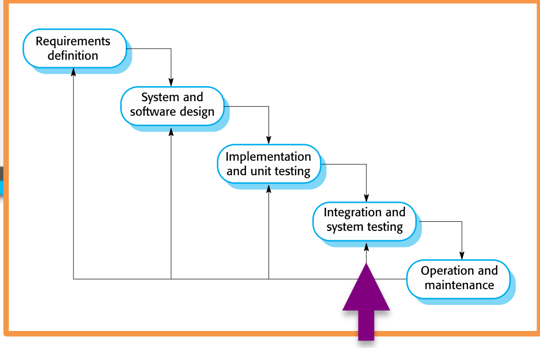
System implementation



- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

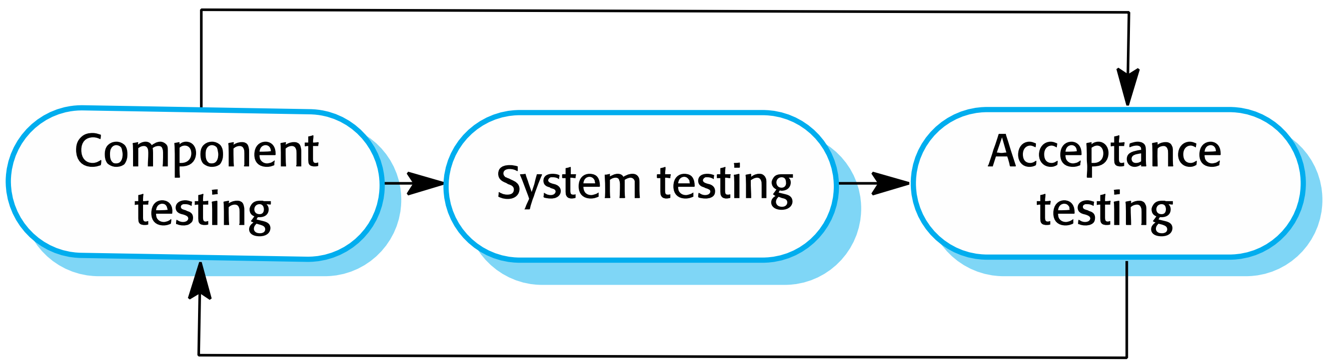
Department of Distributed and Dependable Systems

Software validation



- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

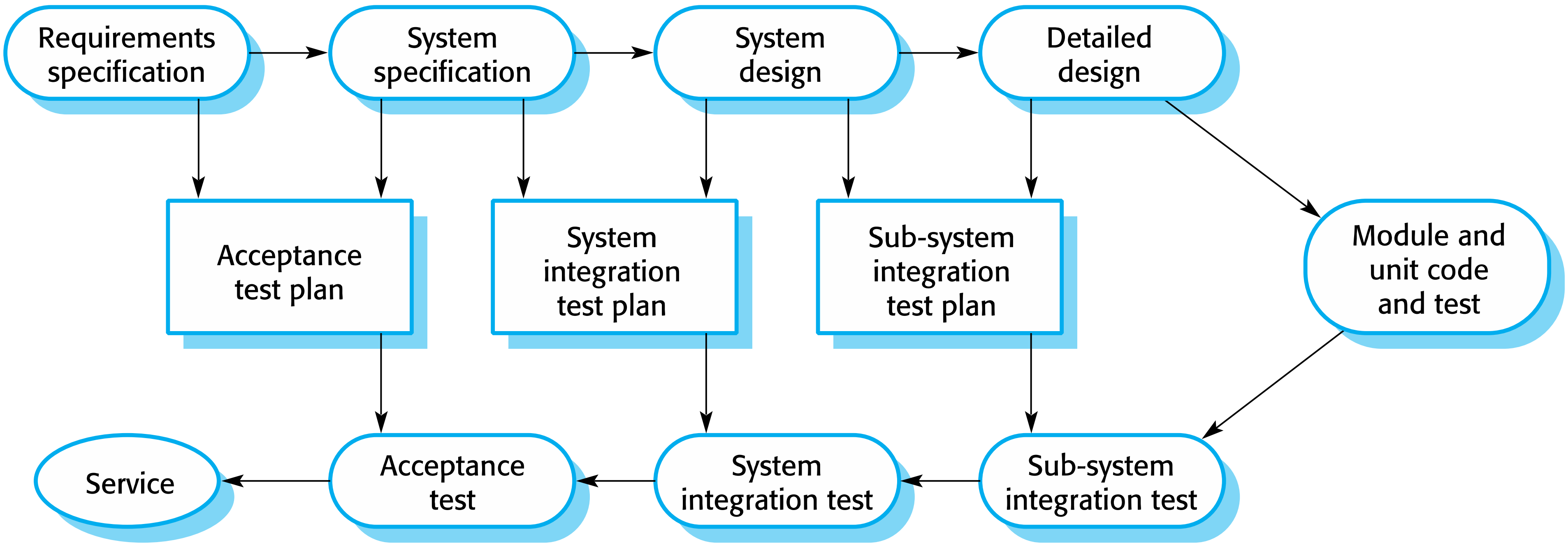
Stages of testing



- Component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Customer testing
  - Testing with customer data to check that the system meets the customer's needs.

Testing phases in a V-model

V-model – a plan driven development process



Requirements Engineering 1

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

D3S

Software Engineering for Dependable Systems

Tomas Bures

bures@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Requirements Engineering

- The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.
- The system requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process.

Types of Requirements

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Department of Distributed and Dependable Systems

User and Systems Requirements

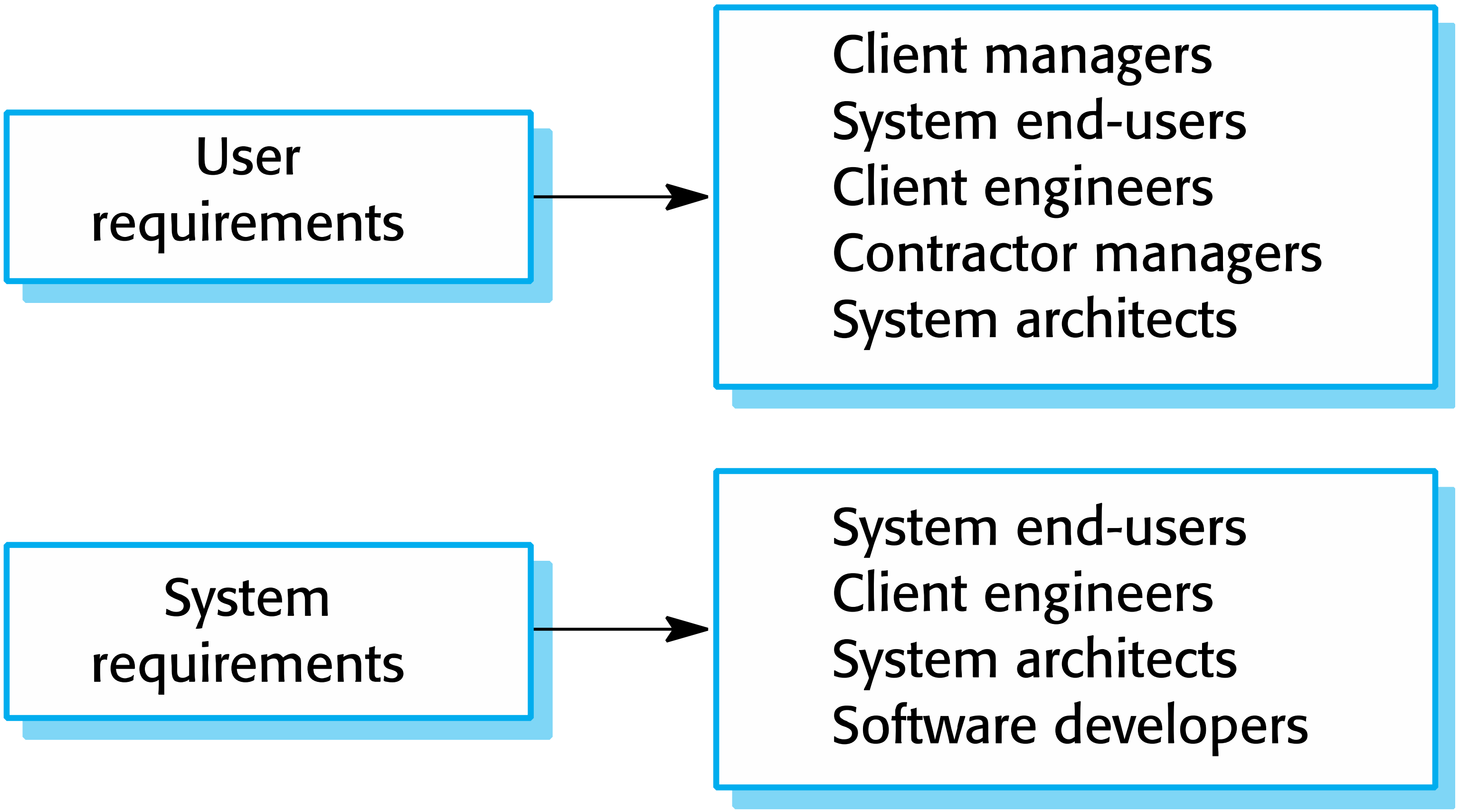
User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

