



Elements of System Dynamics

Antoine B. Rauzy

PART 4. SYSTEMIC DIGITAL TWINS

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

Key notions:

- Systems and their Environment
- Needs, Missions and Requirements
- Life-Cycle and Operation Modes
- Use Cases
- Functional and Physical Architectures, Capabilities
- SysML and Architecture Frameworks
- Syntax, Semantics, Pragmatics (Diagrams versus Models)

Rational

Modern engineered systems—whether aircraft, satellites, digital infrastructure, or autonomous vehicles—are becoming increasingly **complex**. They involve numerous **interacting components** across hardware, software, data, and human operations. Traditional **document-based** approaches struggle to keep pace: requirements get lost, **inconsistencies** emerge late, and collaboration across disciplines becomes fragmented (siloed).

Model-Based Systems Engineering (MBSE) addresses this challenge by placing **models**, not documents, at the center of the engineering effort. These models provide a shared, formal, and executable representation of the system across its entire lifecycle—from concept to design, integration, validation, and operations.

Purpose

The purpose of MBSE is twofold:

- Improve System Understanding and Communication
Models enable stakeholders to visualize structure, behavior, and constraints, **reducing ambiguity** and improving design decisions early, when changes cost the least.
- Ensure Rigor, Traceability, and Manageability
A **single source of truth** allows for automated consistency checks, impact analysis, and the seamless propagation of changes across disciplines and lifecycle phases.

Ultimately, MBSE is not just about replacing documents with diagrams—it is about engineering systems more reliably, collaboratively, and efficiently. It aims at strengthening the ability to master complexity, reduce risks, and deliver systems that meet their intended purpose in an increasingly interconnected world.

Objectives

MBSE aims at answering the following questions (among others):

- What does justify the design and operation of the system?
- Who needs the services provided by the system?
- What are the needs of the system with respect to external systems in order to provide these services?
- What are the functions of the system?
- How are these functions implemented? By which physical or organizational parts?
- How to ensure that the system will provide services with the expected level of performance?
- How to optimize the system?
- Is the system safe and reliable to be operated?
- ...

Answering these questions is typically the purpose of a **systems architecture process**.

Agenda

- Section 1. The System and its Environment
- Section 2. Needs, Missions and Requirements
- Section 3. Life-Cycle and Operation Modes
- Section 4. Use Cases
- Section 5. Functional and Physical Architectures
- Section 6. SysML and Architecture Frameworks
- Section 7. Diagrams or Models?
- Section 8. Wrap-Up

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 1. THE SYSTEM AND ITS ENVIRONMENT

System's Boundary and Interface Analysis

A **system** does not exist in isolation. It is always embedded in a larger context — an ecosystem of other systems, actors, infrastructures, regulations, and physical conditions.

Determining the boundary of a system means clearly defining:

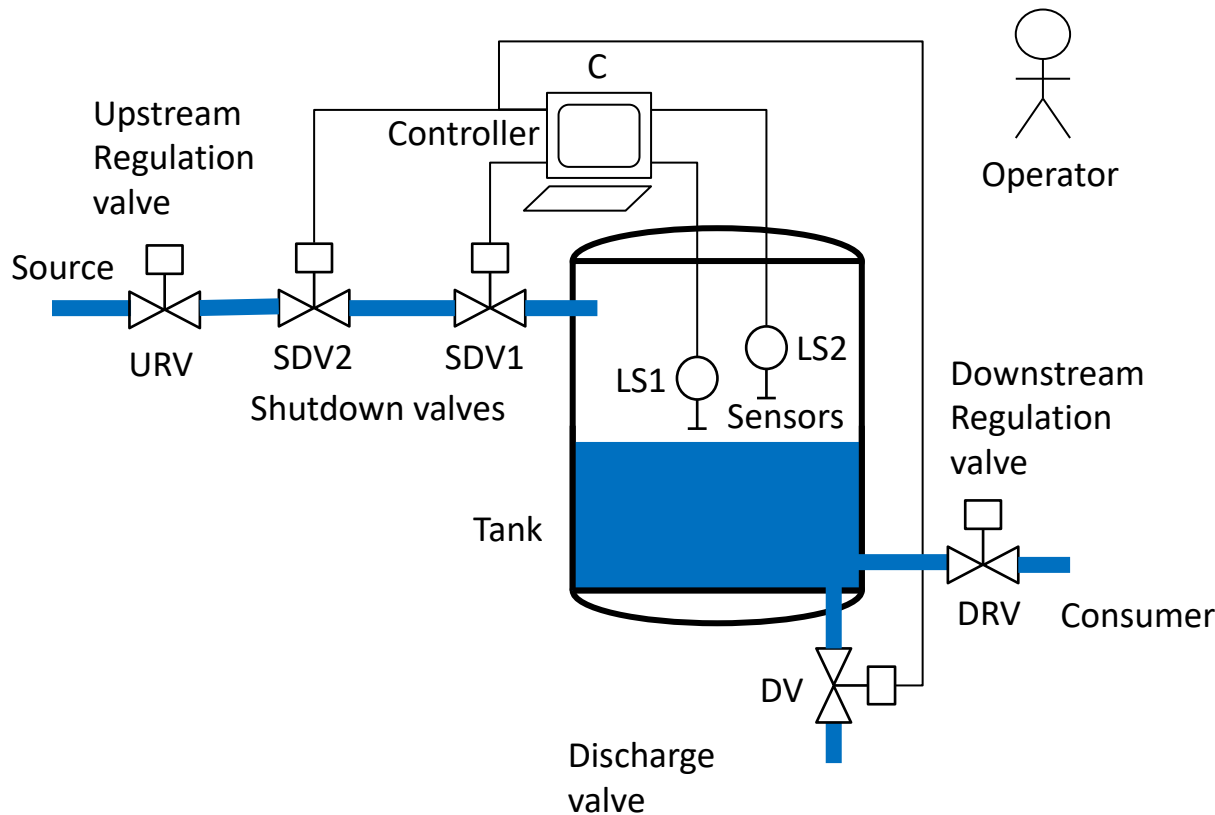
- What is inside the system, i.e. the elements the analyst **designs** or **controls**?
- What is outside the system, i.e. the elements that the analyst does not design nor control but must **interact** with?

The **objective** of the **interface analysis** are:

- To define the **perimeter/boundary** of the system under design.
- To characterize the **environment** of the system, i.e. all **extern systems** with which the system will interact.
- To characterize theses **interactions**.

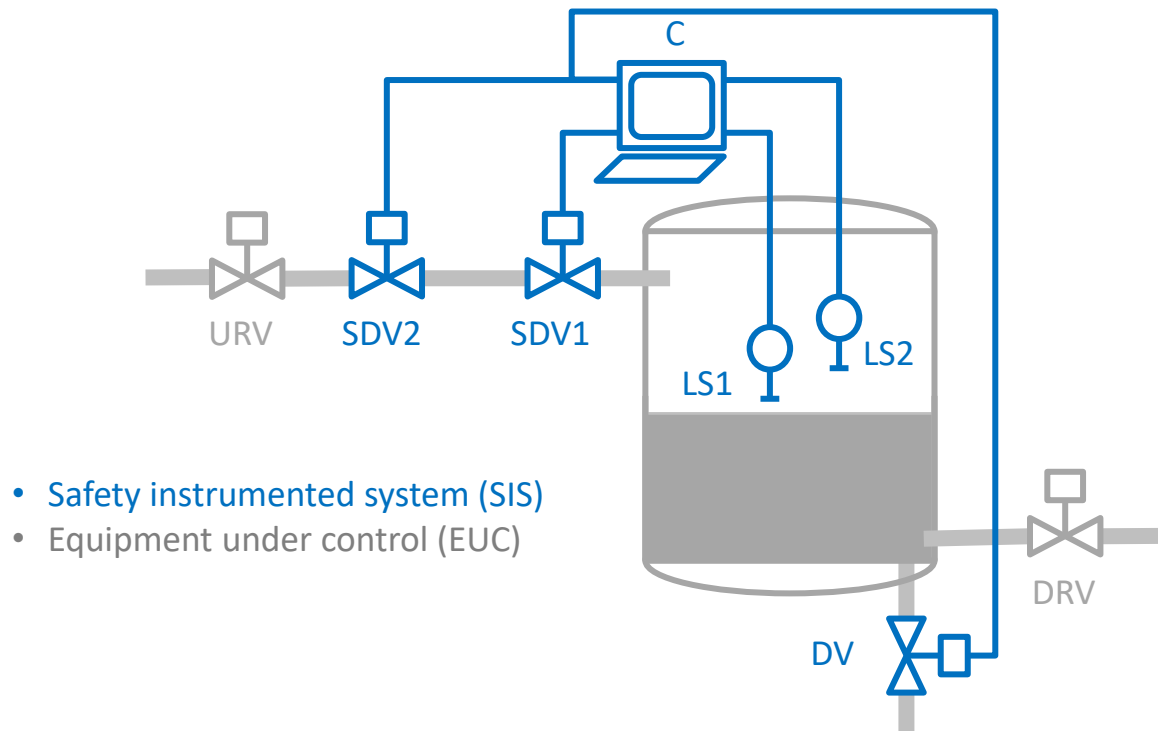
Overflow Protection System

The role of the system is to prevent an overflow of the tank, which may have catastrophic consequences.



