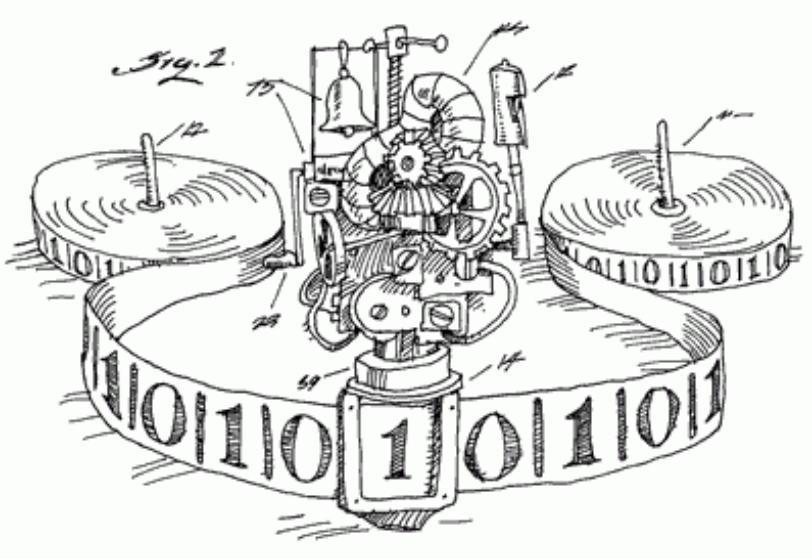
INFO 101 – Introduction to Computing and Security

[2020 - Week 3 / 1]



Prof. Dr. Rui Abreu

University of Porto, Portugal

rui@computer.org

@rmaranhao

Exercise (Assignment 1 material)

- Formalise these statements and determine (with truth tables or otherwise) whether they are consistent (i.e. if there are some assumptions on the atomic propositions that make it true):
  - “The system is in a multiuser state if and only if it is operating normally. If the system is operating normally, the kernel is functioning. Either the kernel is not functioning or the system is in interrupt mode. If the system is not in multiuser state, then it is in interrupt mode. The system is not in interrupt mode.”

https://web.stanford.edu/class/cs103/tools/truth-table-tool/

What we are going to discuss…

- Introducing systems engineering and its relationship to other disciplines
- Introducing system analysis including tasks, processes, and basic concepts
- Brief introduction into modeling for system analysis (and also software analysis)

What is System Engineering?

What are systems?

What are systems?

- *“A system is a set of interacting or interdependent component parts forming a complex/intricate whole.”*

- “*A system is an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective.*”

What are systems?

- IEEE Std 1220-1998: "*A set or arrangement of elements and processes that are related and whose behavior satisfies customer/operational needs and provides for life cycle sustainment of the products.*"
- ISO/IEC 15288:2008: "*A combination of interacting elements organized to achieve one or more stated purposes.*"

What are systems?

- Systems have:
  - Physical and temporal boundaries
  - An environment that is influencing the system
- Systems are described by:
  - Their structure
  - Their objectives
  - Their functionalities

What are systems?

- Systems have a common characteristics:
  - Structure (Components and their interconnections)
  - Behavior
  - Interconnectivity

- Systems themselves can comprise systems (subsystems)

What is System Engineering?

- *System Engineering* is an area of engineering that deals with the design and management of complex systems over their lifespan.

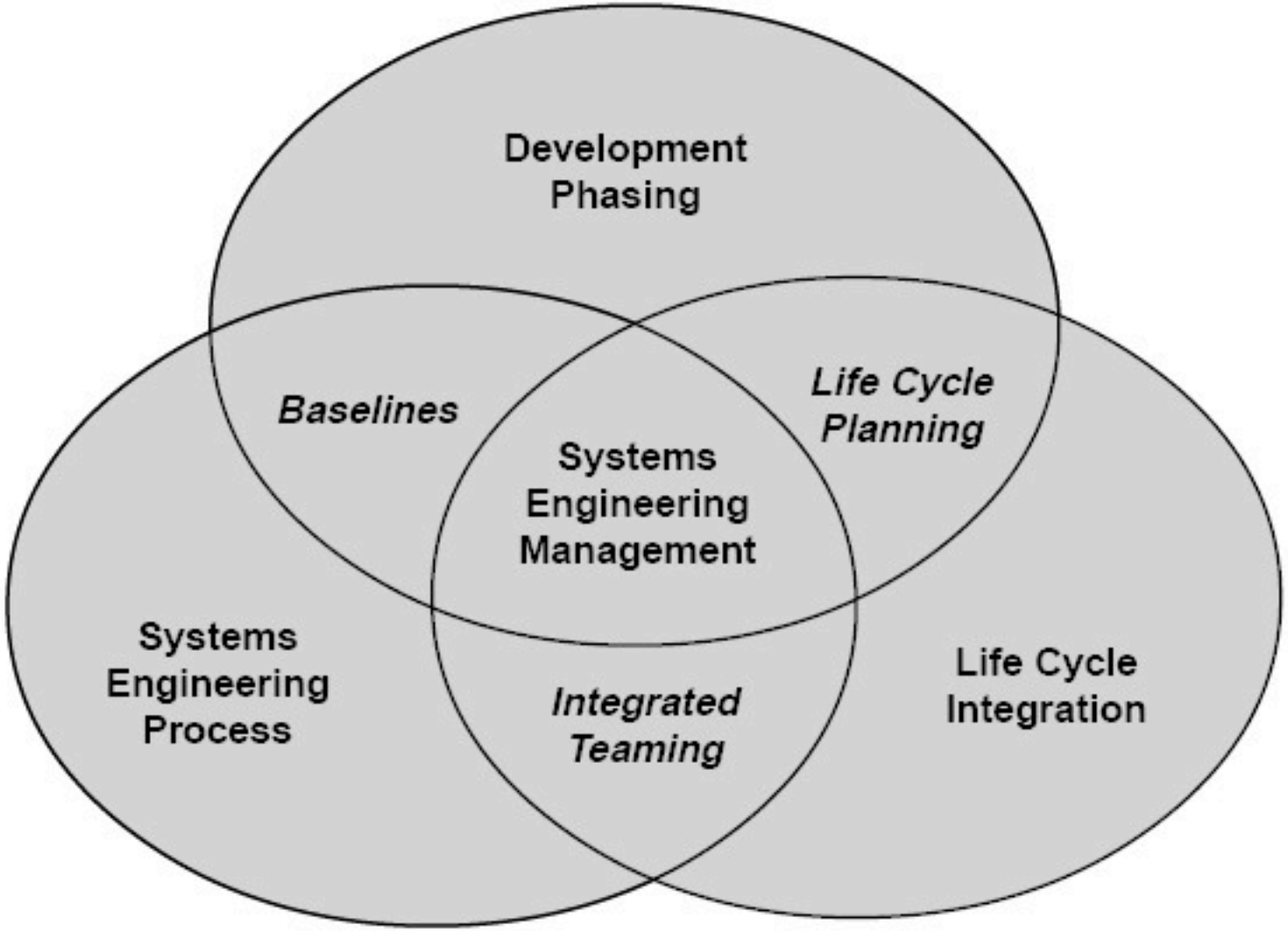
What is System Engineering?