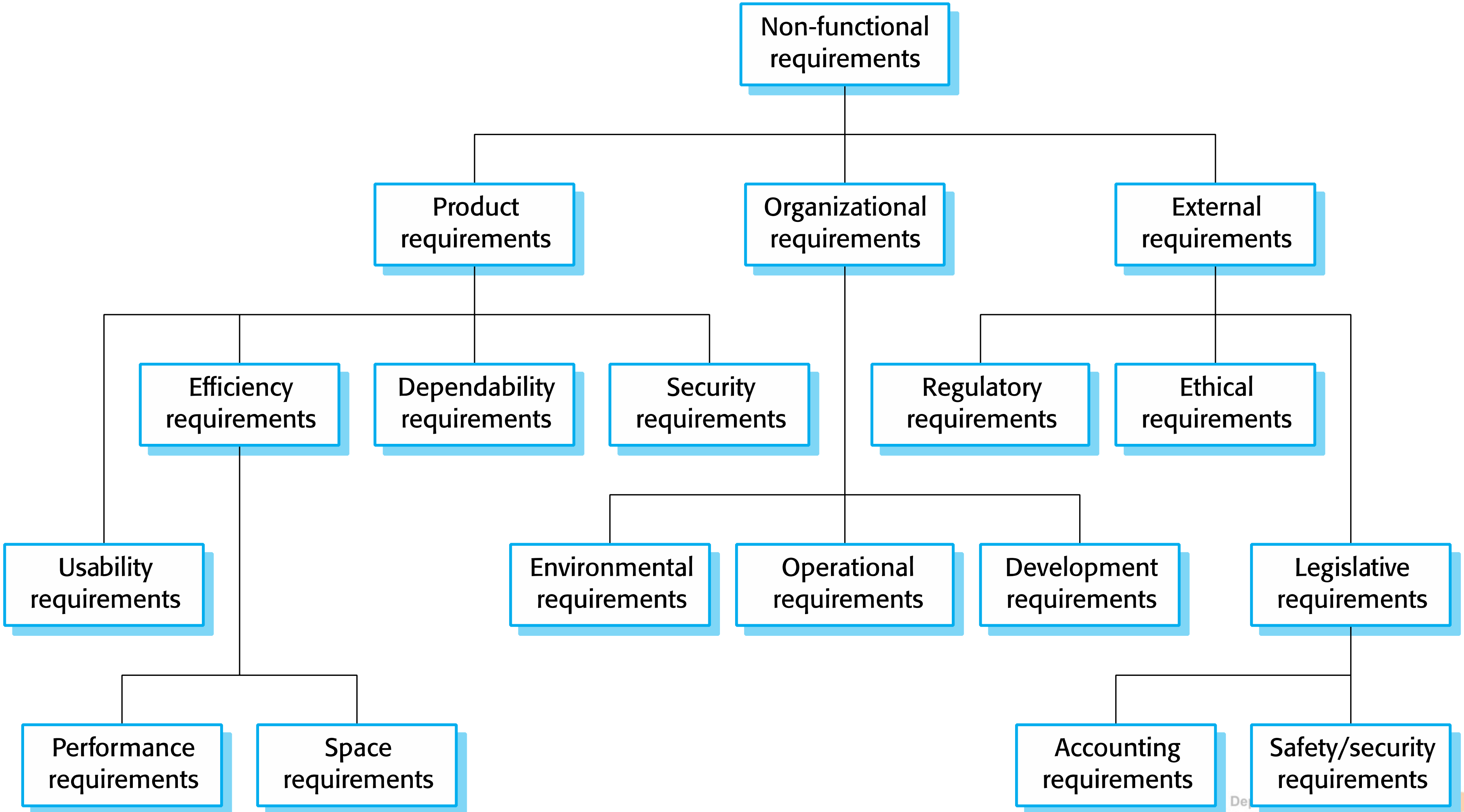
User and Systems Requirements



Functional and non-functional requirements

- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
  - Constraints on the system from the domain of operation

Types of non-functional requirements



Examples of nonfunctional requirements

Product requirement
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

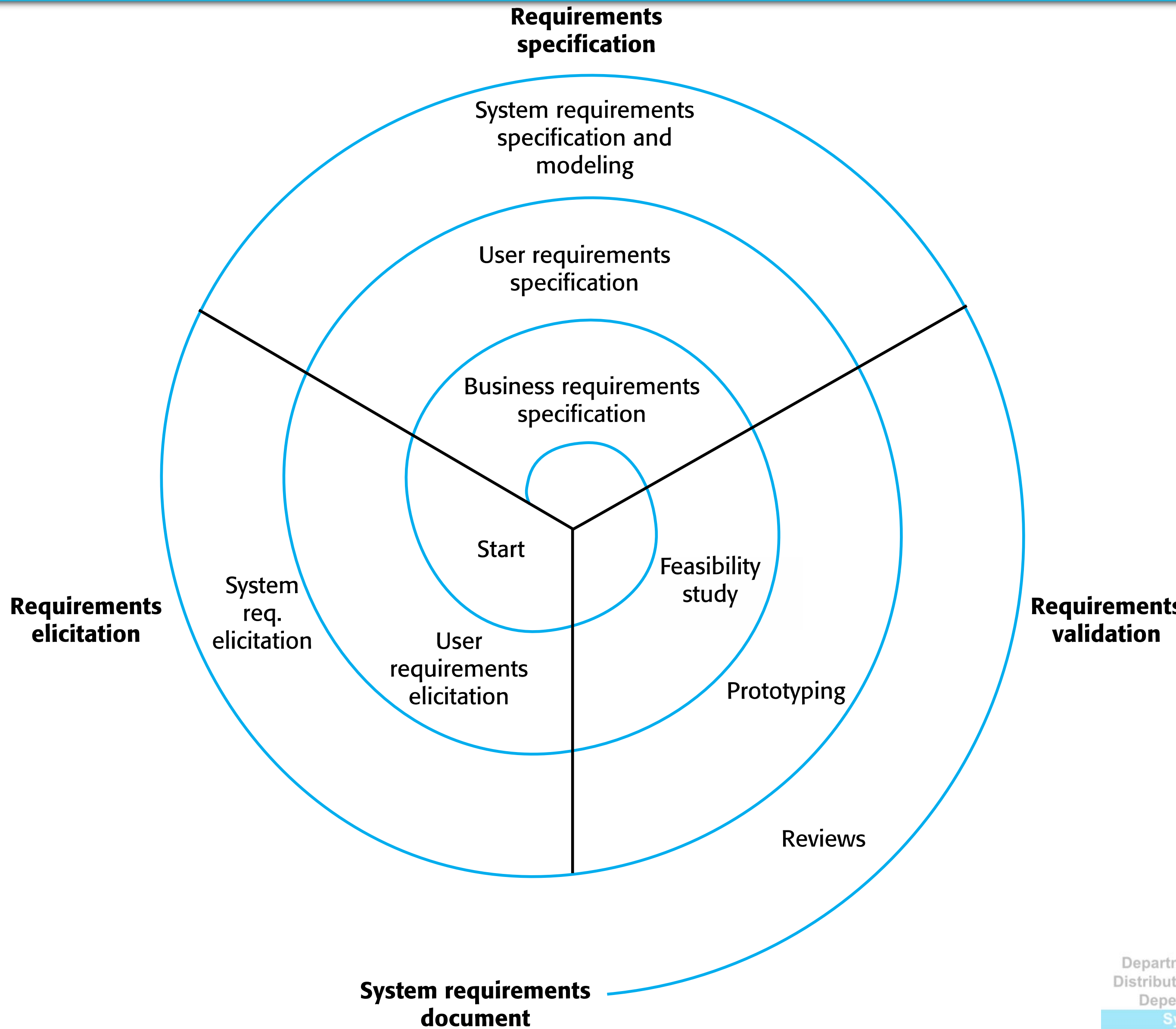
Requirements engineering processes

- Requirements elicitation & analysis
  - Requirements discovery
  - Requirements classification and organization
  - Requirements prioritization and negotiation
  - Requirements specification

- Requirements validation
  - Ensures
    - Validity – does the system provide the functions which best support the customer's needs?
    - Consistency – are there any requirements conflicts?
    - Completeness – are all functions required by the customer included?
    - Realism – can the requirements be implemented given available budget and technology
    - Verifiability – can the requirements be checked?
  - By requirement reviews, prototyping, test-case generation

- Requirements management

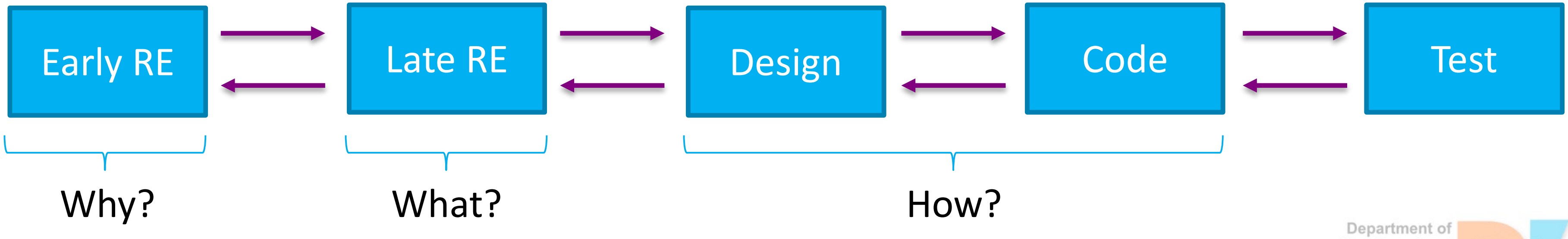
Spiral view of the RE process



Goal-based elicitation and analysis of requirements

Goals

- Goal – a stakeholder objective for the system
- Model of goals hierarchy and their relation
  - Answers the question "Why?"
- Guides requirement elaboration
- Results into concrete requirements which can be turned to Requirements Specification document