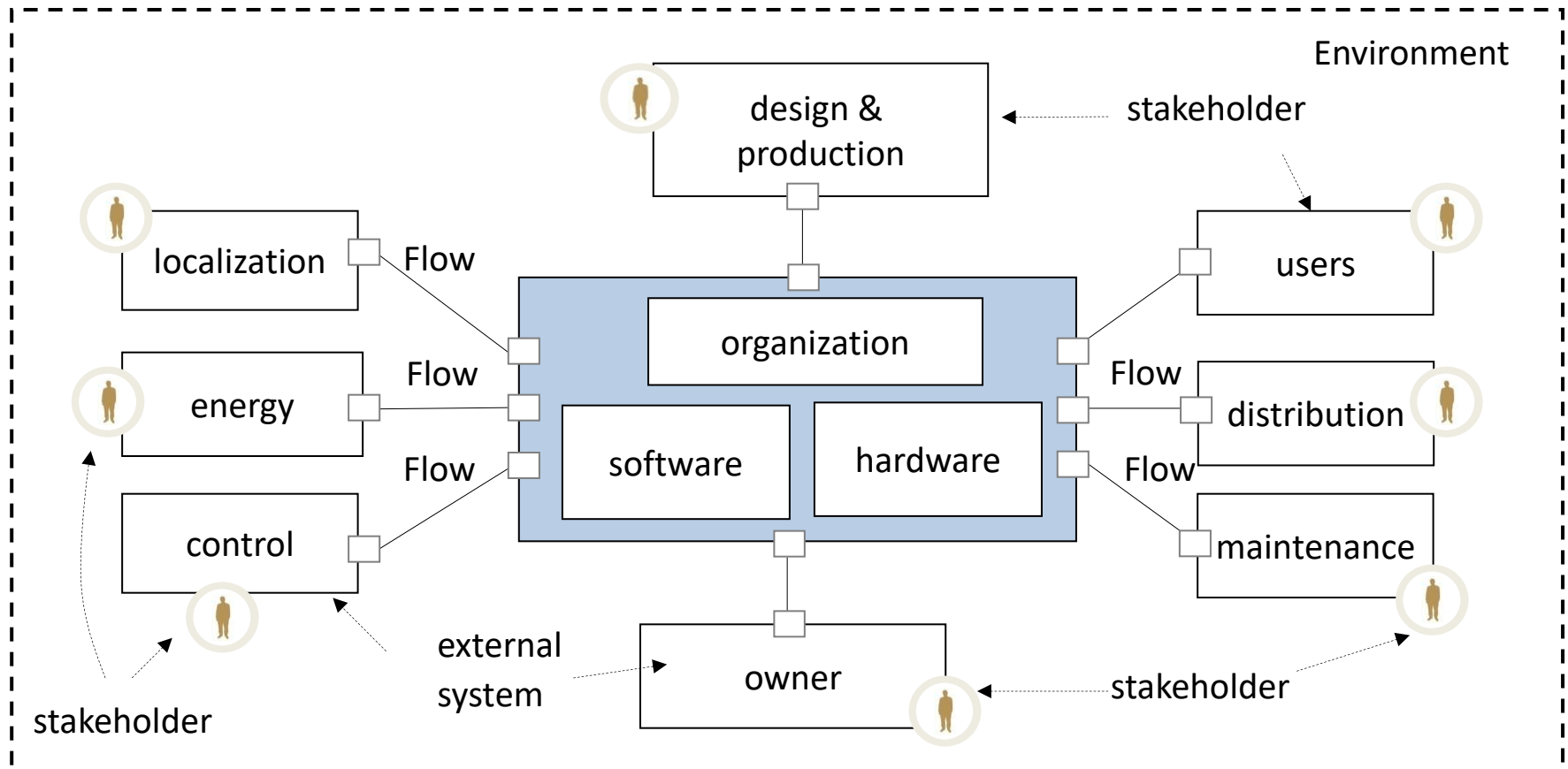
We want to know whether this protection system is good enough for the plant to be operated in sustainable economic and safety conditions, and possibly to suggest modifications to its design.

The System Under Study

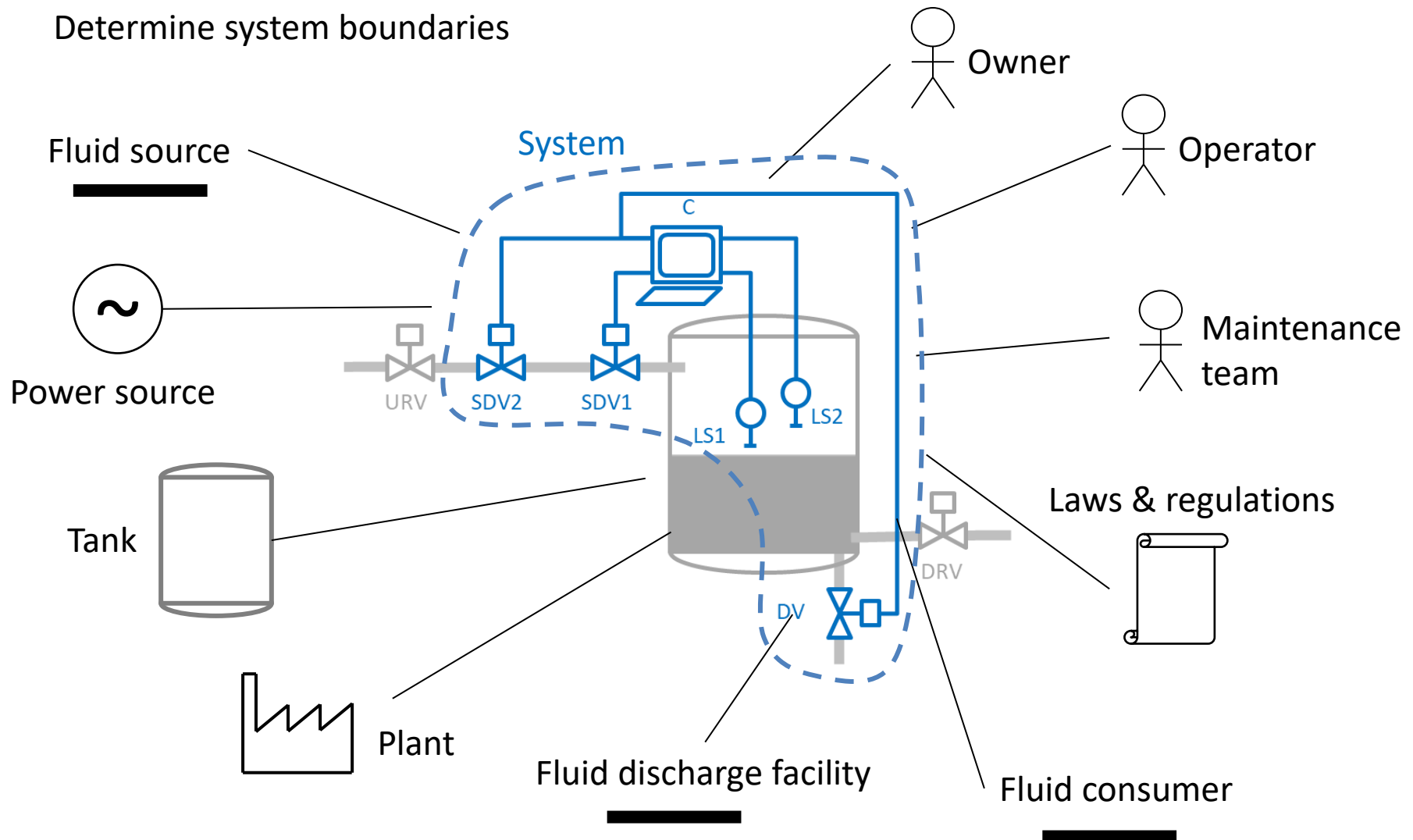


Environment of the System

The **environment** of the system is made of external systems that have an influence on the system. A **stakeholder** is a **human actor** who represents one of the external systems. **Block diagrams** are useful to represent the system and its environment.



Environment of the System

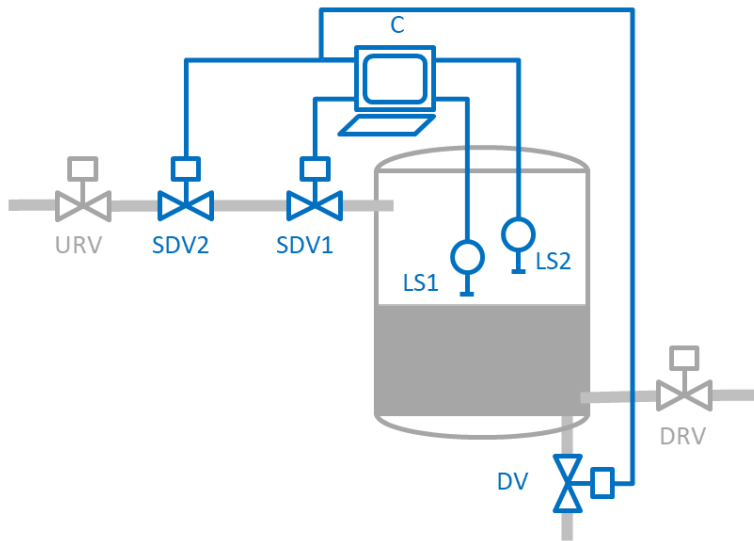


Interactions

Interactions between the system and external systems must be characterized

External System	Interaction
Fluid source	The fluid source supplies the tank with fluid. The system is in charge of the regulation of this input flow.
Fluid consumer	The fluid consumer consumes the fluid stored into the tank. The system has non interaction with it.
Tank	The system is in charge of preventing an overflow of the tank.
Electric power supply	The electric power supply supplies power to the system.
...	...
Owner	There should not be more than 2 false alarms per month.
Laws & regulations	The probability per year of an overflow should be lower than 10^{-5} .

Black Box versus Grey Box



- Safety instrumented system (SIS)
- Equipment under control (EUC)

Ideally, the interface analysis should consider the system as a **black box**.

In practice, this often turns to be impossible. One should refer to internal components of the system.

Nevertheless, one gets a better understanding of the system when trying to consider it “from outside”. This makes it possible to study **tradeoffs** between possible variants/versions...

This is sometimes called a **grey box** approach.

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 2. NEEDS, MISSIONS AND REQUIREMENTS

Needs and Missions

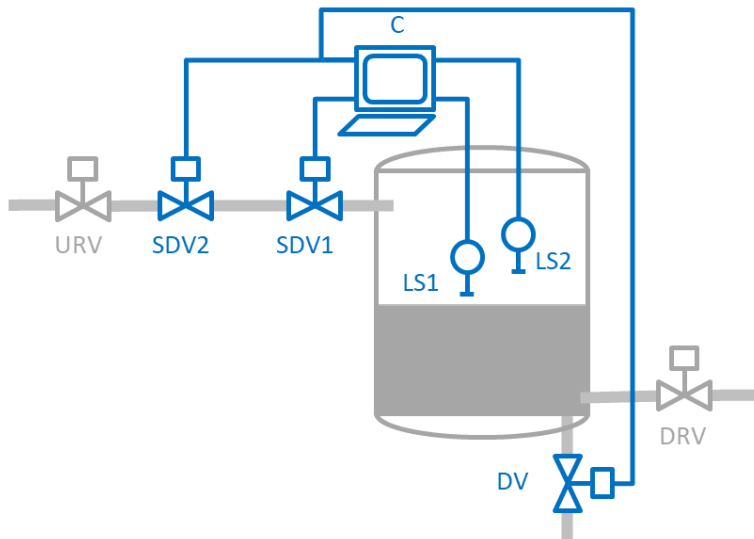
A system is designed and operated because it fulfils certain **needs**. Clarifying what these needs are is the starting point of the analysis of the system.