- *"A logical **sequence** of **activities** and **decisions** that **transforms** an operational **need into** a description of system performance **parameters** and a preferred system **configuration**."*
  (MIL-STD-499A, Engineering Management, 1 May 1974.Now cancelled.)

- *"An interdisciplinary **approach** that encompasses the entire technical effort, and evolves into and verifies an integrated and life cycle balance set of **system** **people, products, and process solutions that satisfy customer need**s."*
  (EIA Standard IS-632, Systems Engineering, December 1994.)

- *"An interdisciplinary, collaborative **approach** that **derives, evolves, and verifies** a life-cycle balanced **system** solution which **satisfies customer expectations** and meets **public acceptability**."*
  (IEEE P1220, Standard for Application and Management of the Systems Engineering Process, [Final Draft], 26 September 1994.)

Tasks of System Engineering



Important sub-tasks of System Engineering

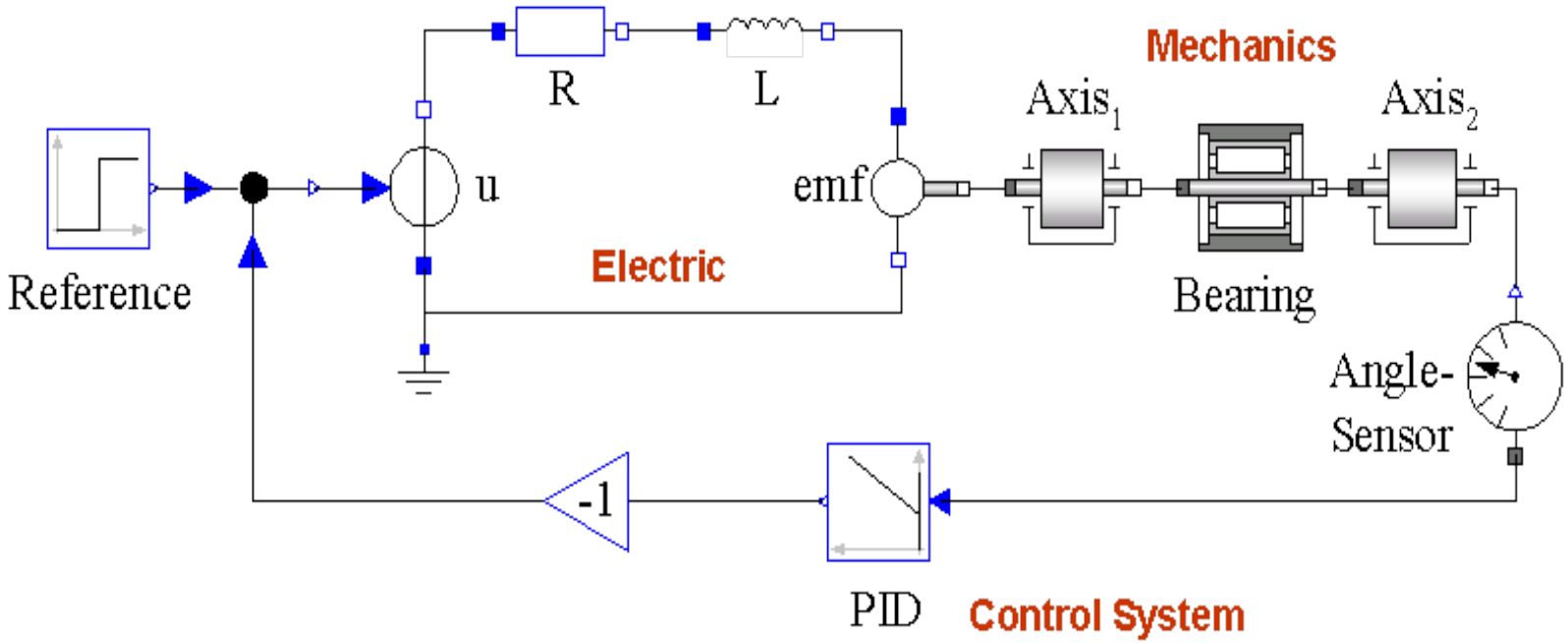
- Use of models and simulation to validate and evaluate system assumptions.
- Use of methods to detect faults as early as possible!

Safety Engineering

- Evaluate and make critical decisions as early as possible (wrt their consequences)