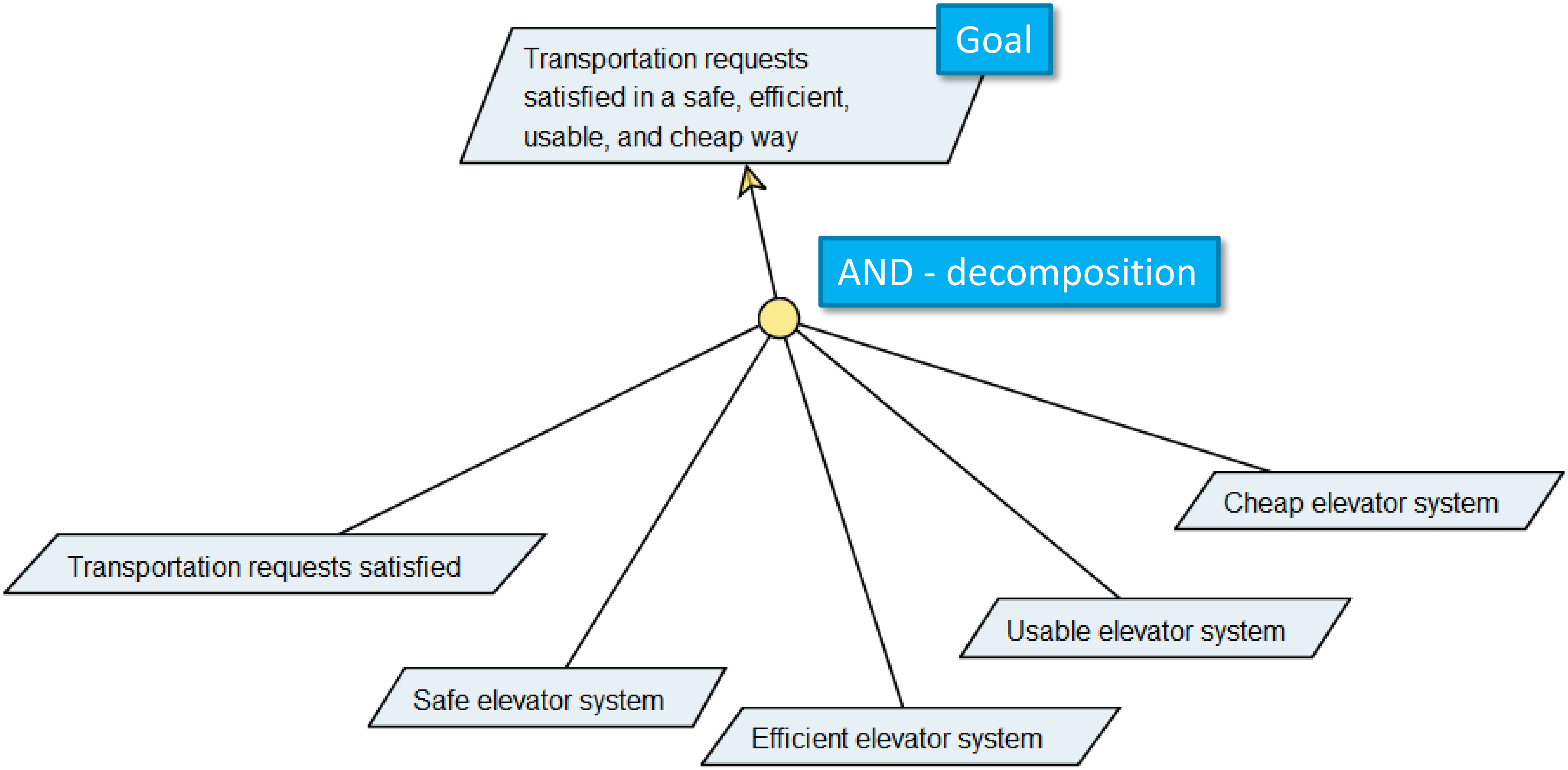


KAOS

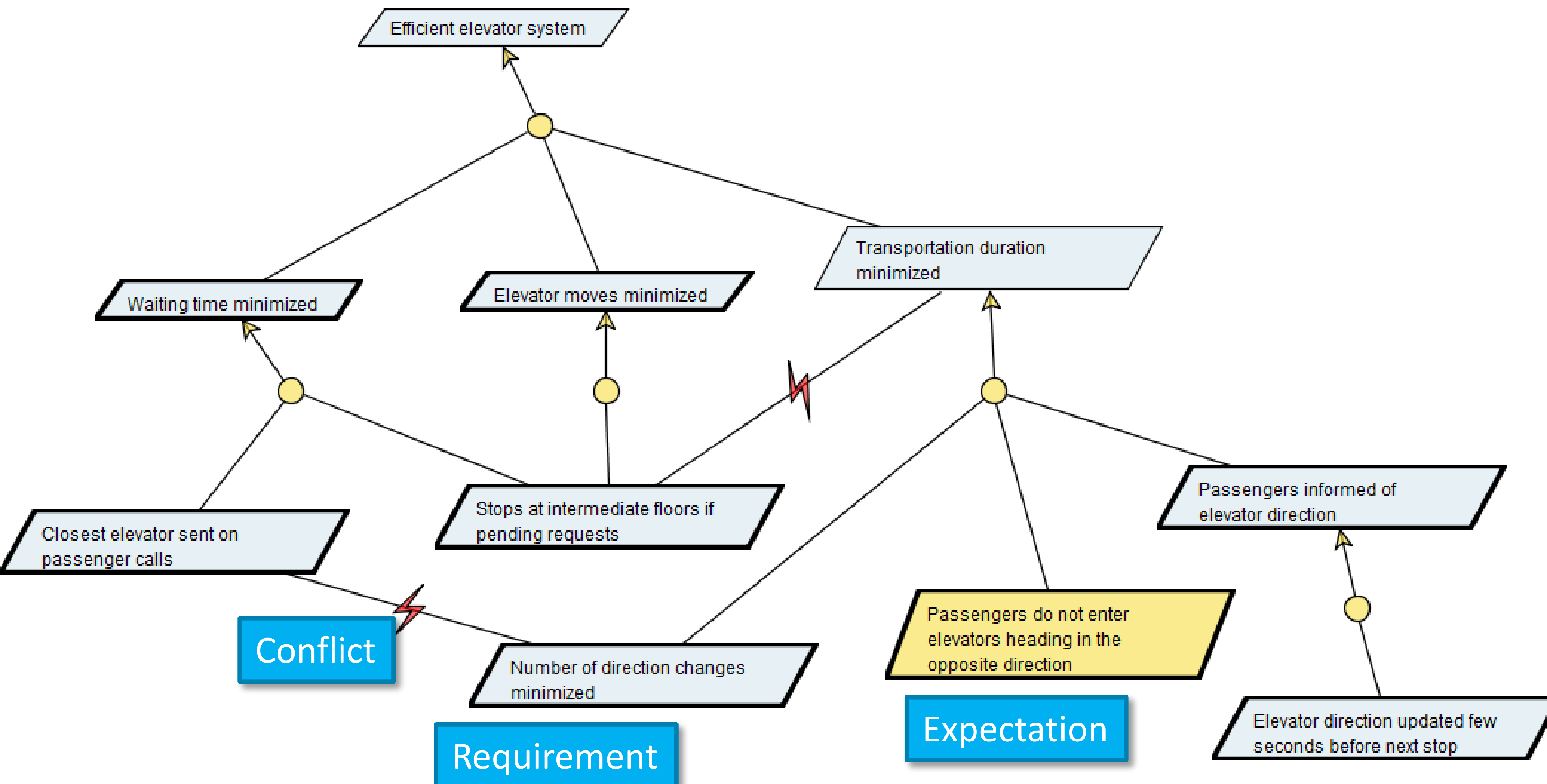
- A methodology for goal-based requirements engineering
- http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf

- Models:
  - Goal model
  - Object model
  - Agent responsibility model
  - Operation model