The descriptions of needs stands in general at a high level and may be a bit vague, subjective, or incomplete.

The **mission(s)** of a system is what the system must **do in its operational context** to satisfy the identified needs. The objective of the description of the mission(s) of the system under study is:

- To define the **operational goals** of the system;
- To clarify its **intended use** in the real world.

Needs and Missions



Need: operate safely the equipment under control

Mission: prevent an overflow of the tank

- Safety instrumented system (SIS)
- Equipment under control (EUC)

Requirements

Needs and missions are too informal to support an unambiguous description of the system. Requirements provide a more formal description.

According the IEEE Standard 1220-1998 of systems engineering:

A **requirement** R(S) on a system S is a **non ambiguous, testable and measurable property** that expresses a characteristic of a constraint that the system **should satisfy** to be **accepted** by the **stakeholders**.

General pattern: *The <<system>> should <<do something>> with <<this level of performance>> in <<that context>>.*

Justification: Any requirement must be justified by some need of an external system w.r.t. the system or vice versa.

Satisfaction criterion: Models are used to make explicit and precise the pragmatic conditions that the system must meet in order to fulfil the requirements.

Presentation of Requirements

Name	Reference	Maturity
<<name>>	<<reference>>	e.g. weak/average/strong
Statement		
The <<system>> should <<do something>> with <<this level of performance>> in <<that context>>		
Justification		
Justification of the requirement (e.g. demand of a stakeholder)		
Satisfaction criterion		
Means to test that the requirement is fulfilled by the system.		

Main Requirement

Name	Reference	Maturity
Safety	REQ 42	Weak
Statement		
The expected production of the sea mining system should be over 10^6 tons of ore per year.		
Justification		
The sea mining operation are profitable only if this threshold is reached		
Satisfaction criterion		
Design a systemic digital twin of the sea mining system and assess they yearly annual production from that model.		

Requirement Engineering

There exists a vast literature on **requirement engineering**, especially in the context of software system engineering. The analyst **elicits** the **corpus** of requirements.

- Higher level requirements can be **decomposed** into lower-level requirements.
- It is often difficult to find the **right number** of requirements: too few requirements let the system is underspecified, too many requirements and it may be overspecified, not to speak about the difficulty to maintain a large corpus.
- The literature often distinguishes **functional requirements** (what the system must do) and **non-functional requirements** (what is its expected level of performance).
- A blind spot of the literature is that to be operated properly, it is in general the case that the system needs some service from its environments. There are thus not only **requirements on the system**, but also **requirements on its environment**.
- Requirements serve more and more as a contractual basis between **OEM** (Original Equipment Manufacturers) and their **suppliers**.

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 3. LIFE-CYCLE AND OPERATION MODES

Life-Cycle

The **life-cycle** of a system is the **full sequence of phases** that a system goes through from the moment a need is identified until the system is retired and disposed of. It encompasses all **activities**, **decisions**, and **stakeholders** involved in conception, realization, operation, and end-of-life.

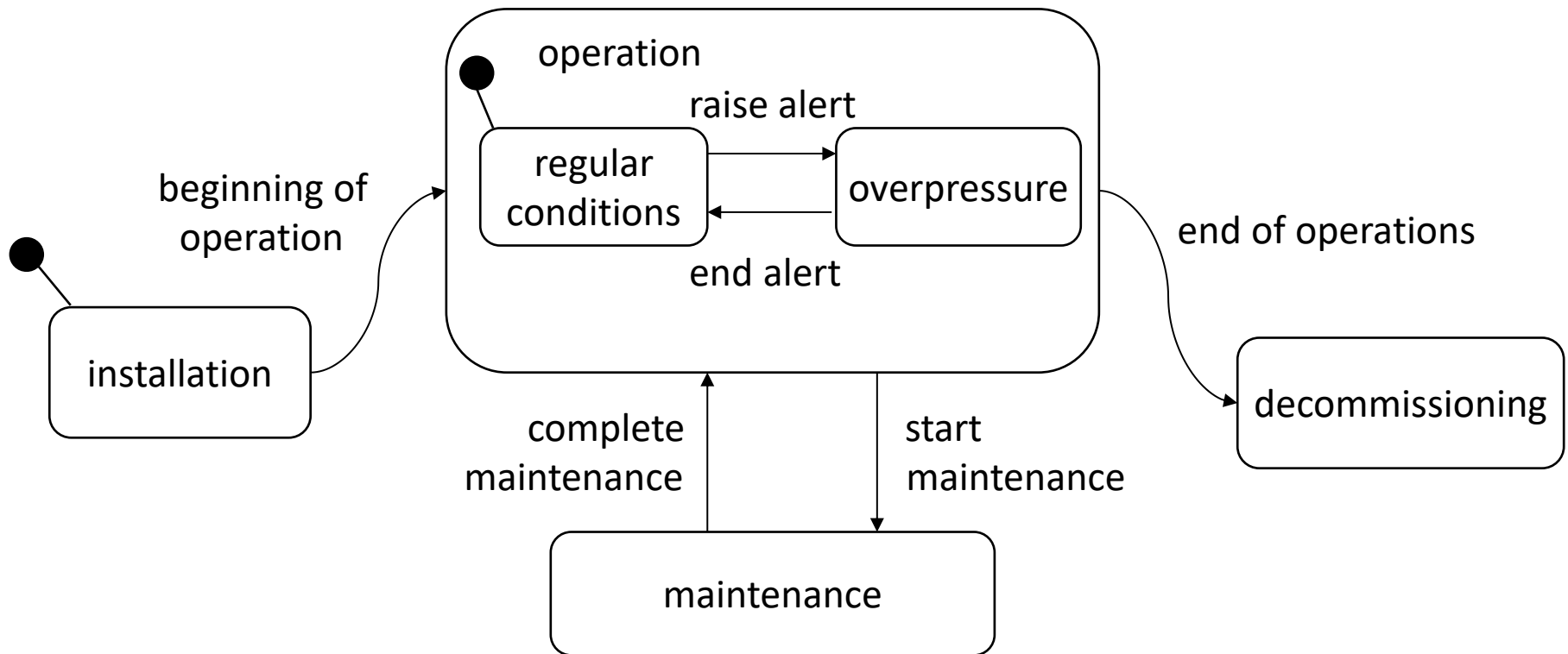
A life-cycle perspective ensures engineering decisions consider **long-term performance**, **cost**, **risks**, and **sustainability**, not just initial development.

Phase	Key Purpose	Activities
Concept / Needs Analysis	Understand the problem and context	Stakeholder analysis, feasibility
Requirements & Architecture Definition	Specify what the system must do and how	Requirements, system modeling, trade studies
Design & Implementation	Realize system elements	Detailed design, coding, manufacturing
Integration & Verification	Assemble and verify compliance	Testing, integration, interface validation
Validation & Deployment	Ensure system meets stakeholder needs	Acceptance testing, installation
Operations & Maintenance	Use the system and sustain performance	Operation, monitoring, upgrades
Disposal / Retirement	Safely remove the system	Dismantling, recycling, decommissioning

Operation Modes

A system goes through different **operation modes**. It changes of mode when some event occurs. Events can be either forecasted/planned or unforeseen. The **missions** of the system differ in general from one mode to another one.

State-charts, also called **state diagrams**, are well suited to represent modes and transitions between modes.



LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 4. USE CASES

Definition and Role

In software and systems engineering, a **use case** is a sequence of actions or steps defining the interactions between the system under study, or parts of it, and one or more external systems.

External systems, whether they are human beings or not, are often called **actors**.

Use cases have three main roles:

1. They avoid **abstract non-sense** by making things very concrete.
2. They cover the **main scenarios of uses** of the system.
3. They prepare **tests** to be performed on the system, once built.

As a general rule, for each feature introduced somewhere in the specification, there must be at least one use case involving this feature.

Closing of Valve SDV1 (1)

Preconditions: The sensor LS1, the controller C and valve SDV1 are functioning.

Postconditions: The upstream flow is stopped.

Trigger: Too much liquid flowing into the tank via the source pipe.

Story:

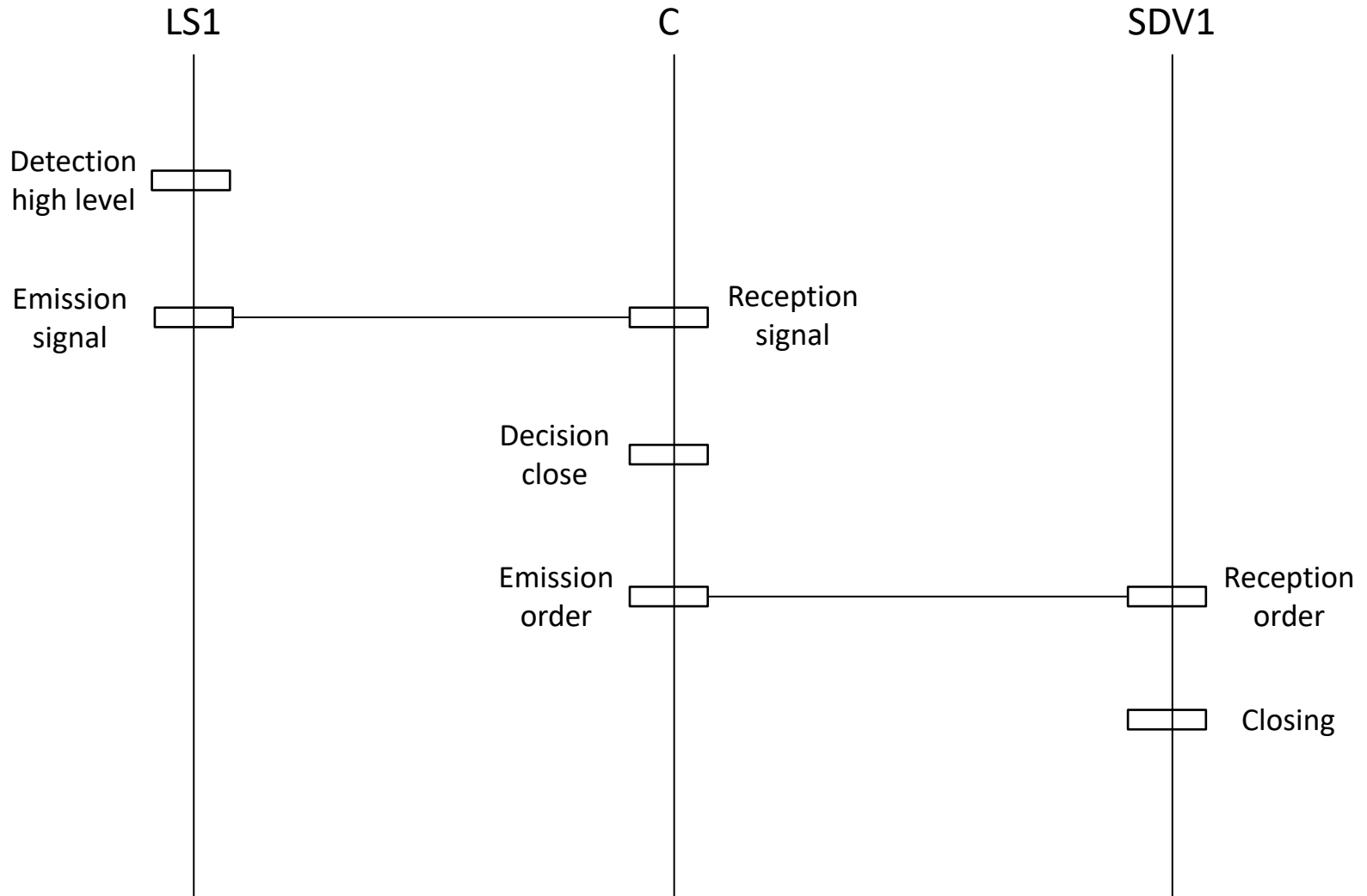
1. Too much liquid is flowing in the tank via the source pipe.
2. The sensor LS1 detects that the liquid goes above the threshold level tl1 in the tank.
3. The sensor LS1 sends a signal for more than x seconds in a row to the controller C.
4. The controller C makes the decision to close the valve SDV1.
5. The controller C sends to the valve SDV1 the order to close.
6. The valve SDV1 closes.