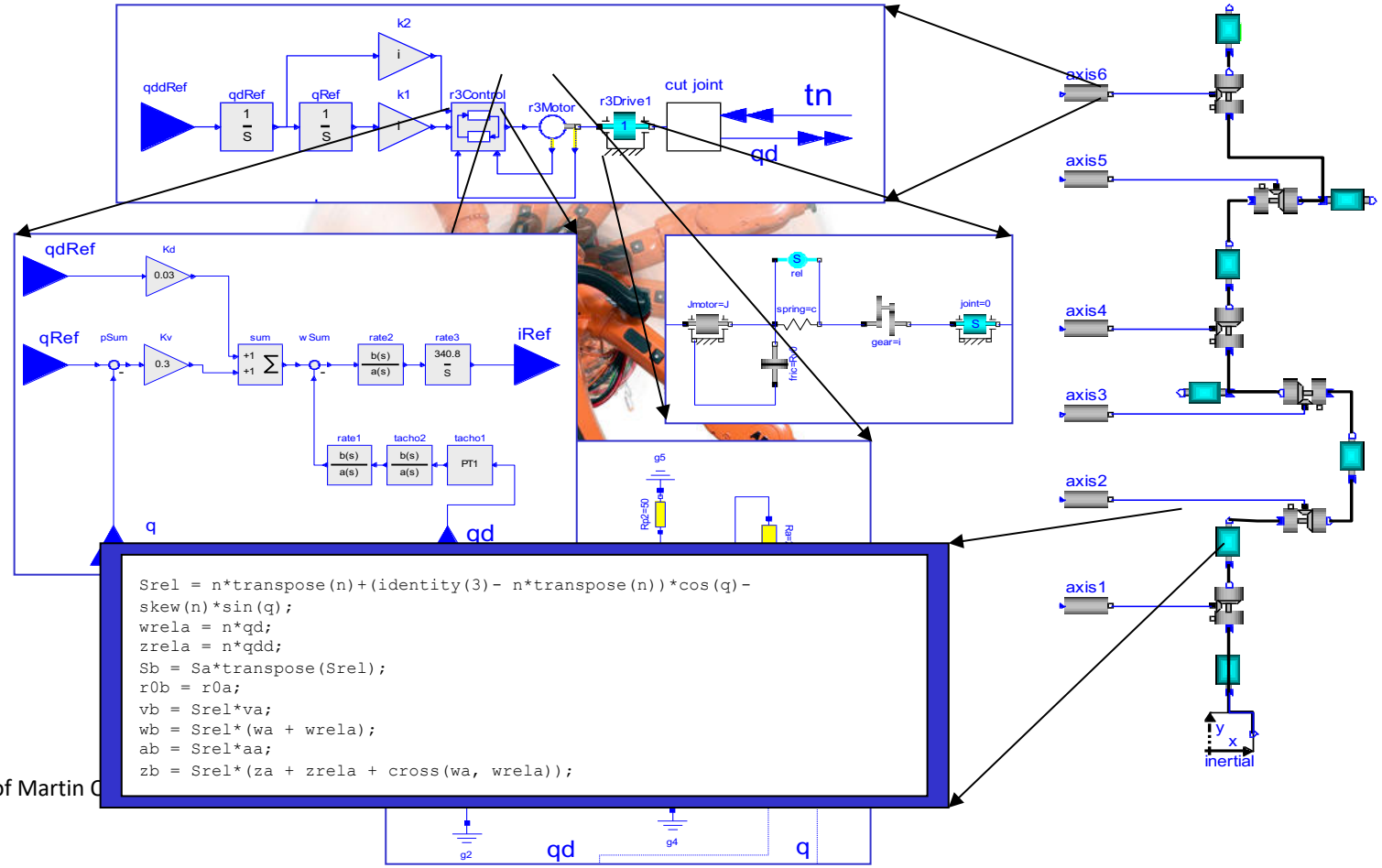
Simulation models

- Multi-domain models



Simulation models

- Hierarchical models



Courtesy of Martin C

Safety Engineering

- **Goal**: To make systems safe (as required)
- Often risk driven analysis
- Risk *r* of a damage *e* is defined as function of the probability *p* that the damage happens and its corresponding costs *c*:

$$r(e) = p(e) * c(e)$$

Safety Engineering

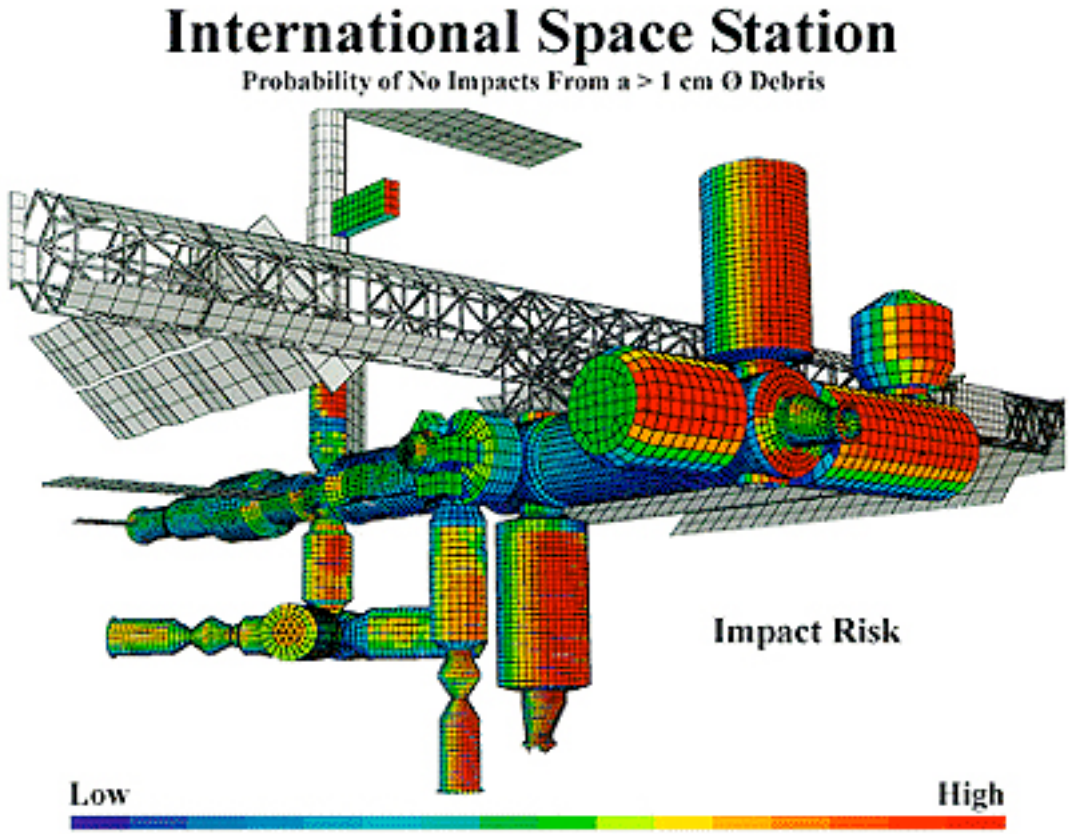
- Objective during development: To eliminate risks or at least to bring them under a certain given threshold.