KAOS – Goal model

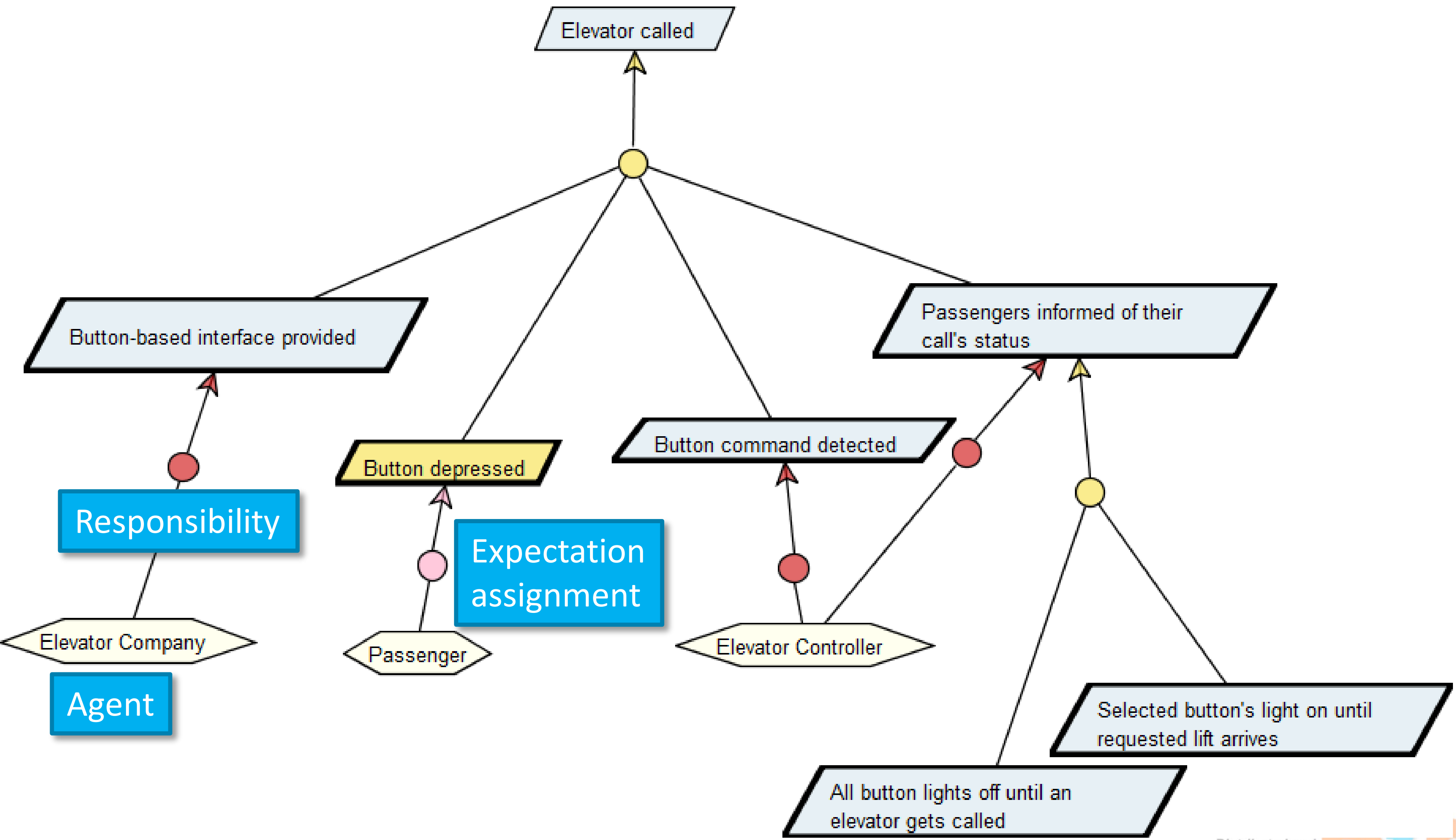


KAOS – Goal model



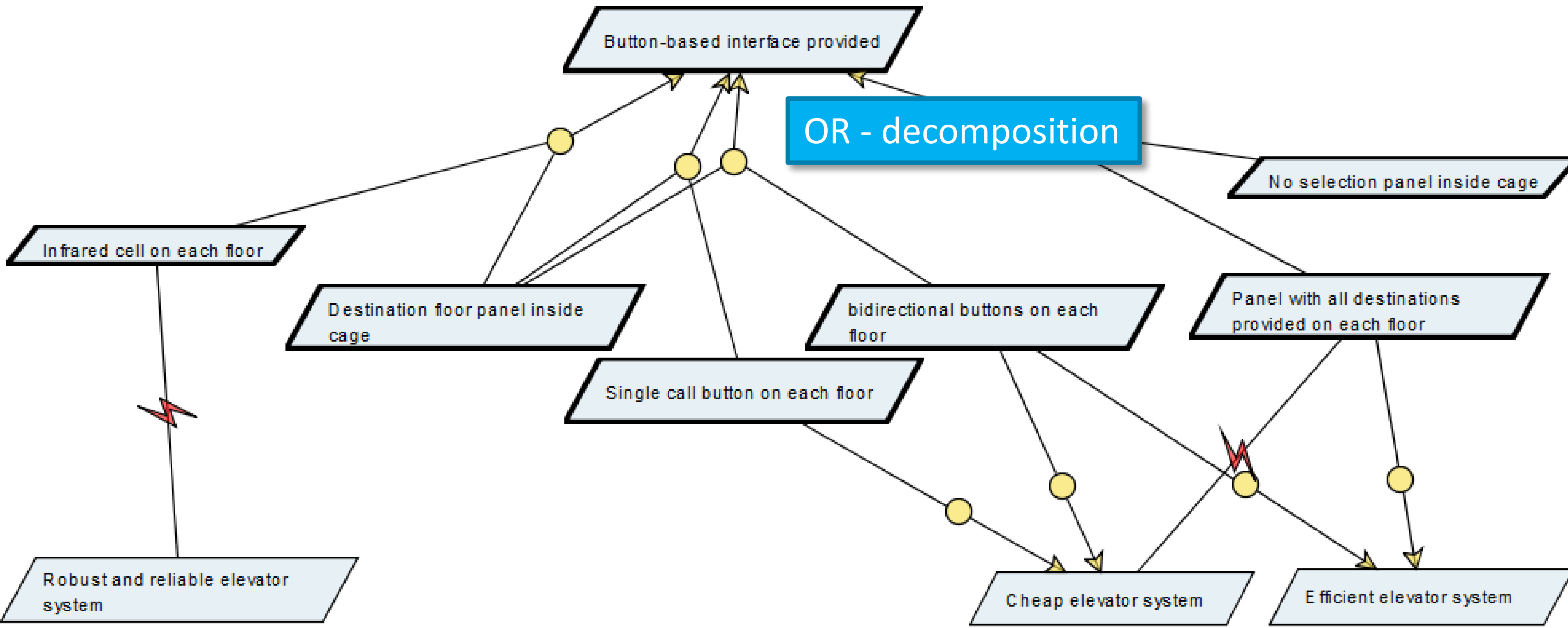
Department of Distributed and Dependable Systems

KAOS – Goal model



Distributed and Dependable Systems

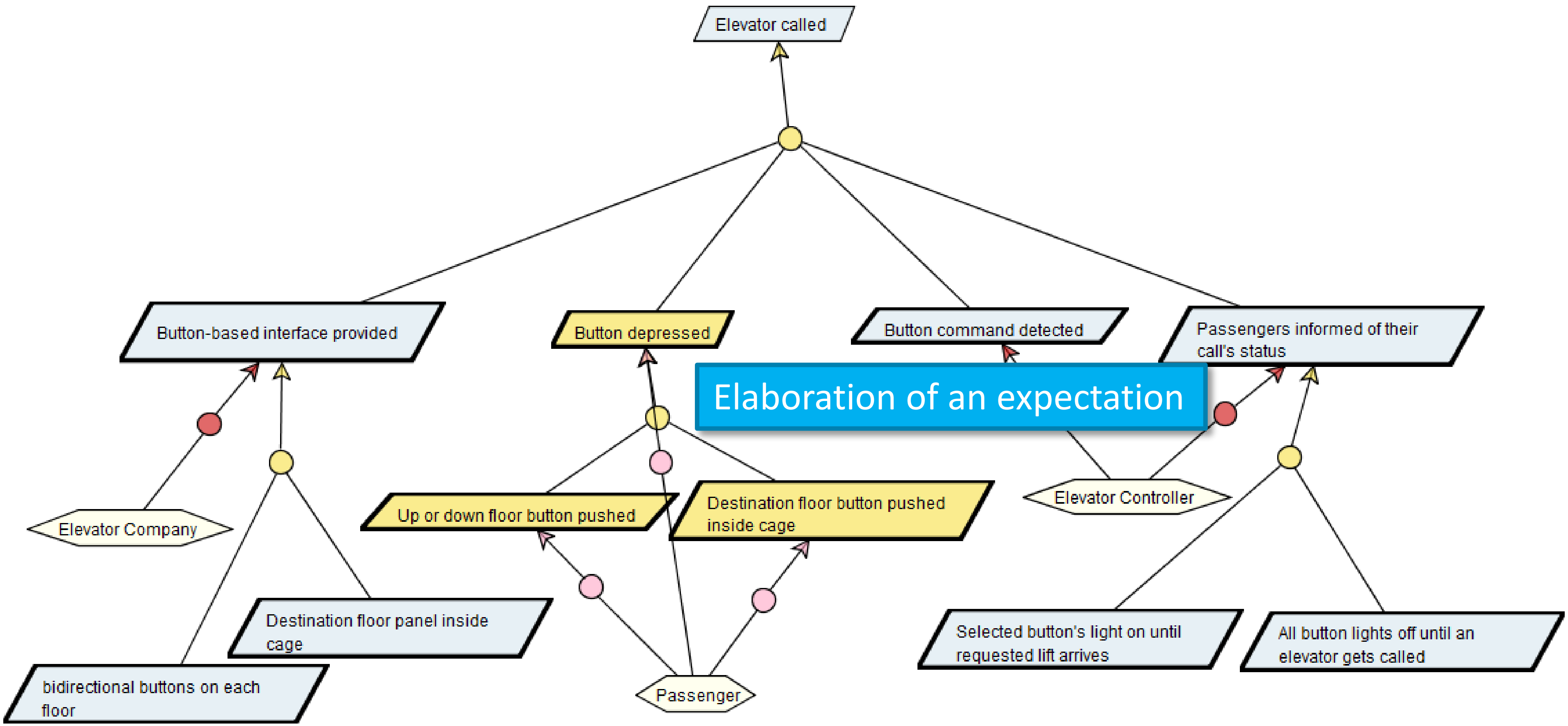
KAOS – Goal model



OR - decomposition

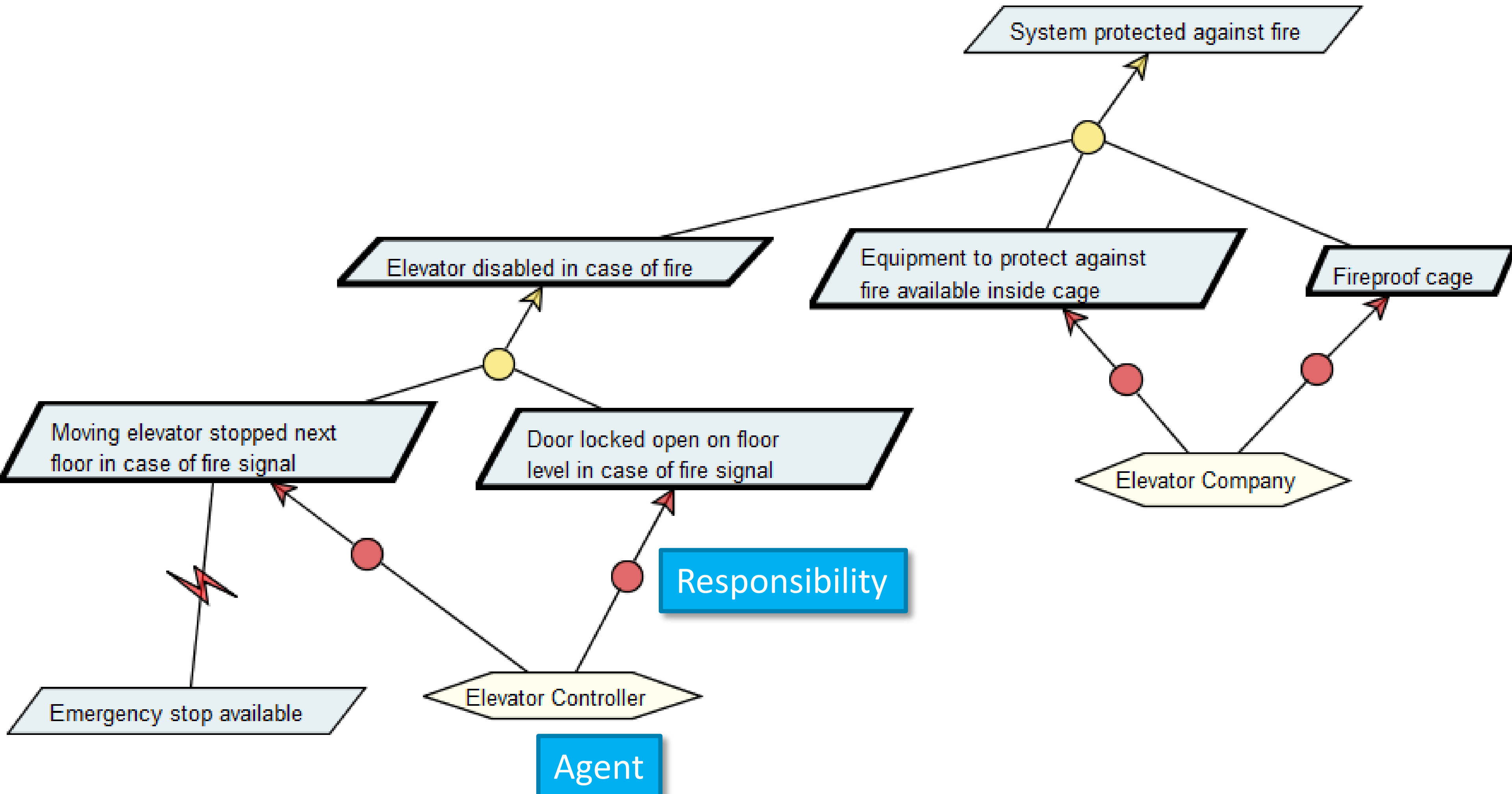
Qualitative goals to enable selection of a particular decomposition