Closing of Valve SDV1 (2)

Examples of questions raised by the use case, e.g.

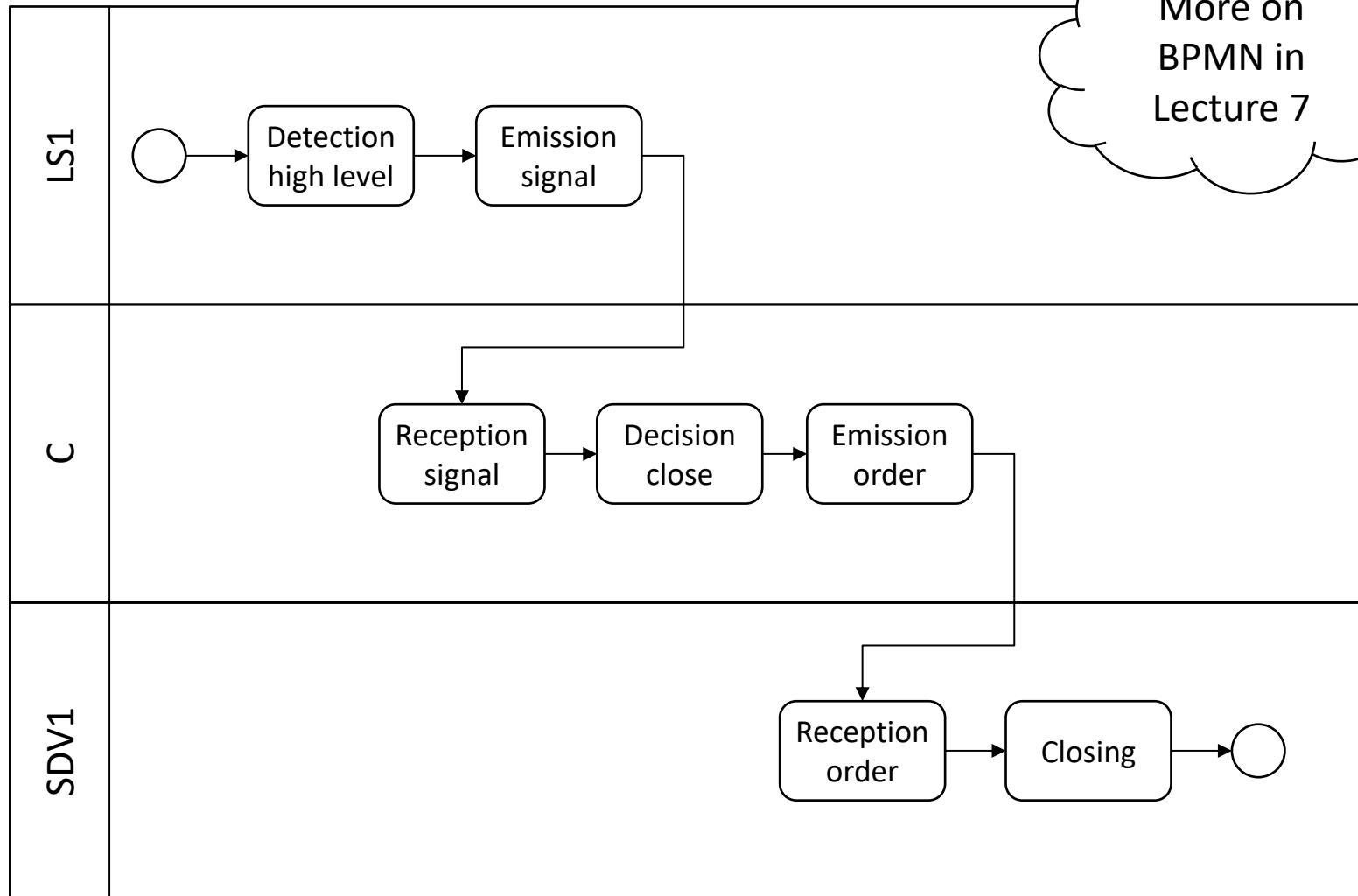
- Is it possible that too much liquid flows "naturally" in the tank?
- If too much liquid flows in the tank, is the consumer pipe insufficient to evacuate the surplus of liquid?
- In case the liquid goes above the threshold level tl1 in the tank, how long does the controller C "wait" before sending the order to close to the valve SDV1?
- Does the valve SDV1 close completely or is it possibly sufficient to close it partially?
- Does the controller C warn the operator in case it sends the order to close to the valve SDV1?
- Does the controller C warn the operator in case of the sensor LS1 detects that the liquid goes above the threshold level tl1 in the tank? If so, after how long?
- How is the system put back in normal operation after the valve SVD1 has been closed?
- Who is involved? The operator? Or is this done automatically when the liquid level goes back under the threshold tl1?
- How long does it take to put back the system in normal operation?
- ...

Interaction Diagrams



BPMN

More on
BPMN in
Lecture 7



Use Cases for Modes

It may be useful to design use cases describing a specific operation mode of the system under study. Strictly speaking, these use cases do not "tell a story" about the system but explain what is the state of the components of the system in that mode.

Title: Maintenance of the level sensors

Preconditions: The maintenance team is on site. The customer is informed.

Postconditions: None

Trigger: Maintenance operation

Story:

- The Upstream Regulation Valve URV must be closed
- The Downstream Regulation Valve DRV must be closed
- The upstream valves SDV1 and SDV2 must be closed.
- The discharge valve DV must be opened.
- The controller must be shutoff.
- The tank must be empty.

Requirements on
external systems

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 5. FUNCTIONAL AND PHYSICAL ARCHITECTURES

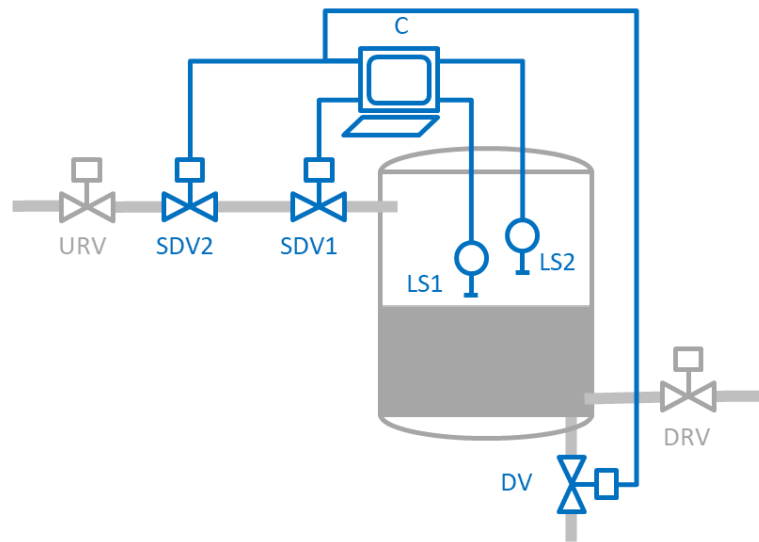
Functional versus Physical Architectures

Model-Based Systems Engineering distinguishes the functional and the physical architectures of a system.

- **Functional Architecture** describes **what** the system must do: its functions, interactions, and the flow of information, energy, or materials. It focuses on fulfilling operational needs and requirements independently of any technical solution.
- **Physical Architecture** describes **how** the system is realized: its components, technologies, and their interfaces in the actual implementation.

Understanding both perspectives is essential in Model-Based Systems Engineering. The functional view ensures that the system delivers the right capabilities, while the physical view ensures that these capabilities can be built, integrated, and maintained. Aligning the two reduces design errors, improves traceability, and supports robust decision-making throughout the life cycle.

Safety Instrumented Systems



- Safety instrumented system (SIS)
- Equipment under control (EUC)

Safety requirements are often met by means of **safety (instrumented) systems**.