- Often there is no need to make quantitative predictions of risks. Instead a qualitative risk statement is sufficient!
  - **Example**: If there is a risk of human losses if an event occurs, then its assigned risk is high (e.g. ASIL D in the automotive industry)

Safety Engineering

- Example: Distribution of risk caused by an impact at the Intl. Space Station

  Source: NASA



Methods for estimating risks

- **Failure Mode and Effect Analysis (FMEA)**
  - Bottom Up Approach
  - Two step approach:
    1. Identify all fault modes for each system component.
    2. For each fault mode identify the effects and assign a risk to it.

Methods for estimating risks

- Example FMEA:

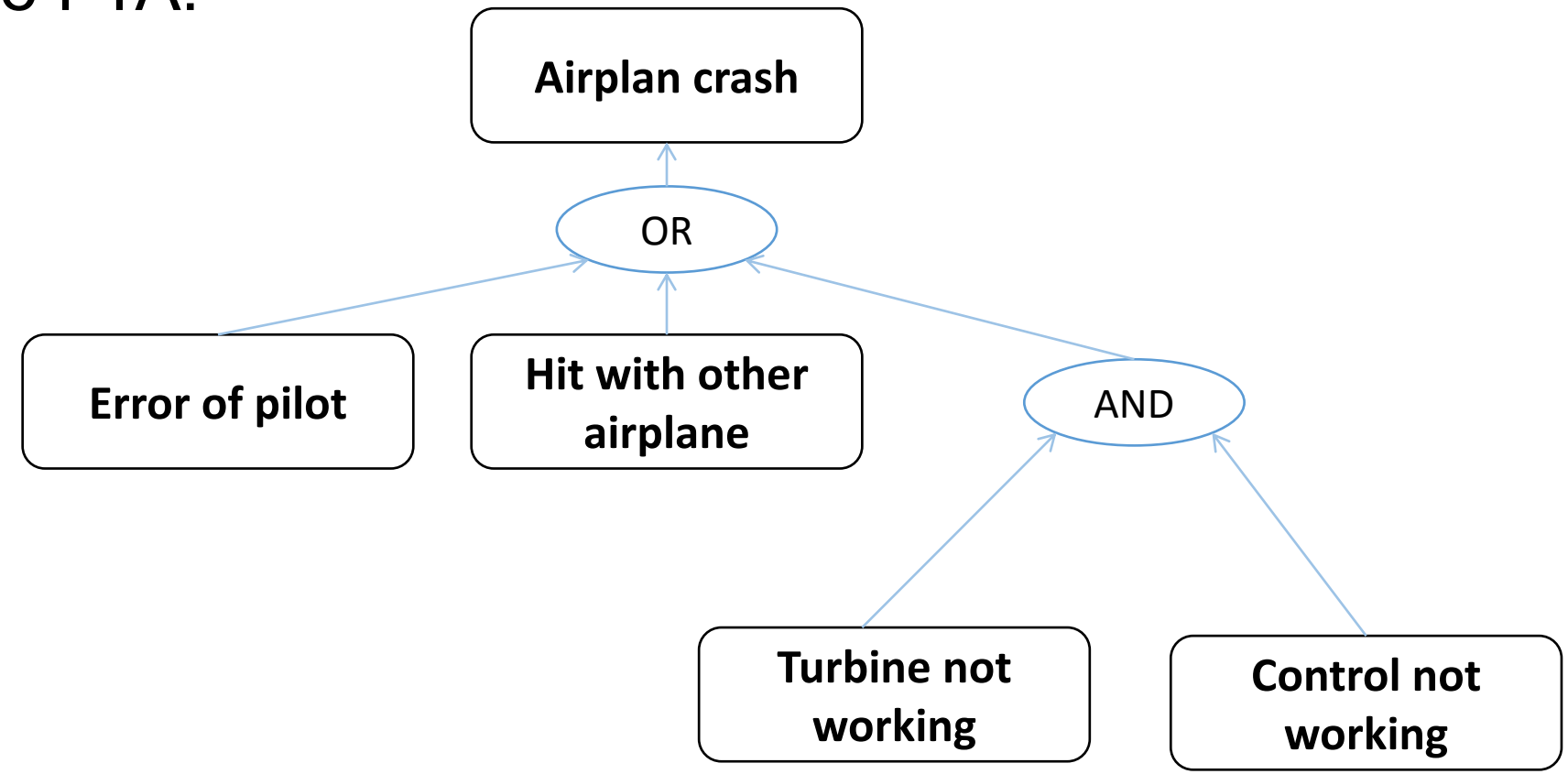
Component	Mode	Effect	Risk
Fuel system	Indicates too much fuel	Goal cannot be reached with available fuel	Medium to high
Jet engine	Not working	At least reduced redundancy and change in flight behavior	Medium
Jet engine	Burning	At least reduced redundancy and change in flight behavior	Medium