KAOS – Goal model



Department of Distributed and Dependable Systems

KAOS – Goal model



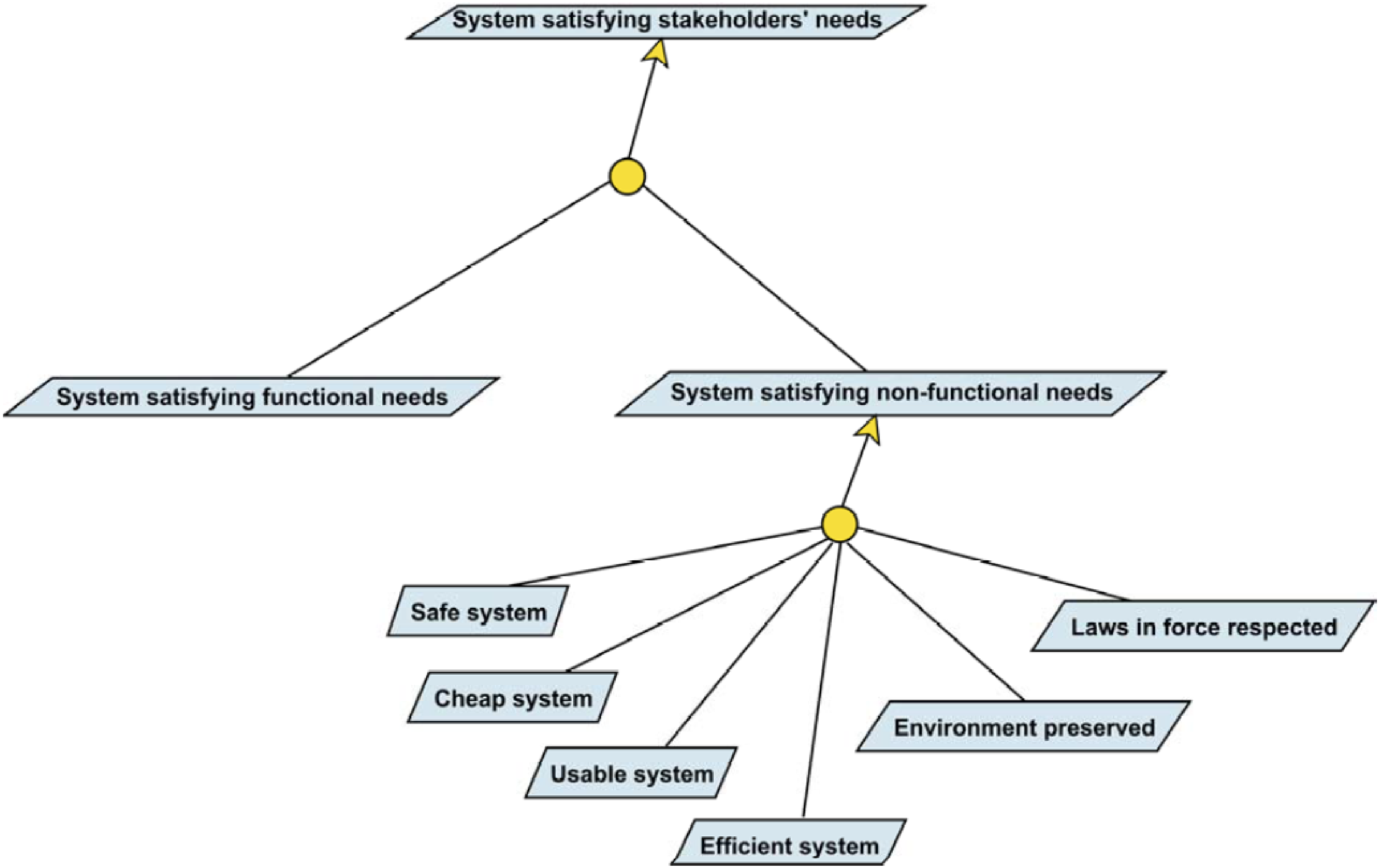
Goal patterns

- Achieve – achieve the goal at some point in the future
- Cease – undo a goal at some point in the future
- Maintain – maintain a goal for some time
- Avoid – prevent a goal from becoming true
- Optimize – maximize or minimize some measure (soft-goal)

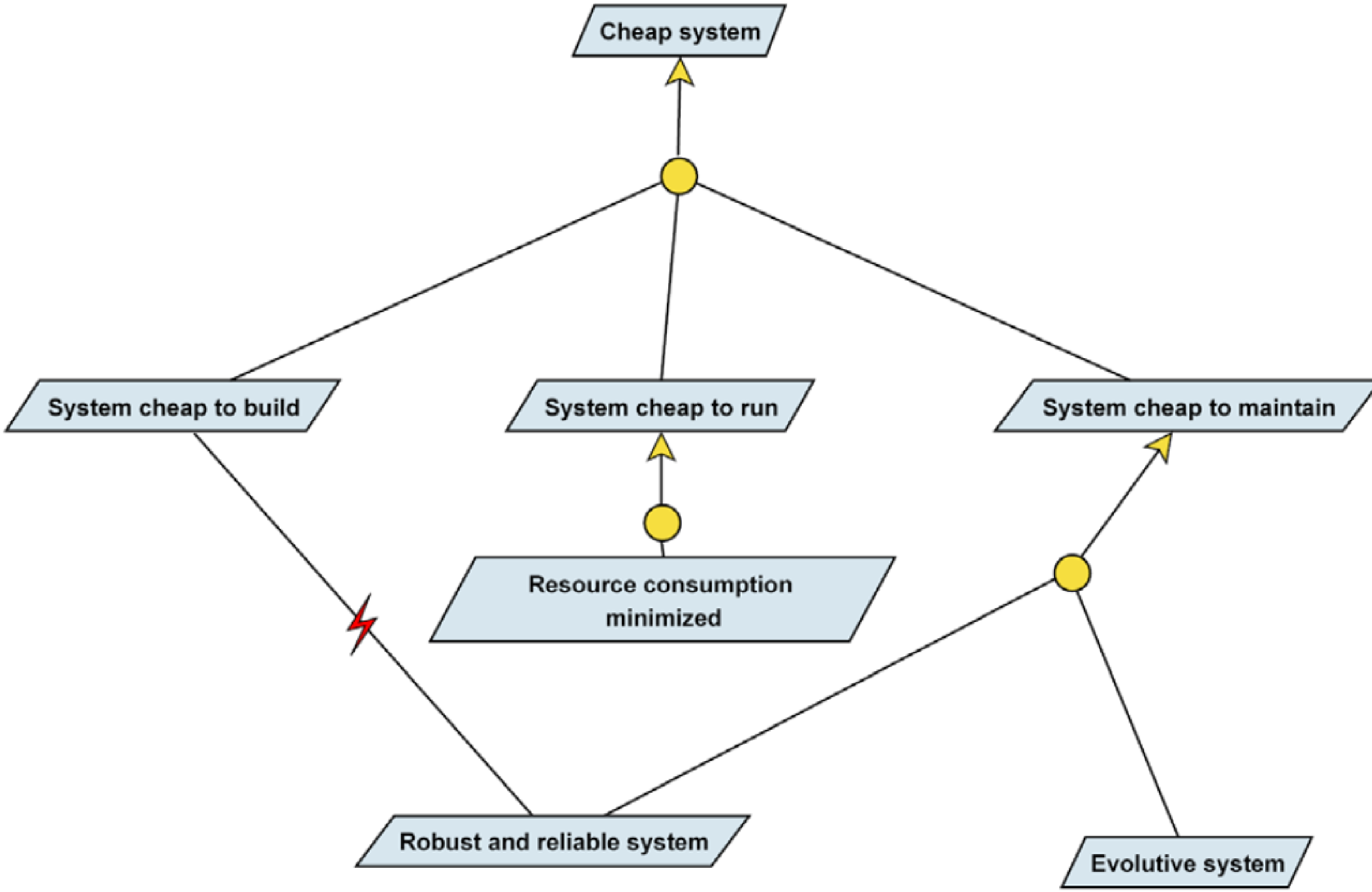
Tactics for decomposing goals

- Case-driven decomposition
  - E.g. *The goal of the system is to build a system that satisfies all stakeholders' needs: functional and non-functional ones.*
- Milestone driven decomposition
  - E.g. *System is cheap to build, (then) cheap to run, and (then) cheap to maintain.*