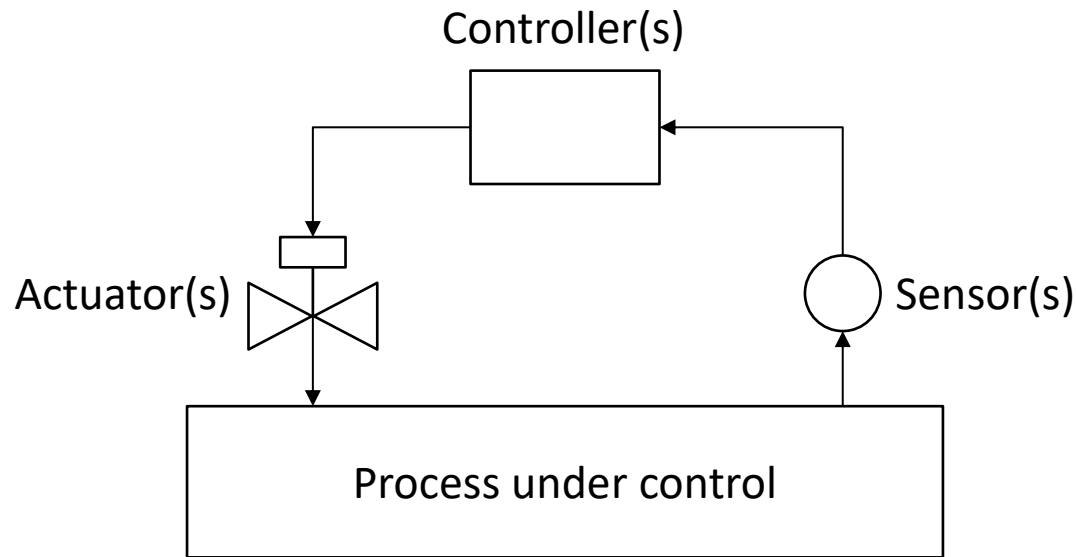
The **safety (instrumented) system** is here a control system made of some sensors, a controller and some actuators (the valves).

Its role is to prevent the occurrence of the **risk** “overflow”, in case of occurrence of the **hazard** “too-high in flow”.

It is a system working **on-demand** (low demand rate) as opposed to safety systems having a **continuous** action on the equipment under control (systems with high demand rate), e.g. the brakes of your car.

Control Systems

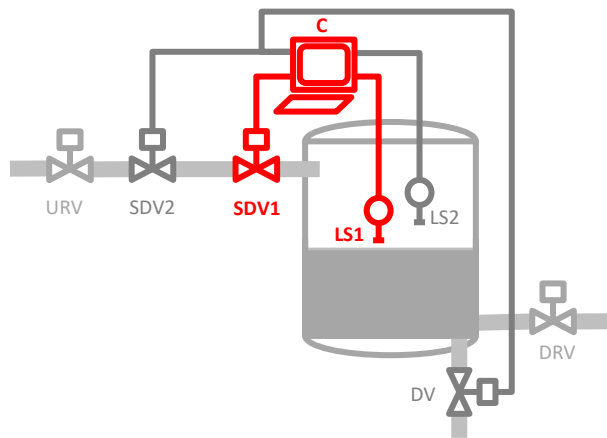
Safety instrumented systems (and active safety barriers) belong to the broad category of **control systems**.



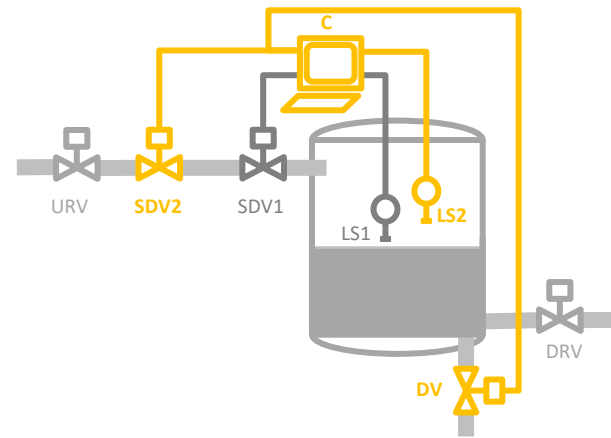
Control systems introduce on a **feedback loop** in the process under control.

Determining Main Functions

The system acts as successive **safety barriers** for the equipment under control.



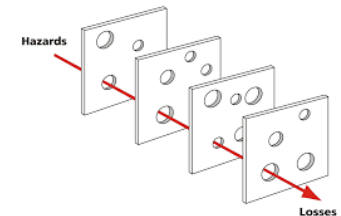
First (active) barrier prevention



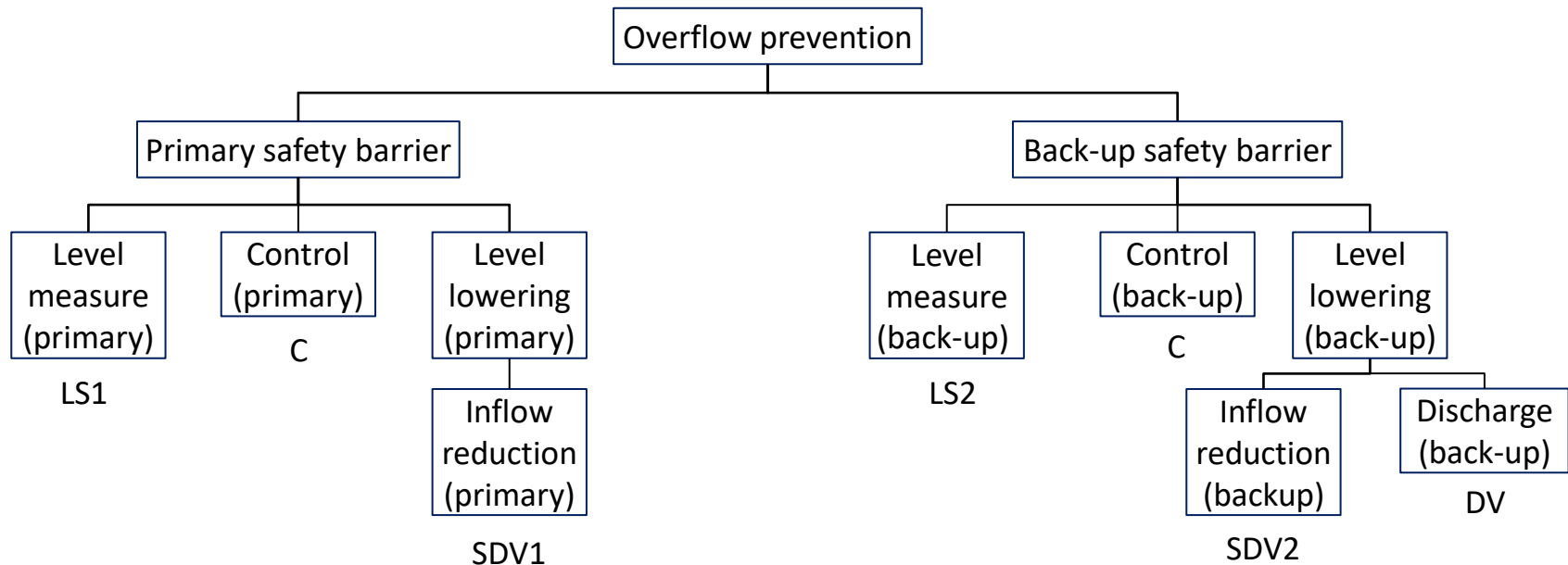
Second (active) barrier prevention

The failure of the controller causes the **loss** of the first and the second barriers. They are not **independent**.

The failure of the controller is a **common cause failure** for the two barriers (there may have some other, e.g. a shock on the two shutdown valves).

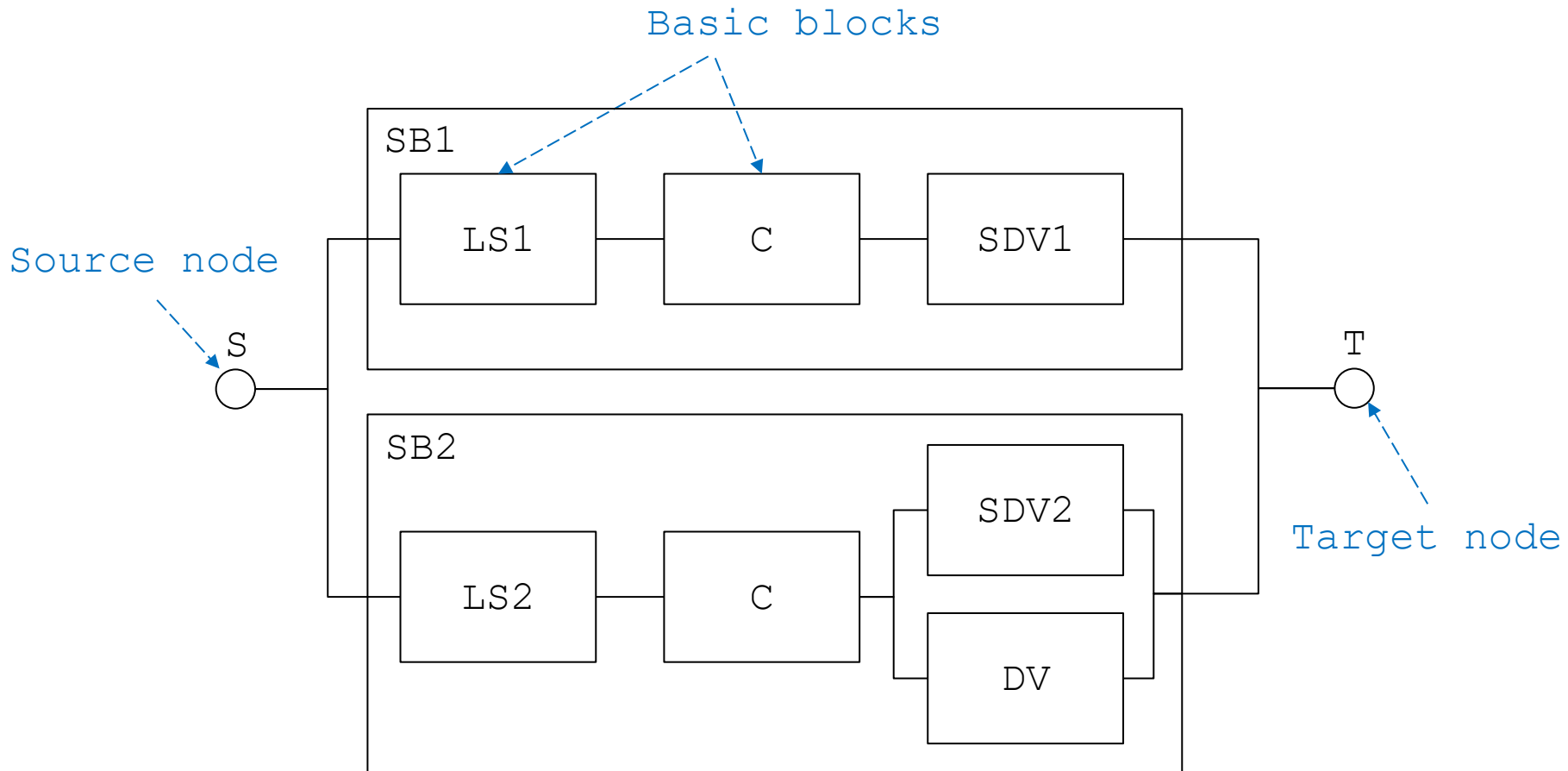


Functional Architecture



Functions are decomposed into sub-functions. These sub-functions are themselves decomposed into sub-sub-functions and so on until the **suitable granularity** is reached. The result is a **tree-like representation**. However, some nodes may be shared by several branches, e.g. to represent support functions like power supply.