Methods for estimating risks

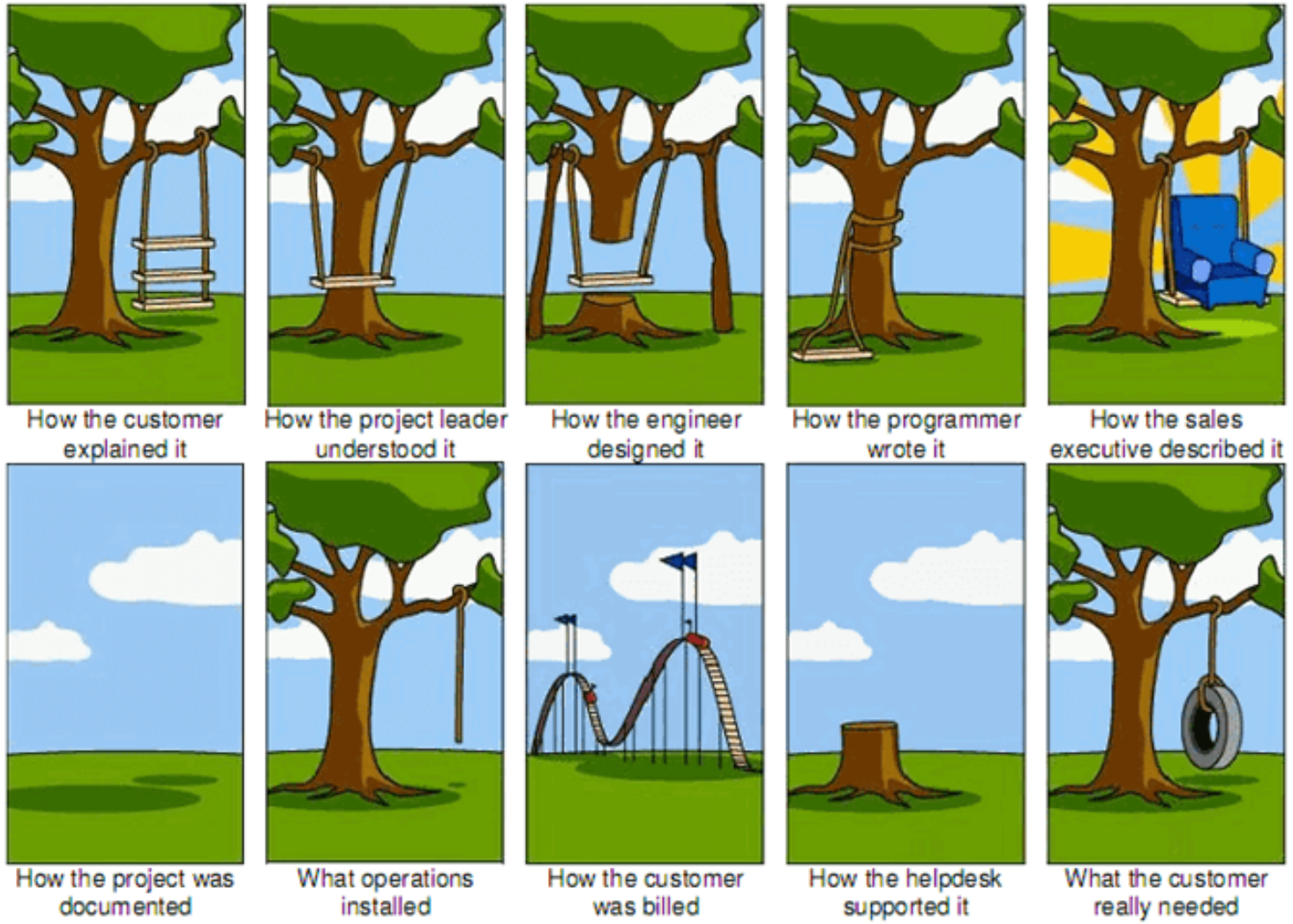
- Fault Tree Analysis (FTA)
  - Top Down Approach
  - Definition of top events like airplane crashes or car breaks not working anymore.
  - Assign primary events like component faults, human error, or external events using Boolean logic to the top event.

Methods for estimating risks

- Example FTA:



The main challenge of software and systems engineering



Consequences

- Need for a process that allows avoiding trouble
- Need to identify stakeholders
- Communication with stakeholders is important
  - Obtain requirements
  - Discuss potential solutions

Example process – SYSMOD

- Underlying idea:
  - Consider systems as black boxes
  - Do the following:
    - Identify elements
    - Describe the context
    - Detail the inner part of the system
- Tasks:
  - Describe the idea and the objectives behind a system
  - Outline the base architecture of the system
  - Identify the requirements
  - Model the system context, i.e., its environment
  - Model use cases
  - Model expert knowledge

Some remarks

- A project is itself always unique, has a start and end date, an associated organization (project leader & participants, stakeholders,…), resources.
- Start of a project is an IDEA!
  - It is important to describe this idea
  - Identify necessary innovations
  - Some innovations may not be easy to reach
  - Requirements may prevent innovations
  - Requirements should never describe how to solve the idea, i.e., build the system.
  - Outline different potential solutions, and compare them!

Documents to be written during a project in systems engineering

- Requirements document
  - Describes and structures the requirements coming from the stakeholders
- Use cases
  - Describe how users and other systems interact with the system to be developed
  - Important to identify the system's boundaries and its environment
- System architecture
  - Starting with a coarse structure and refine it
- System models
  - Used to further understand the system and for simulation purposes (feasibility checks)