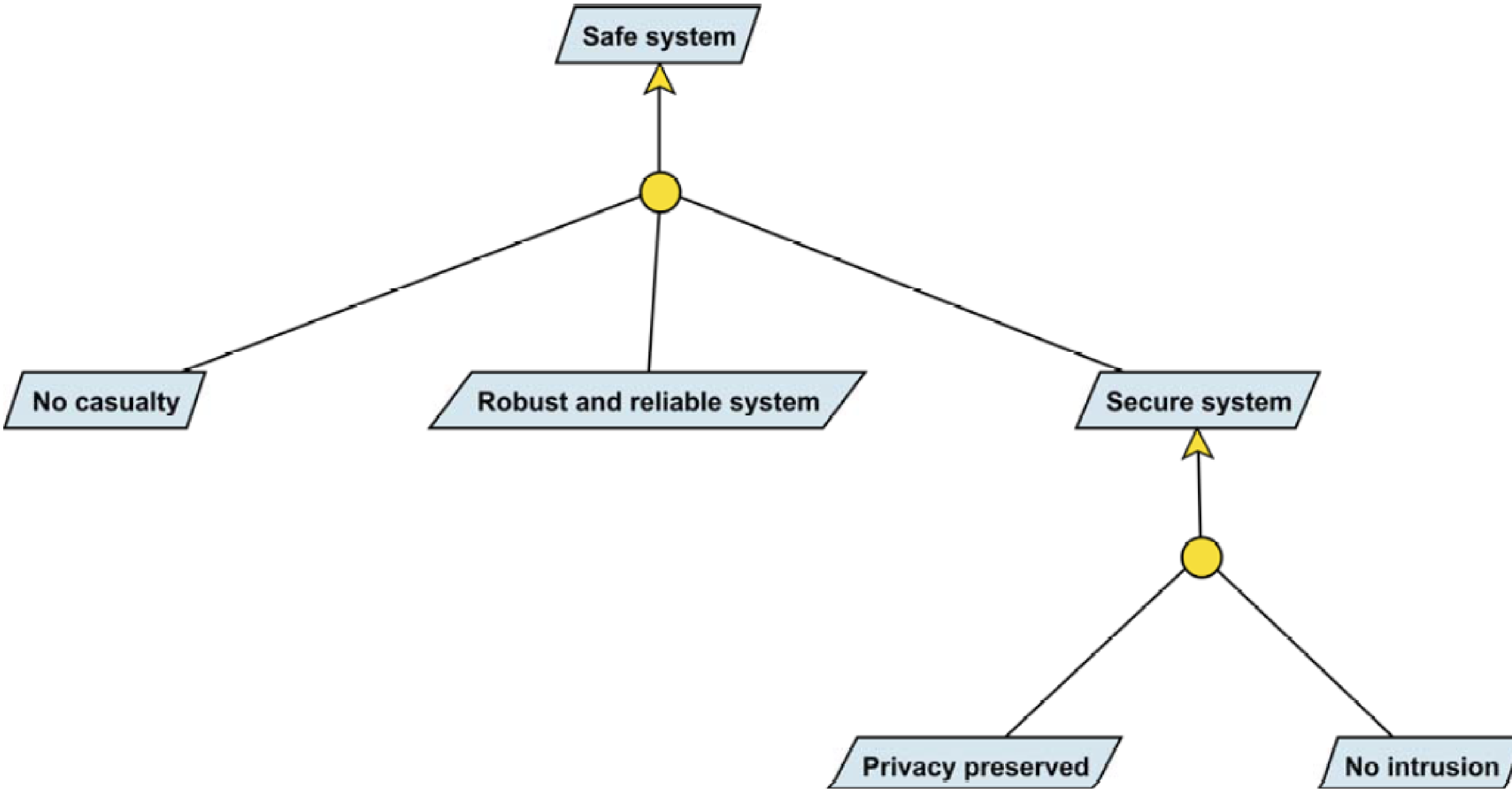
Generic goal patterns



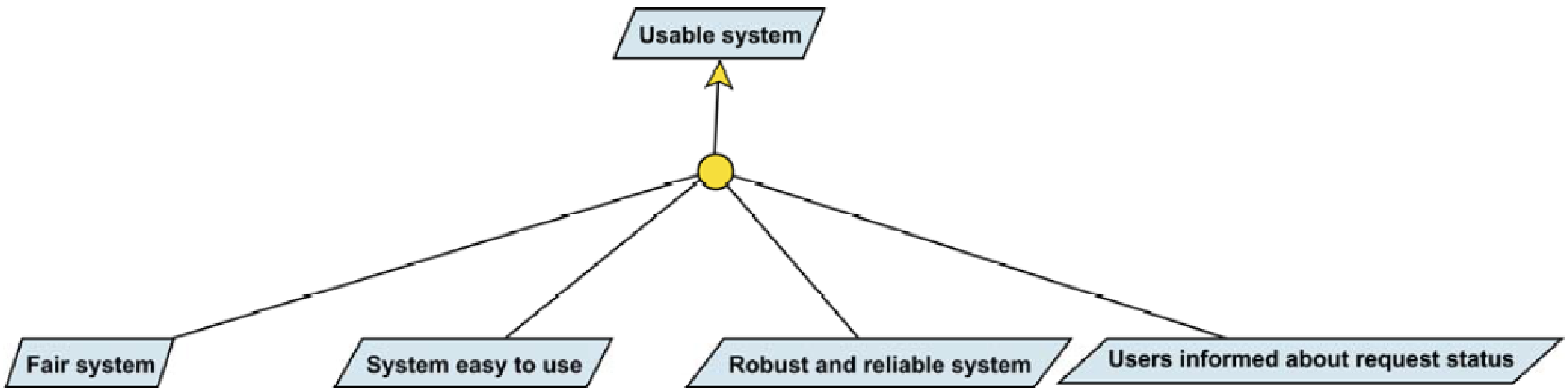
Generic goal patterns



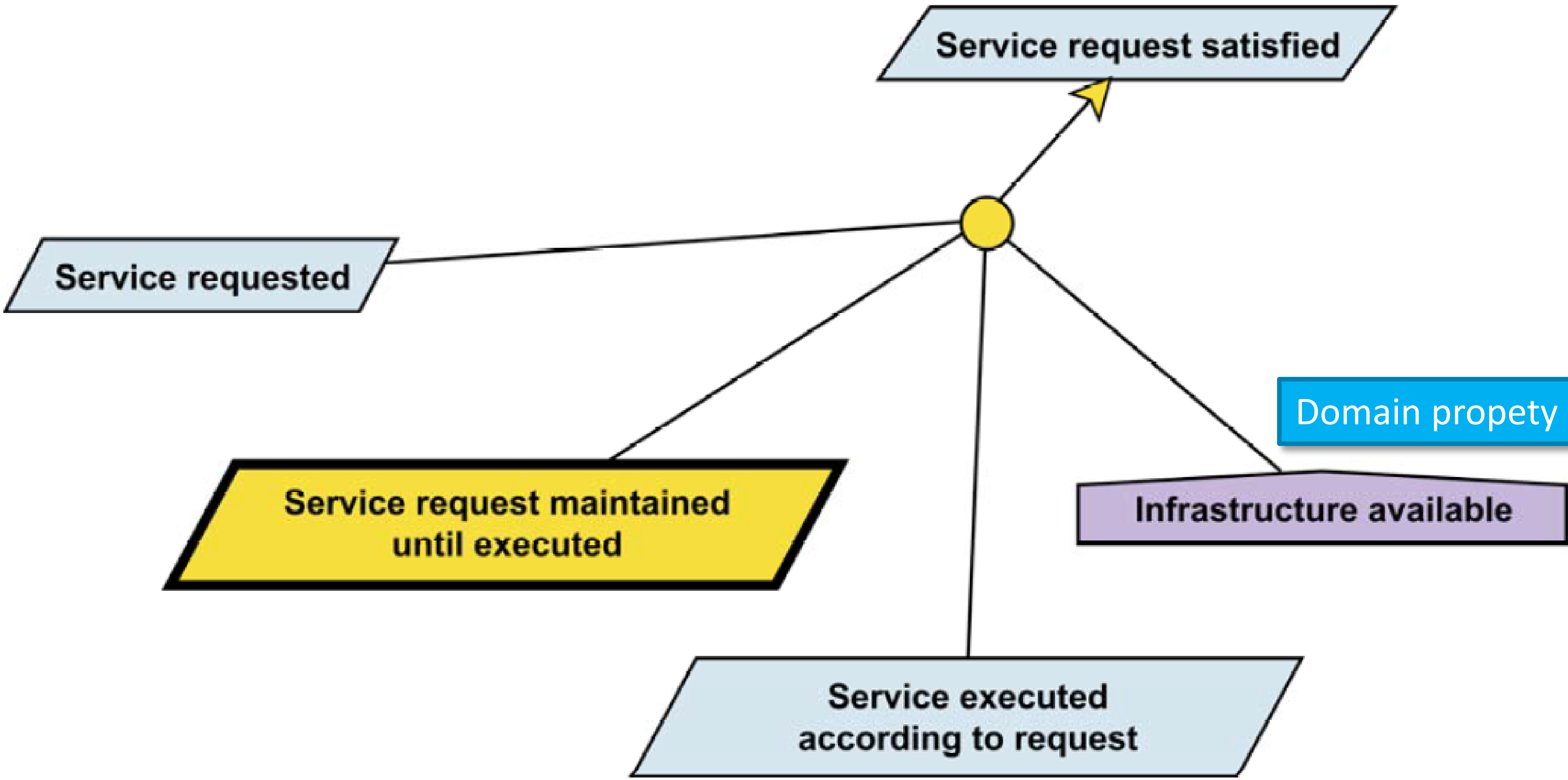
Generic goal patterns



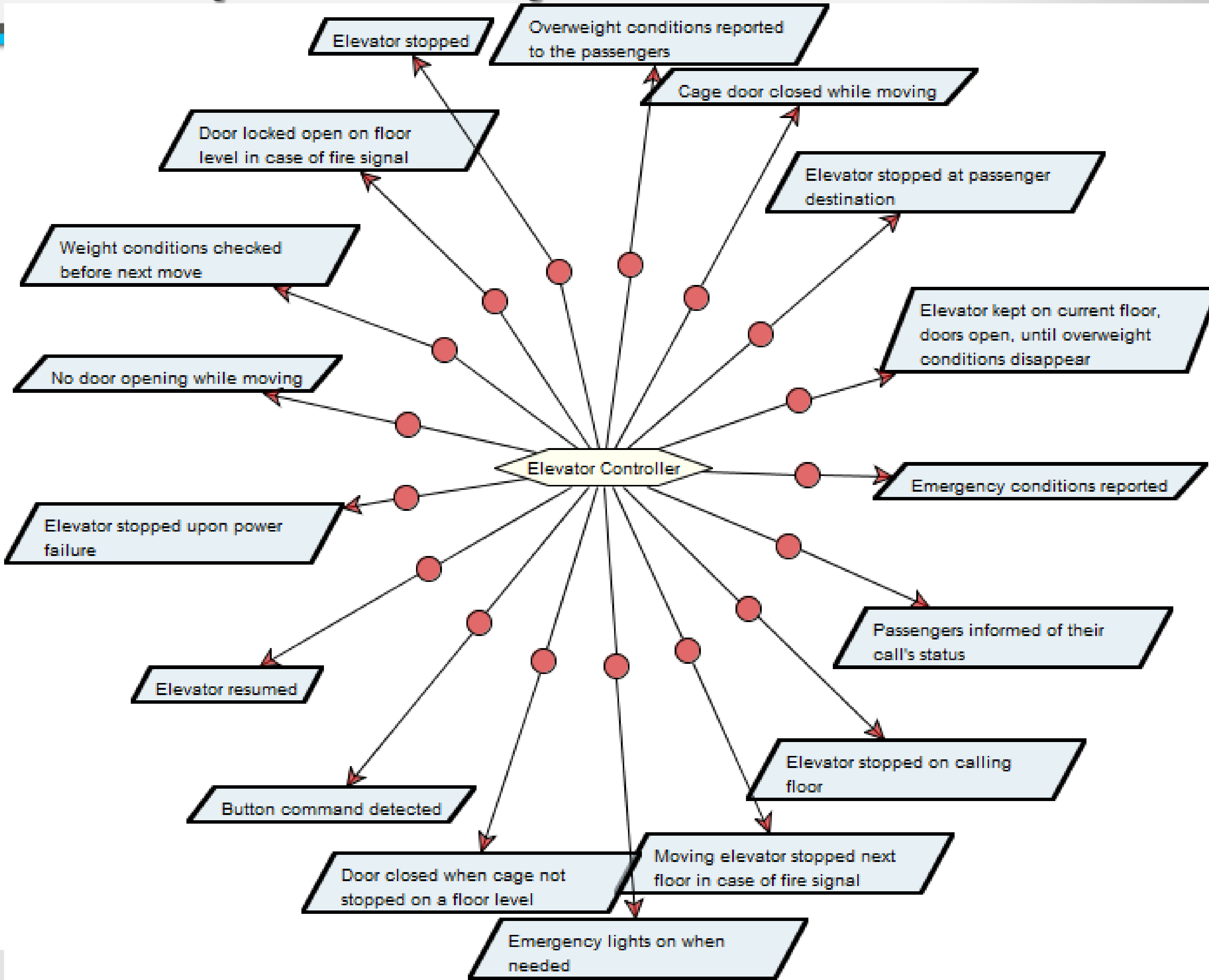
Generic goal patterns



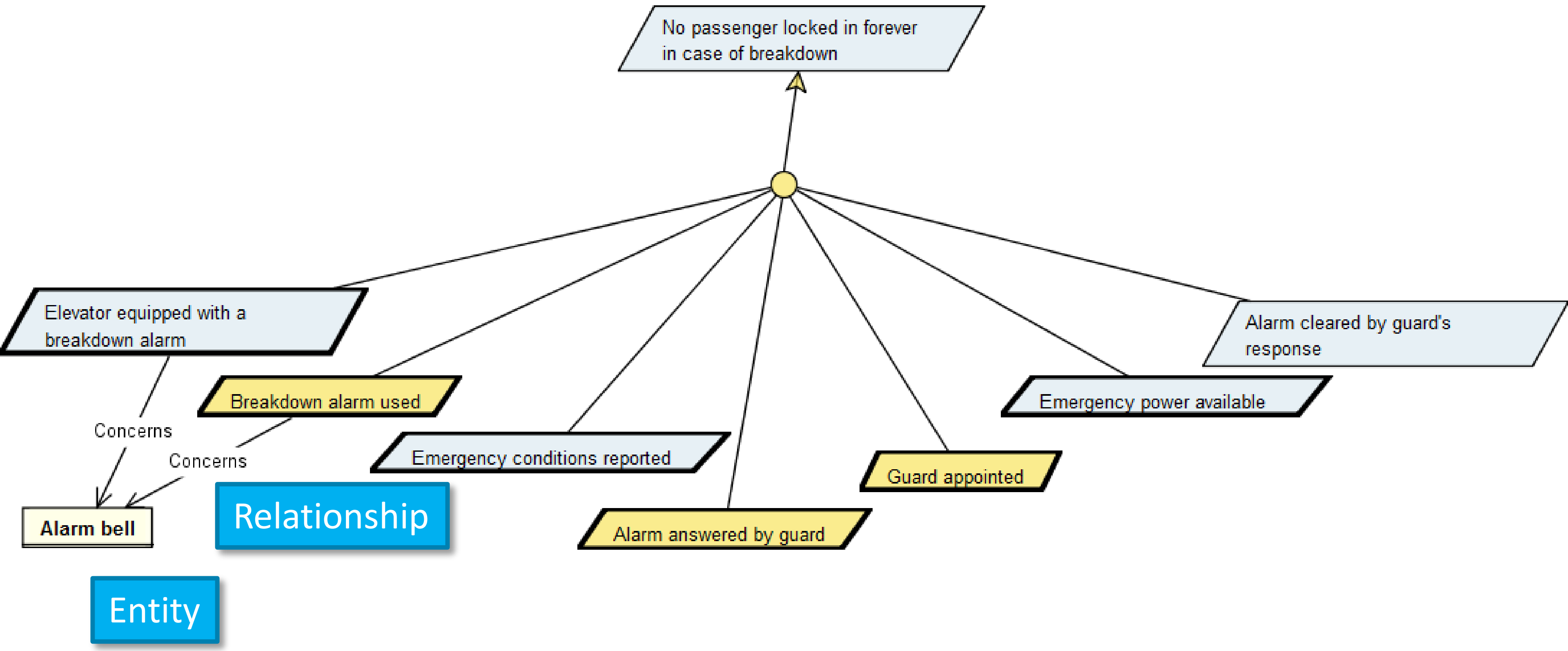
Generic goal patterns



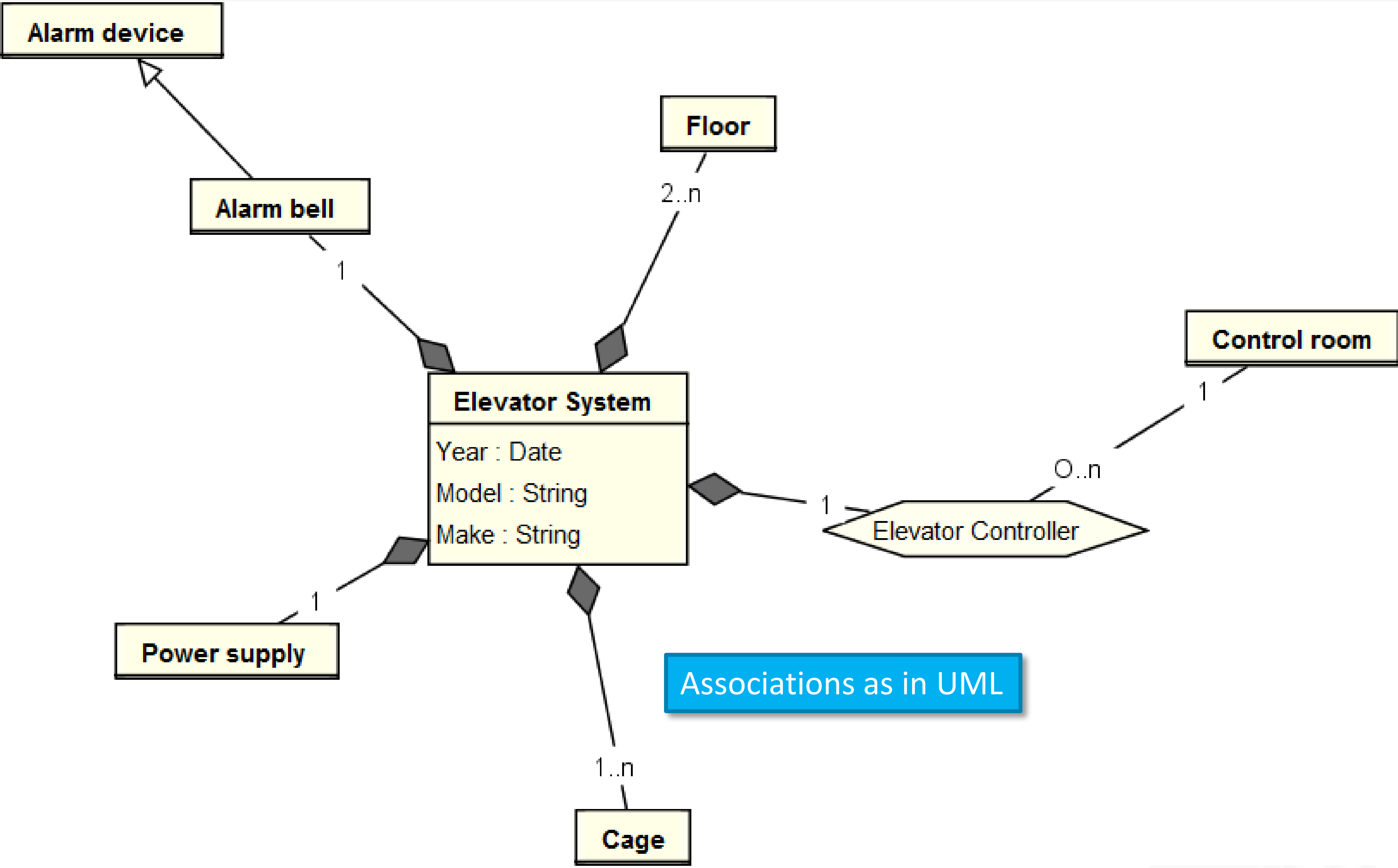
Agent responsibility model



Object model

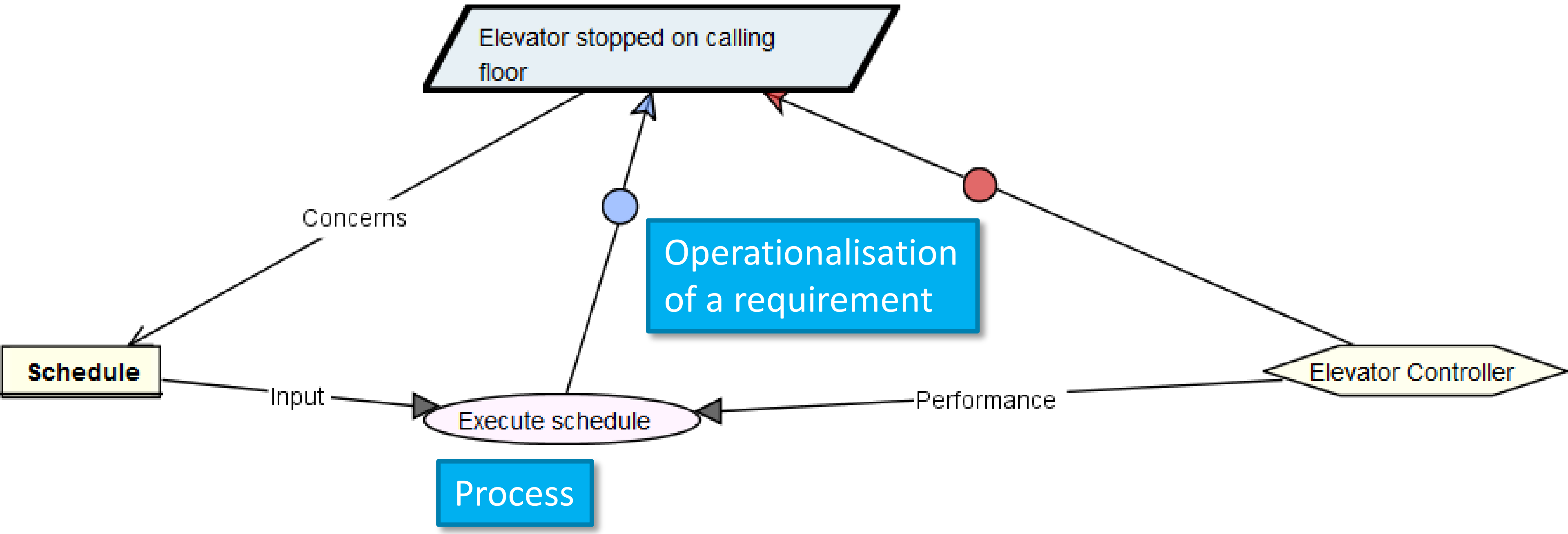


Object model

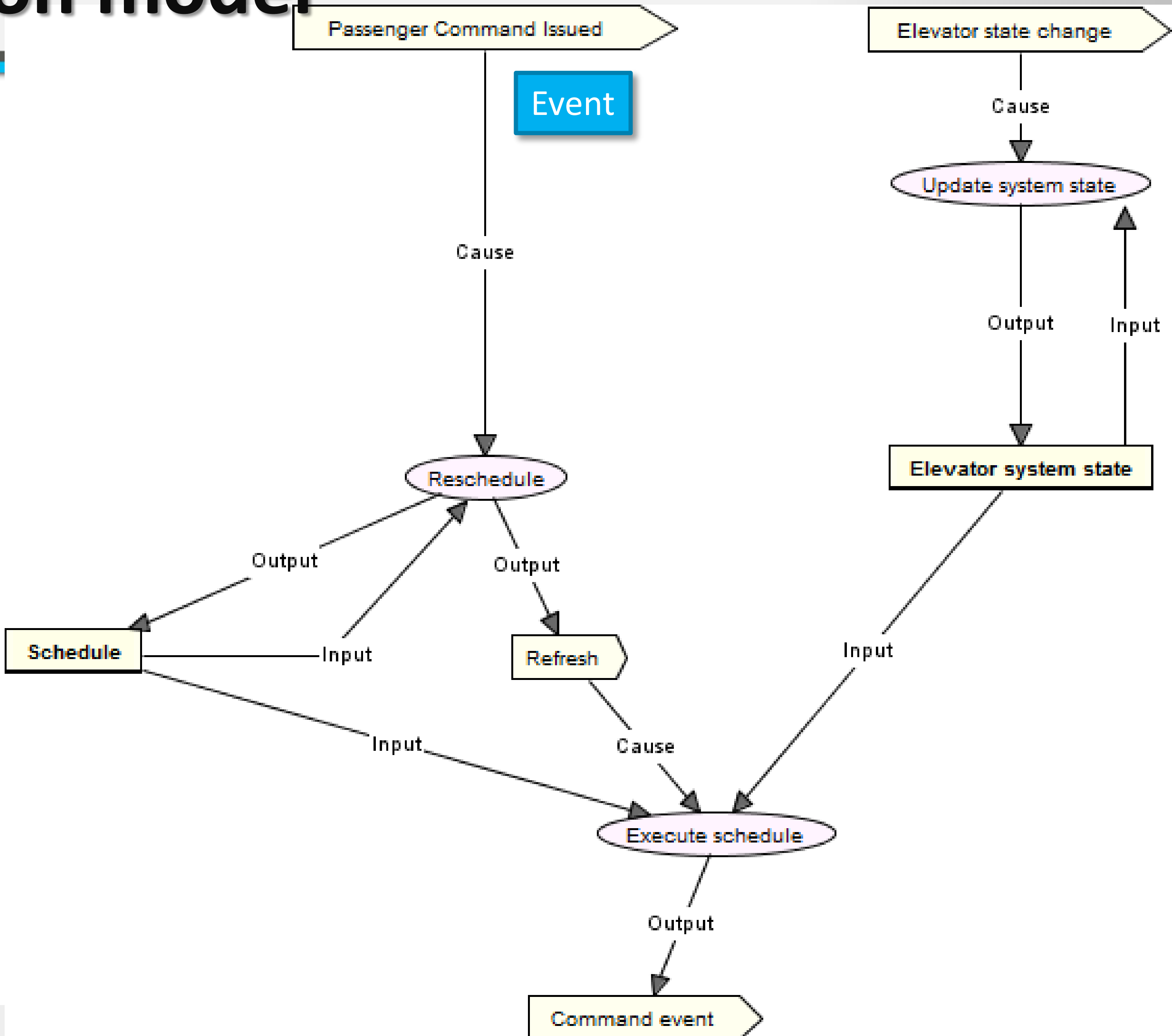


Associations as in UML