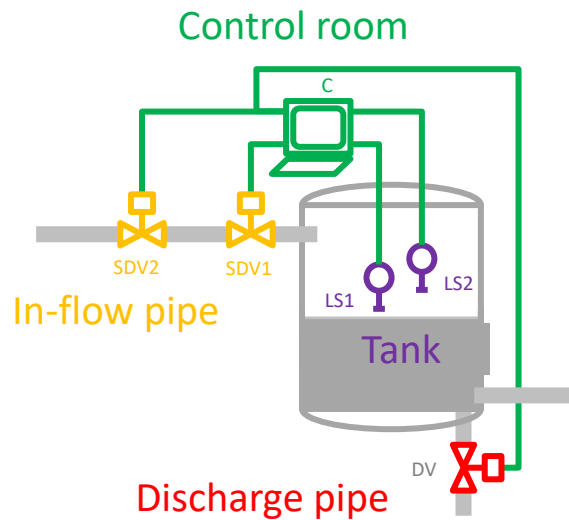
Functional Block Diagrams



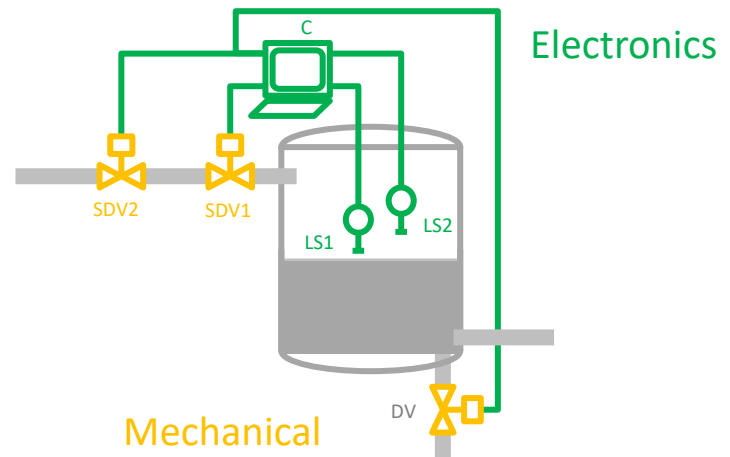
- Hierarchy more difficult to grasp
- Flows (of energy, matter, information) easier to represent

Determining Main Components

Several physical decomposition of the system may be possible, depending on the point of view. E.g.



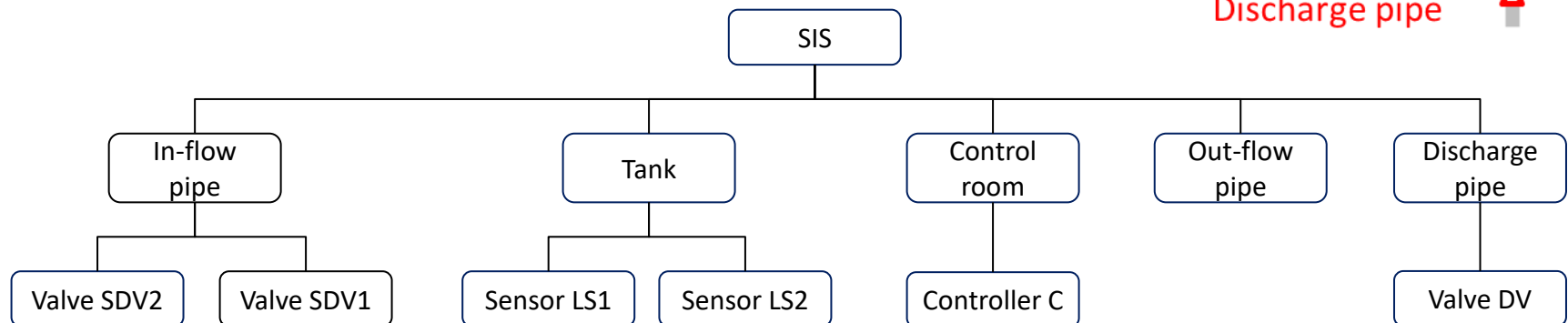
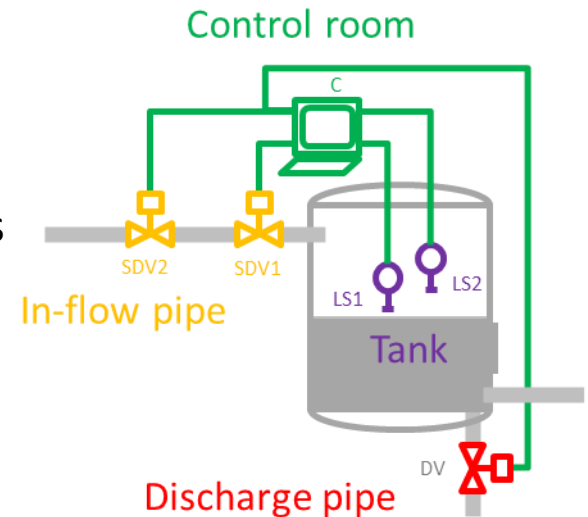
Zonal decomposition



Decomposition by types of components

Physical Architecture

The system is decomposed into sub-systems. These sub-systems are themselves decomposed into sub-sub-systems and so on until the **suitable granularity** is reached.



Design Structure Matrices

Design Structure Matrices (DSM) are a simple but efficient tool to understand and to organize **interactions** between components of a system. In a word, DSM are a good support tool to **structure** the vision (and the models) of a system.

A **Design Structure Matrix** is:

- a **square matrix** showing **dependencies** amongst the components (the component 1 uses the component 5 and is used by the component 2),
- that allows to represent asymmetric relationships.

The analysis of a DSM consists in permuting components, i.e. lines (and thus columns) so to:

- visualize **architectural patterns** (**modules**, **loops**), and
- find out **modules** (**clusters**) and **sequences**.

Components	1	2	3	4	5
1		x			
2			x	x	
3				x	
4					x
5	x				

When designing a DSM, it is of primary importance to well document interactions. It is sometimes of interest to design one matrix per type of interaction.

Design Structure Matrices

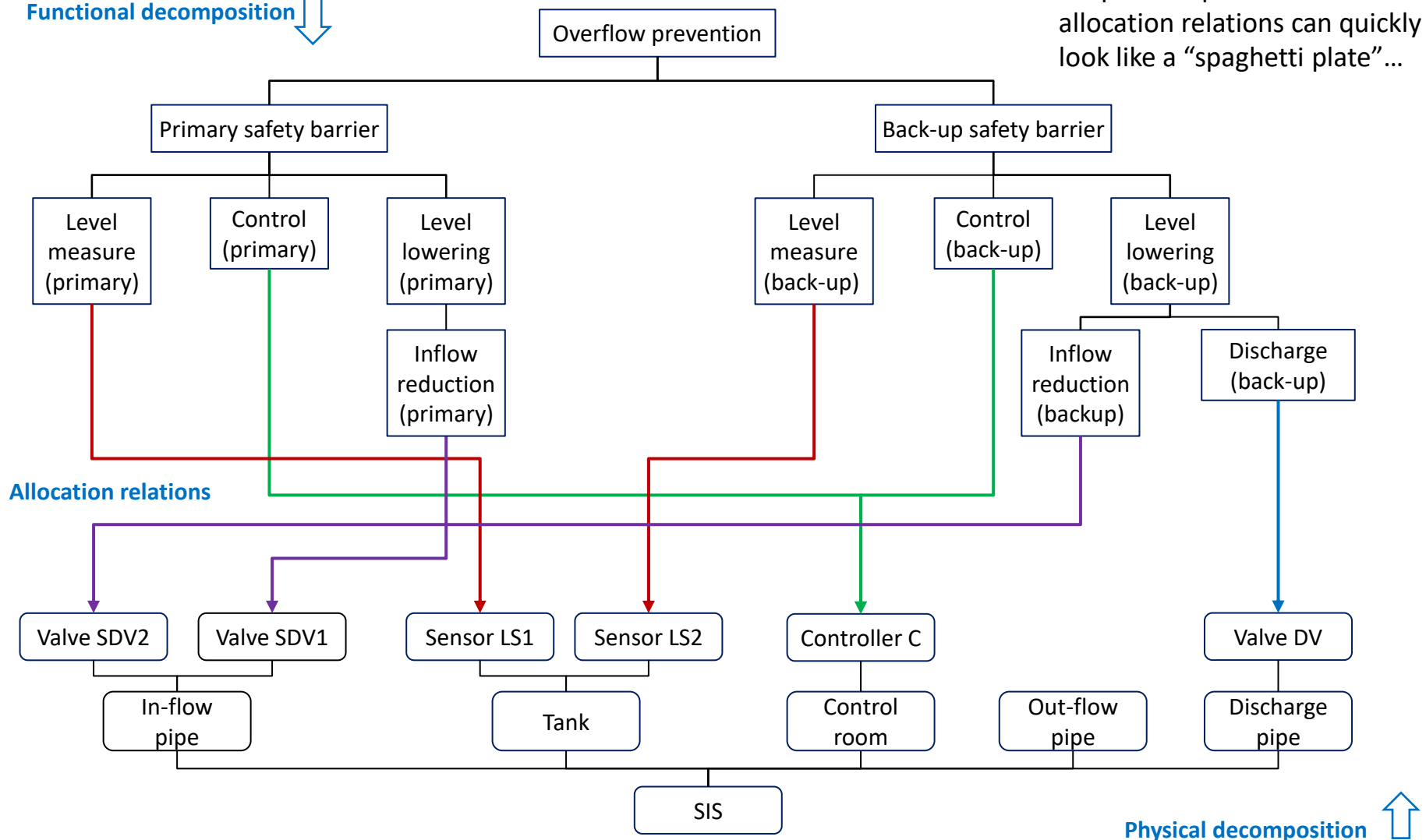
	1	2	3	4	5	6	7	8	9	10	11
1. In flow pipe		L	L								
2. SDV1	L										C
3. SDV2	L										C
4. Tank					L	L					
5. LS1				L							C
6. LS2				L							C
7. Out-flow pipe											
8. Discharge pipe									L		
9. DV								L			C
10. Control room											L
11. Controller		C	C		C	C			C	L	

Cells of the matrix may contain various information (here L: location, C: connection).
Lines may be reorganized of to make relevant information emerge.