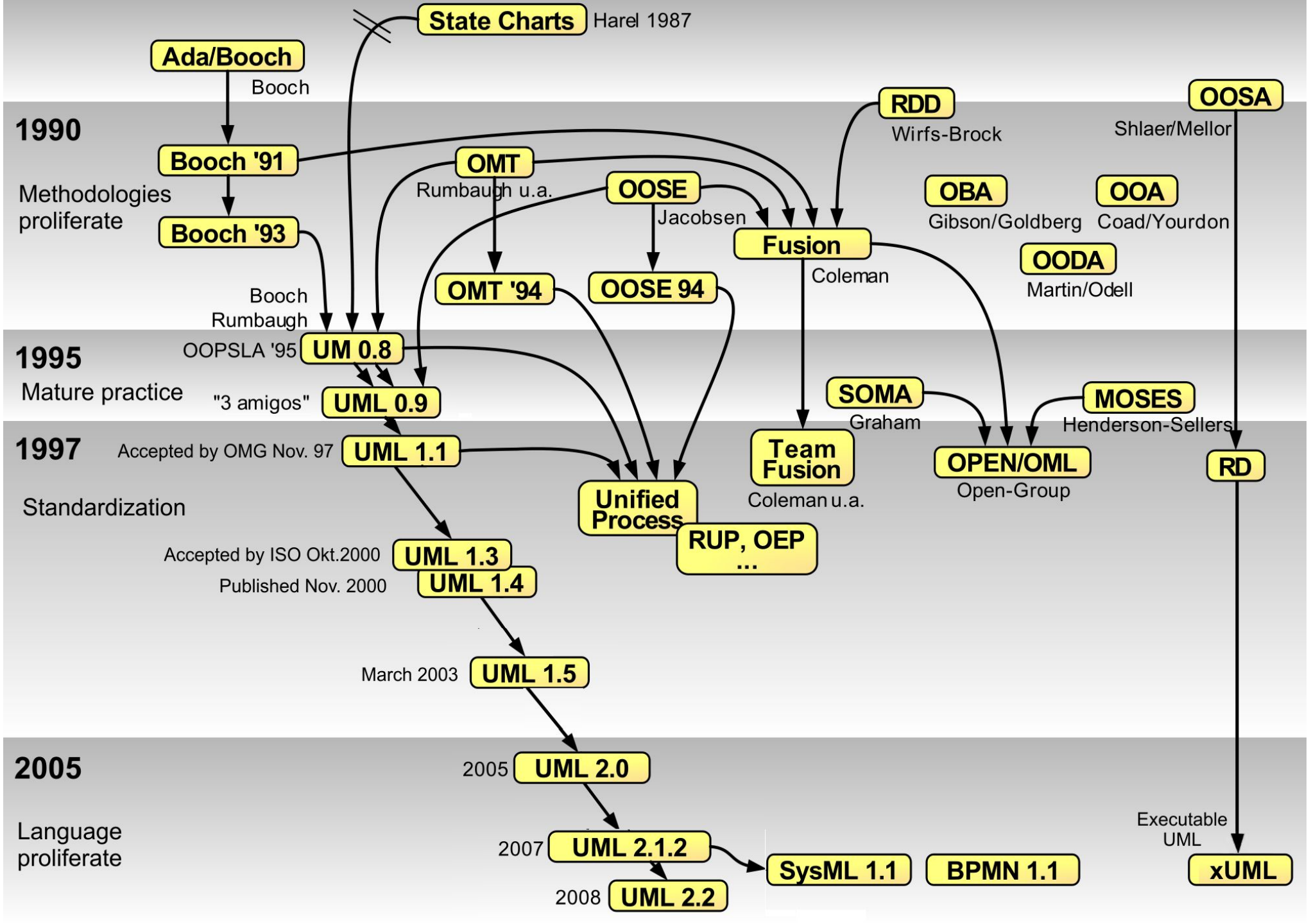
Other project documents

- All other development documents (description of hardware & software, user documentation, test documentation,…)
- Project meeting reports
  - Formal document
  - Comprises:
    - A title
    - Place and data of the meeting
    - Participants
    - Description of things discussed as well as their results
  - Part of the official project documentation!
- Project diary
  - Collection of private notes

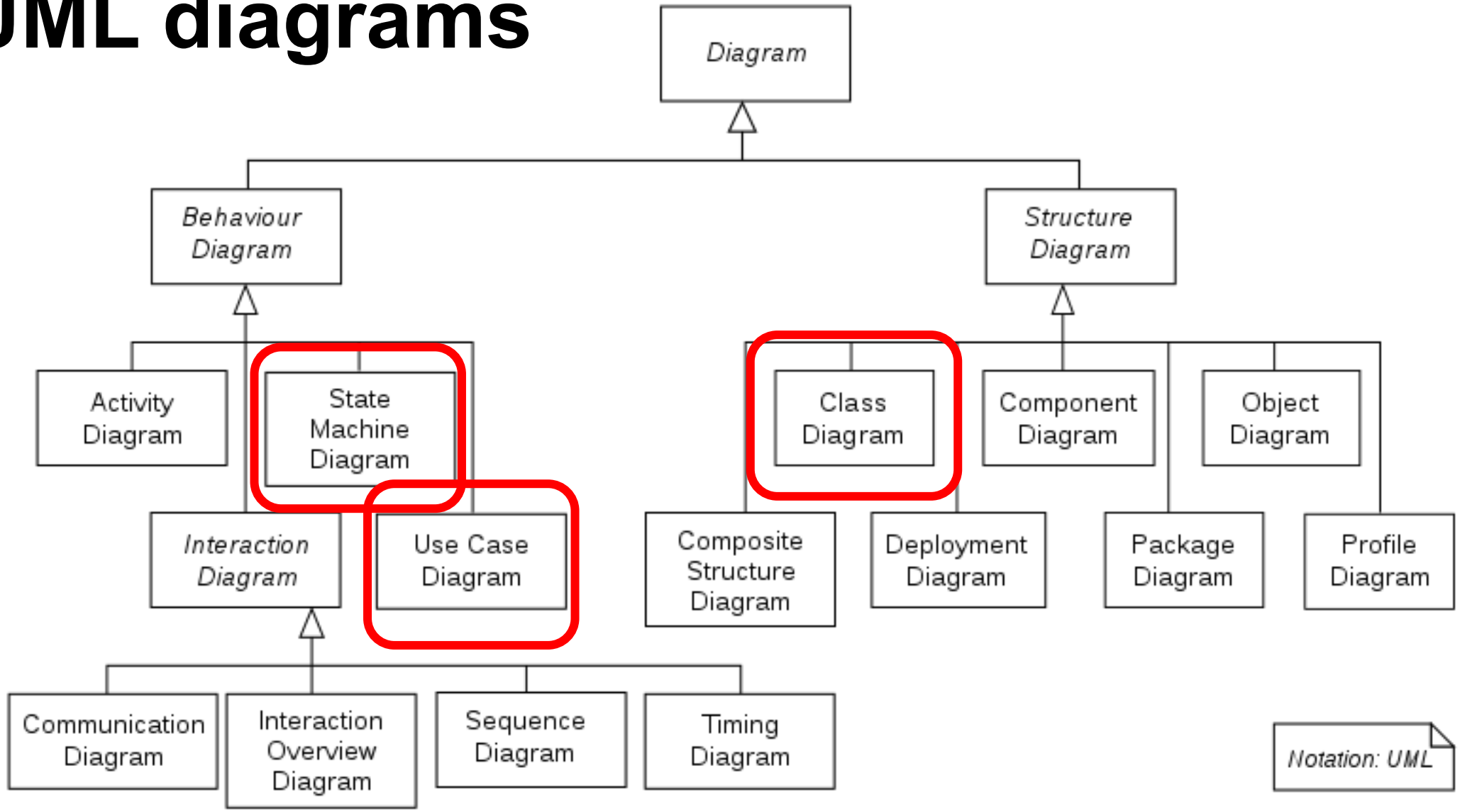
Modeling for system analysis

The Unified Modeling Language (UML)