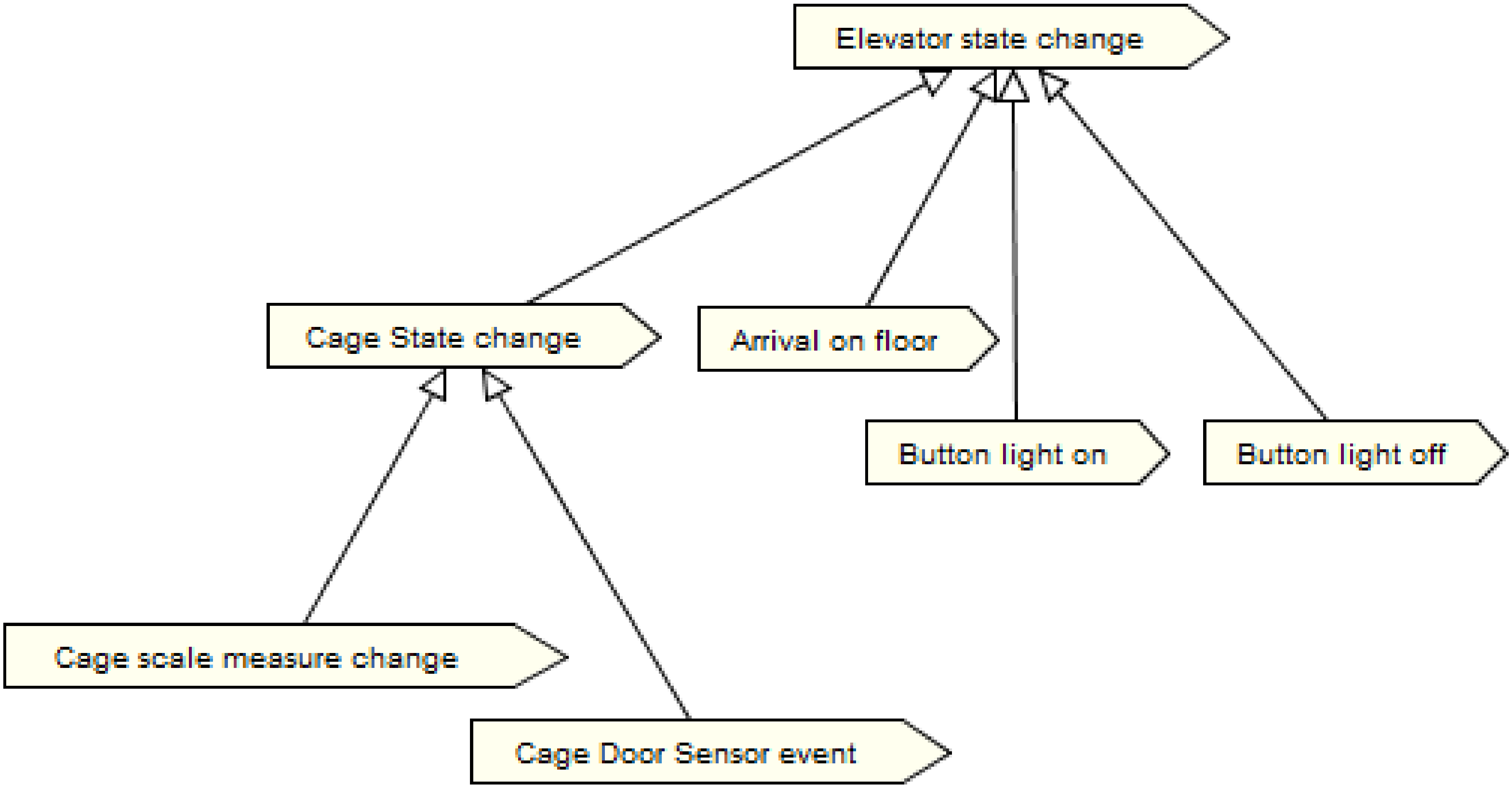
Operation model



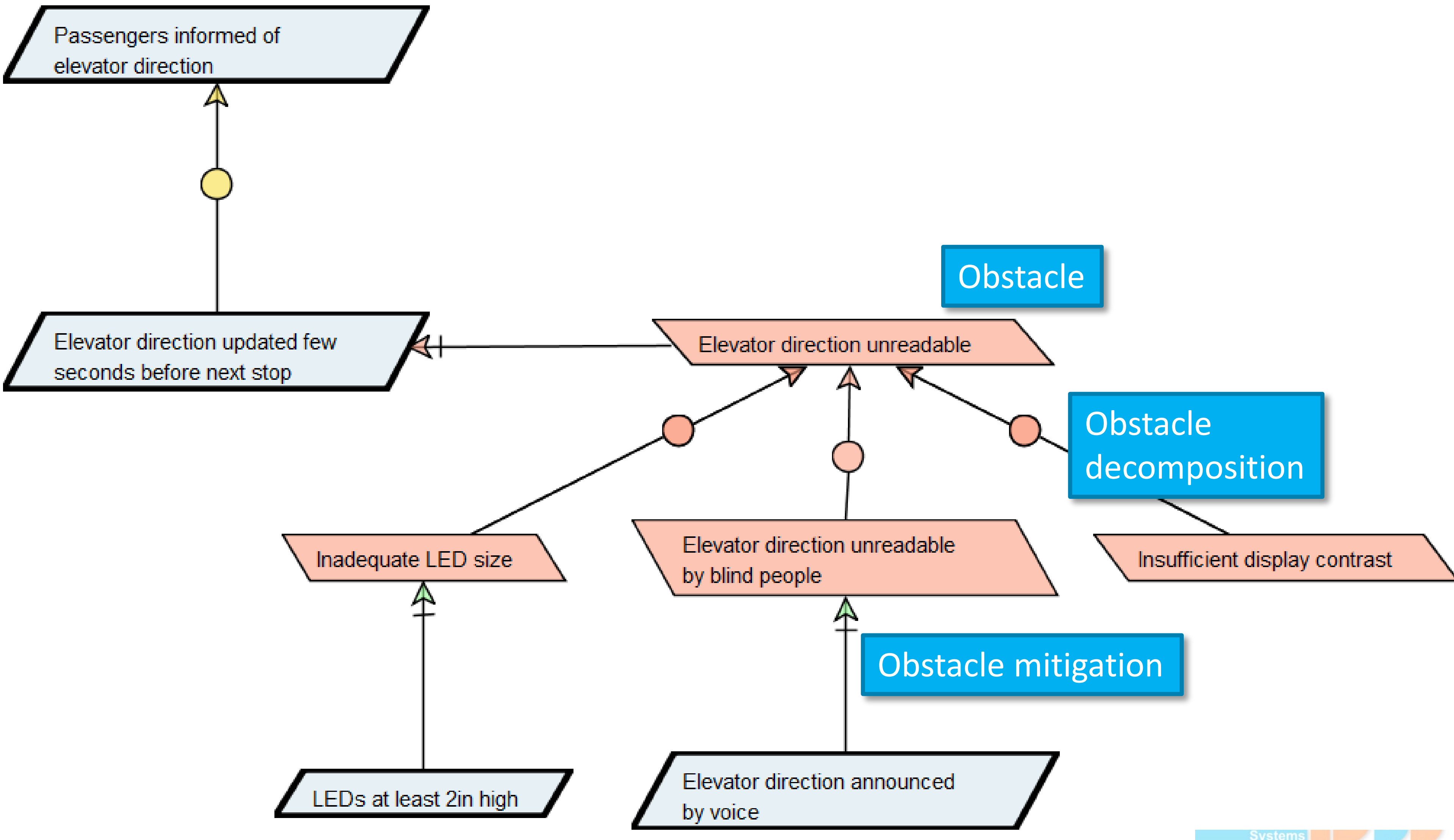
Operation model



Operation model



Obstacle model



Completeness Criteria

- A goal model is said to be complete with respect to the refinement relationship 'if and only if' every leaf goal is either an expectation, a domain property or a requirement.
- A goal model is complete with respect to the responsibility relationship 'if and only if' every requirement is placed under the responsibility of one and only one agent (either explicitly or implicitly if the requirement refines another one which has been placed under the responsibility of some agent).
- To be complete, a process diagram must specify
  - the agents who perform the operations
  - the input and output data for each operation.
- To be complete, a process diagram must specify when operations are to be executed.
- All operations are to be justified by the existence of some requirements (through the use of operationalization links).

Department of Distributed and Dependable Systems

KAOS Notation Summary