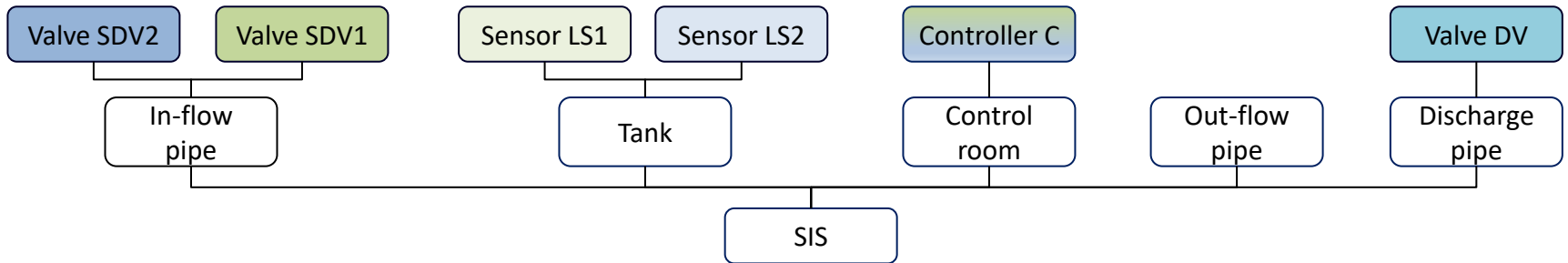
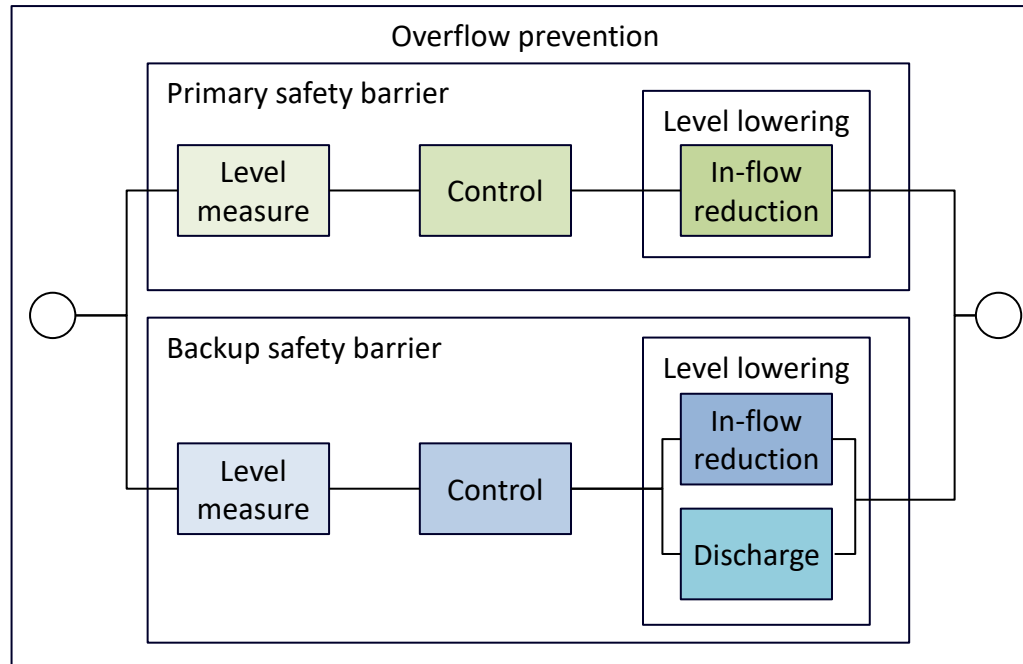
Allocation of Functions onto Components

Functional decomposition ↓

Graphical representations of allocation relations can quickly look like a “spaghetti plate”...



Logical Architecture of the System



Capabilities versus Functions

In modern systems engineering, especially within defense and enterprise architectures, **capability** has become the preferred term because it goes beyond what a system does.

- A **function** describes an internal transformation or activity within the system (*the “what”*).
- A **capability** expresses the **ability to deliver value in an operational context** (*the “why” and for whom*).

Capabilities:

- are tied to mission outcomes and stakeholder needs, not just internal behavior
- remain valid even when technology or system configurations evolve
- support high-level planning, trade-offs, and portfolio management across multiple systems
- allow reasoning about performance, availability, and constraints in real operations

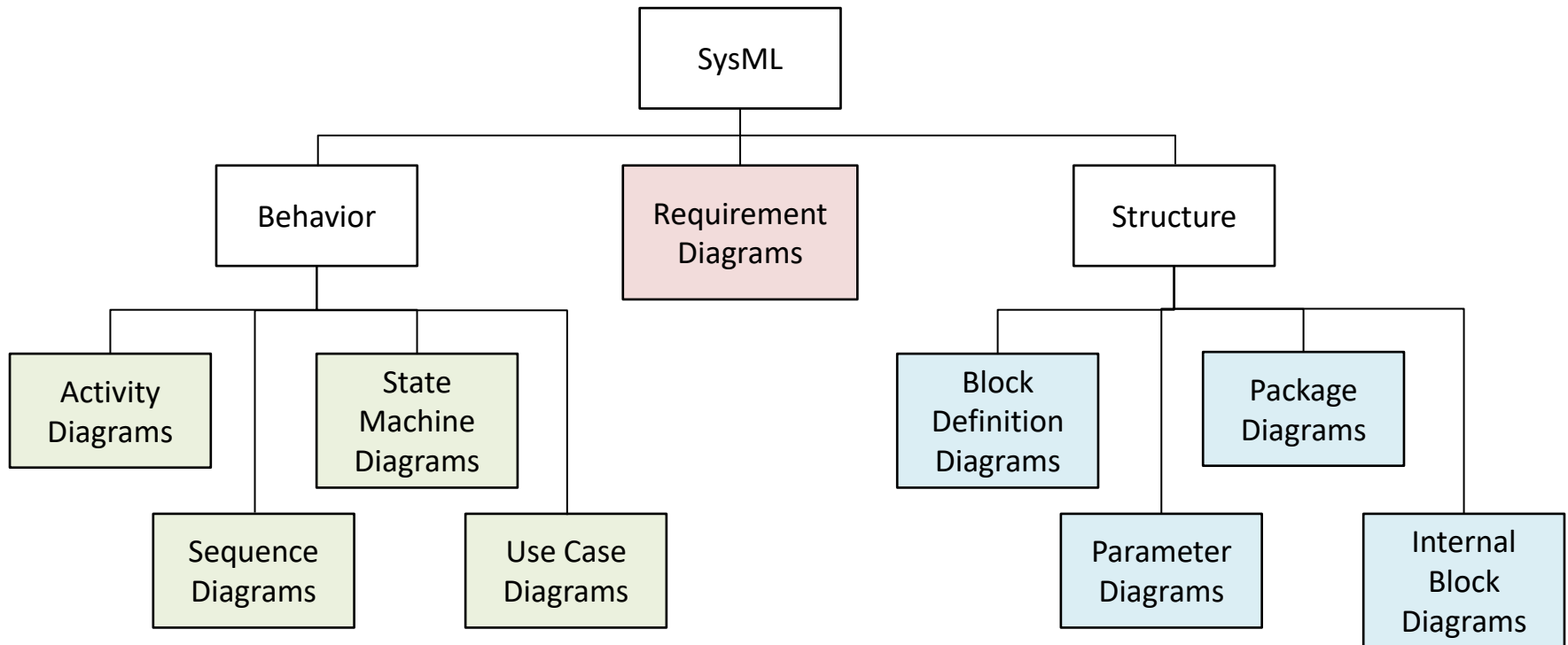
LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 6. SYSML AND ARCHITECTURE

FRAMEWORKS

SysML

SysML (Systems Modeling Language) is a set of normalized diagrams. It is by far the most popular such set, to the point that Model-Based Systems Engineering is often confused with the use of SysML. SysML includes 9 types of diagrams.



Systems Architecture

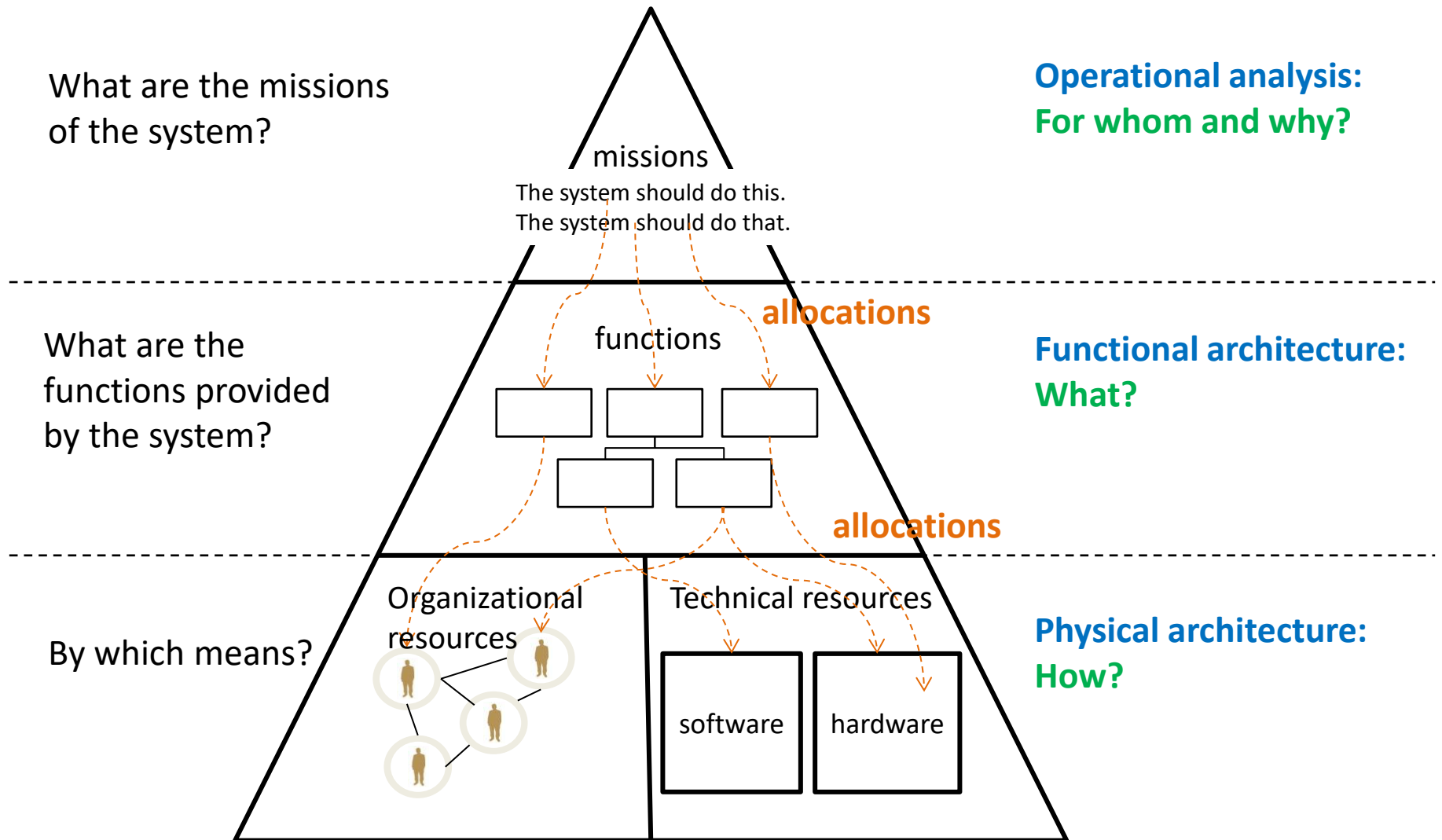
Systems Architecture is an emerging discipline that aims primarily at **integrating** other engineering disciplines. Industrial systems are nowadays so complex that the traditional discipline silo-ed, **divide-and-conquer**, approach for system design is no longer sufficient.