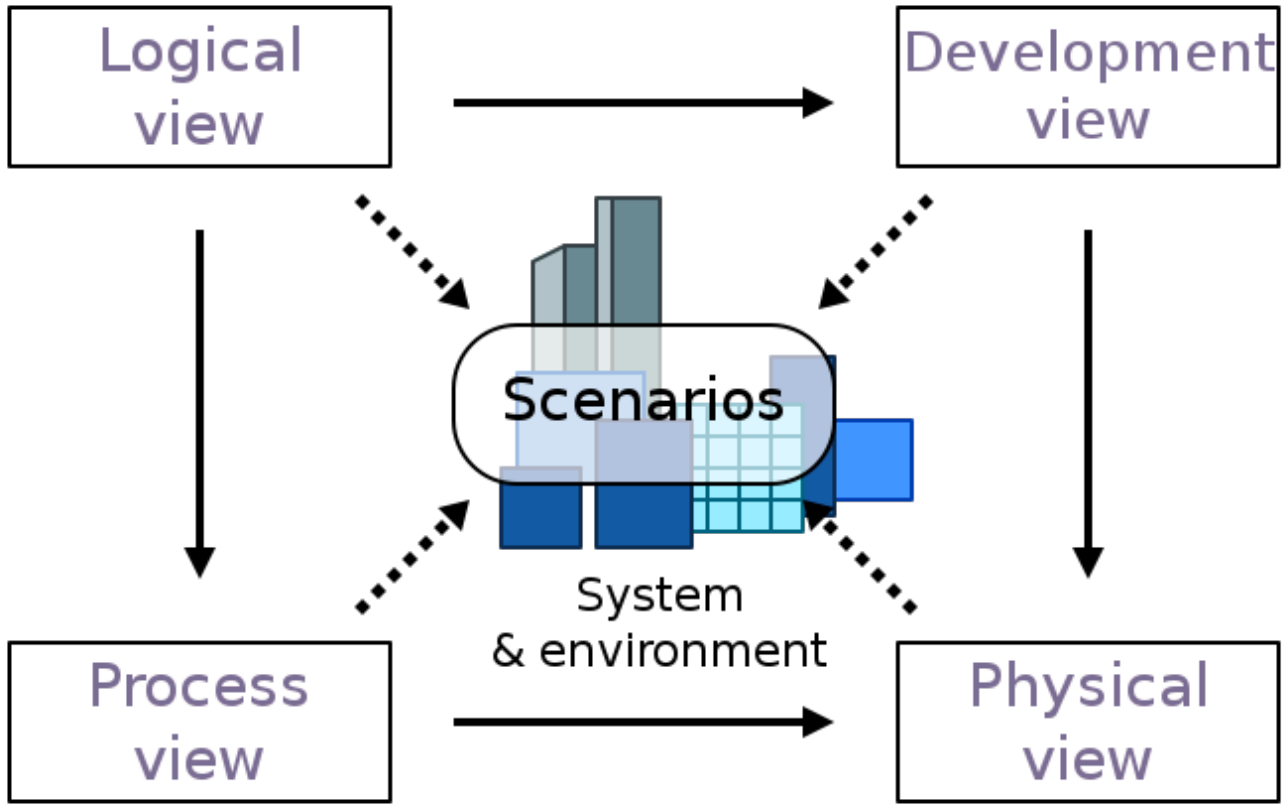
- A collection of diagrams for supporting software (and systems) development
- Each diagram has a syntax and a semantics
- To be used to outline systems and their components
- Different views on the system supported by different diagrams
- Reduction of complexity via deconstructing a system into its parts and using refinement
- Supporting communication with stakeholders via use case diagrams


State Charts
Harel 1987
Ada/Booch
Booch
1990
Booch '91
Booch '93
Methodologies
proliferate
OMT
Rumbaugh u.a.
OOSE
Jacobsen
RDD
Wirfs-Brock
OOSA
Shlaer/Mellor
OBA
Gibson/Goldberg
OOA
Coad/Yourdon
Fusion
Coleman
OODA
Martin/Odell
OMT '94
OOSE 94
Booch
Rumbaugh
1995
OOPSLA '95
UM 0.8
Mature practice
"3 amigos"
UML 0.9
SOMA
Graham
MOSES
Henderson-Sellers
1997
Accepted by OMG Nov. 97
UML 1.1
Team
Fusion
Coleman u.a.
OPEN/OML
Open-Group
RD
Unified
Process
RUP, OEP
...
Standardization
Accepted by ISO Okt.2000
UML 1.3
Published Nov. 2000
UML 1.4
March 2003
UML 1.5
2005
2005
UML 2.0
Language
proliferate
2007
UML 2.1.2
SysML 1.1
BPMN 1.1
Executable
UML
xUML
2008
UML 2.2

UML diagrams



Home work: check out the 4+1 Architectural Diagrams



Use Cases

- An (itemized) story of possible use
- Created from a user interview
  - Tied to a user
- Use diagrams and text

- Use case prioritization

Use Cases Capture Functional Requirements

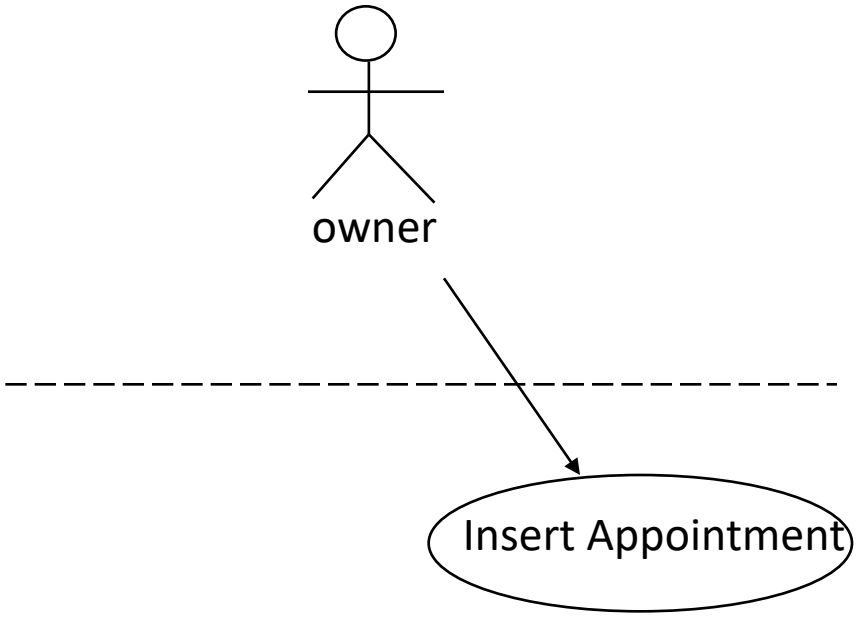
- Not captured:
  - Performance,
  - Reliability,
  - Hardware compatibility,
  - …
- Need to capture non-functional requirements separately

What to do

- Three tasks:
  1. Find system boundary
  2. Find actors
     - Actors are Roles
  3. Find use Cases

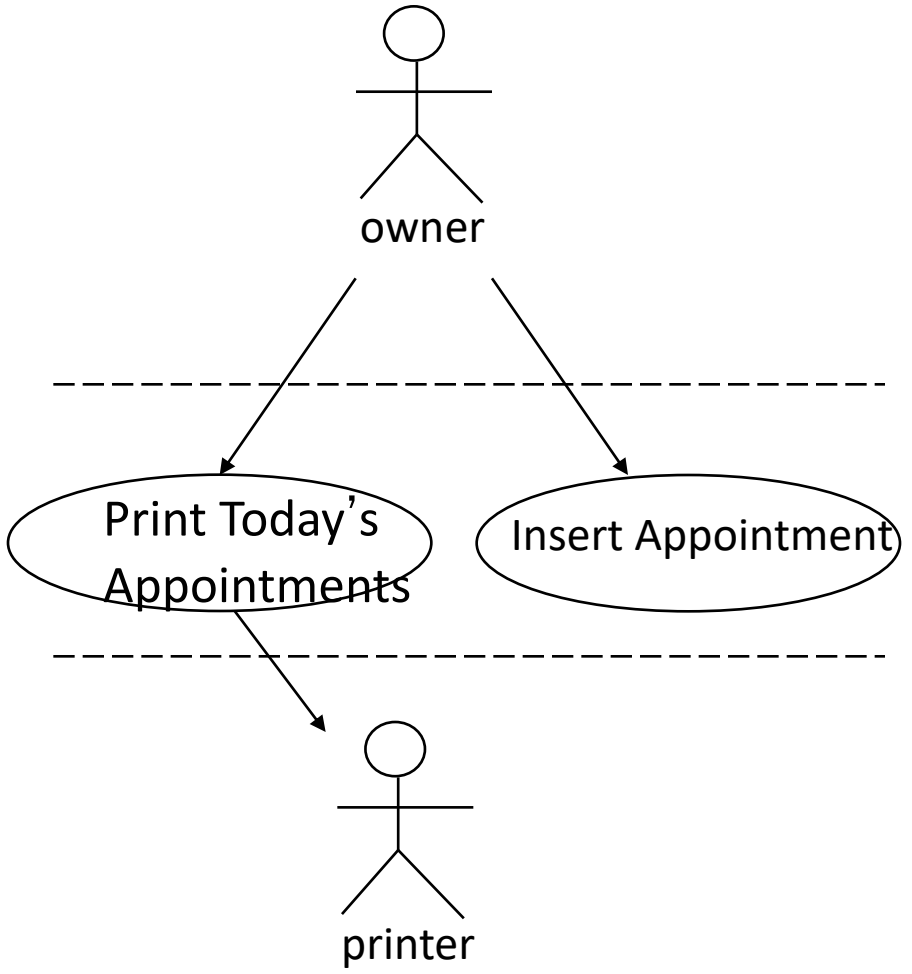
Use Case Diagrams

- Actors
  - Are outside the system
  - Interact with it
  - Are *roles*, not people
  - E.g.: calendar-owner, secretary, MS-Outlook, Time
- Use Cases
  - Describe one use of the system
- Relationships
- System Boundary
  - Actors are outside, use cases inside



Use Case Diagrams

- Actors
  - Are outside the system
  - Interact with it
  - Are *roles*, not people
  - E.g.: calendar-owner, secretary, MS-Outlook, Time
- Use Cases
  - Describe one use of the system
- Relationships
- System Boundary
  - Actors are outside, use cases inside



Identifying Actors

- Who or what uses the system?
- What roles do they play in interaction?
- Who maintains the system?
- What other systems interact with it?
- Who gets and provides information?
- Do things happen at fixed times?

Use Case

- For every role, describe typical interactions with the system
- We want to understand the requirements, not model the system in detail.

Example

UseCase: Insert Appointment

ID: UC1.1

Actors: owner

Preconditions:

Flow of events:

1. The user specifies date, beginning time, end time, and description of appointment
2. The calendar warns about conflicting appointments
3. The appointment is entered in the calendar

Postconditions: The appointment is in the calendar

Requirements

- The uses are
  - Clear
  - Detailed
- Is anything missing? (Who, what, when, where?)
- Bad use case: The user enters the appointment data.
- Use cases describe one example!
- Detail depends on phase

Example: a Later Iteration

UseCase: Insert Appointment

ID: UC1.2

Actors: owner

Preconditions:

Flow of events:

1. The owner selects the day of the appointment
2. The system shows the day, including all appointments already scheduled
3. The user clicks on the begin time, drags to end time, and releases the mouse button
4. The system shows a box for the appointment, and a cursor inside the box. If there are conflicting appointments, the box is in s different color.
5. The user types a description.
6. The appointment is entered in the data base

Postconditions: The appointment is in the calendar

More Complicated Use Cases

- Use case describes one example of use!

- But: what to do with very similar cases?

Extra Keywords

- If, for, while
  - If there is a conflict, the system warns
  - For all appointments in the day, the calendar shows a box
- Alternative flows

Example: Alternative Flows

UseCase: Insert Appointment

ID: UC1.1.1

Actors: owner

Preconditions:

Flow of events:

1. The user specifies date, beginning time, end time, and description of appointment
2. If there is a conflicting appointment
    1. The calendar issues a warning
3. The appointment is entered in the calendar

Postconditions: The appointment is in the calendar

Alternative Flows: The user may cancel the creation of an appointment at any time before entering the description

Postconditions: The calendar is not changed

Alternative Flows: The user may leave the description empty

Postconditions: The calendar enters the description "appointment."

Scenarios

- A path through a use case (without branches, ifs, fors, and whiles), is called a *scenario.*

- Try to stick with scenarios as much as possible, especially initially.

- When are use cases good?

Benefits and Drawbacks

- Use cases are good when
  - There are many actors
  - There are many functional requirements
  - The functional requirements are not immediately clear

- Not as good for e.g.
  - A very fast sorting algorithm (too much)
  - A highly secure system (too little)

An example: ATM