Systems architecture designates both a **process** and the **result of this process**.

System architects apply methodologies that involve the design of models. These methodologies are often called **architecture frameworks**. The ISO/IEC/IEEE 42010 Conceptual Model of Architecture Description defines the term architecture framework as: *"An architecture framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community."*

The **SysML** modeling notation is often used to design models, although it has essential drawbacks.

CESAMES Architecture Framework



CESAMES Architecture Framework

Points of view	Questions	Analyses	Keywords	Models
Operational	For whom? Why? When?	Analysis of the environment of the system	Mission, use case, requirement, operational context, life cycle	Interactions of the system with external systems
Functional	What?	Abstract analysis of the system	Function, task, process, mode	Abstract functions of the system
Physical	How?	Concrete analysis of the system	Component, part, architecture, configuration	Concrete components of the system

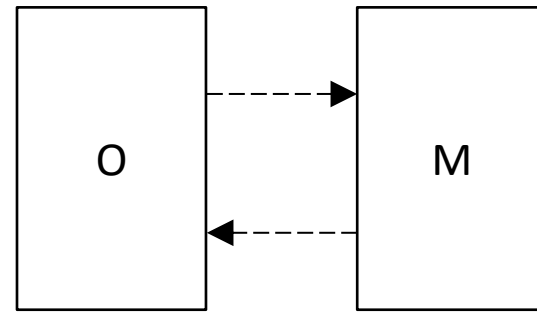
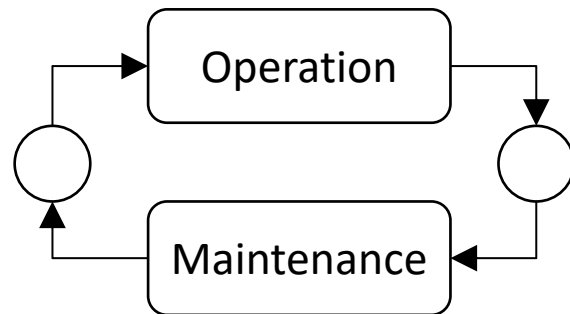
LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 7. DIAGRAMS OR MODELS?

Diagrams versus Models

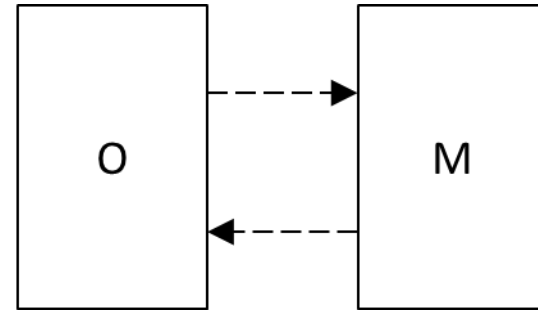
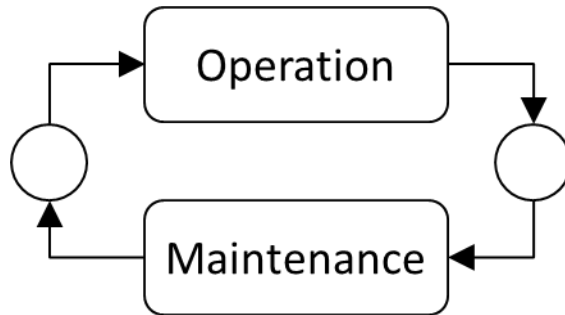
System dynamics and **systems engineering** make use of a lot of **diagrams**, but can we call them **models**? In other words, are these diagrams:

- **Simple drawings/schematics**, possibly obeying some **design conventions**?
- **Graphical models** translatable in **mathematical equations**?



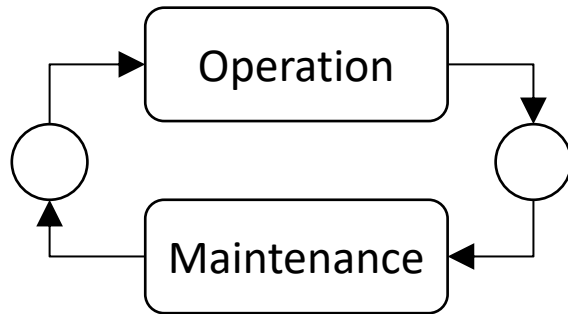
Two **diagrams** *represent* graphically two **business processes** themselves *representing* the behavior of a **periodically maintained component**.

Fundamental Remarks (1)



1. When the “model” gets **complex** (size and/or involved concepts), it is impossible to describe it graphically. Rather, **graphical representations** are **views** on the model.
2. Consequently, we need some **text** in addition to **graphics**.
3. **Diagrams** *denote* graphically **mathematical objects** of some sort which themselves *refer to* some characteristics of **concrete (real-life) systems**.

Diagrams versus Models (2)



```
domain Phase {OPERATION, MAINTENANCE}

system Component
  Phase phase(init = OPERATION);
  bool operating(init = false);
  bool maintaining(init = false);
end

activity Component.Operation
  trigger:
    return phase==OPERATION and not operating;
  start:
    operating = true;
  completion: {
    phase = MAINTENANCE;
    operating = false;
  }
  duration:
    return 6;
end
```

Fundamental Remarks (2)

4. Computers do not understand anything, they barely execute predetermined sequences of instructions. **Models in silico** described textually and/or graphically are eventually **sequences of symbols**. Computers **process** these sequences.

5. Hence the three central questions:

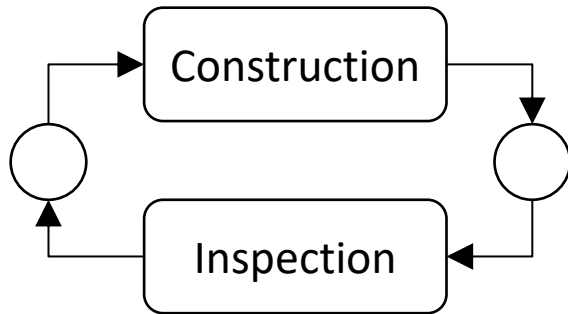
- How to decide whether a given text or graphics is a well-formed model or not?
- How to associate a well-formed model with a well-defined mathematical object?
- How to relate this mathematical object with a characteristic or set of characteristics of a concrete, real-life system?

Syntax

Semantics

Pragmatics

Diagrams versus Models



```
domain Phase {CONSTRUCTION, INSPECTION}

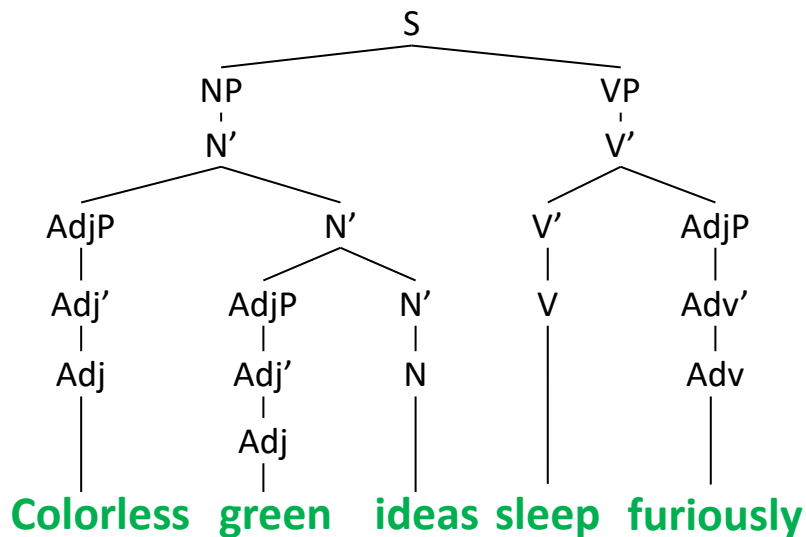
system Building
  Phase phase(init = CONSTRUCTION);
  bool constructing(init = false);
  bool inspecting(init = false);
end

activity Building.Construction
  trigger:
    return phase==CONSTRUCTION and not constructing;
  start:
    constructing = true;
  completion: {
    phase = INSPECTION;
    constructing = false;
  }
  duration:
    return 6;
end
```

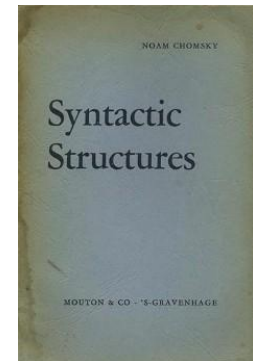
Syntax, Semantics and Pragmatics (2)

The concepts of **syntax**, **semantics** and **pragmatics** come from **linguistics**.

A well-formed sentence may have a perfectly well-defined meaning while making no sense in real life.



Noam Chomsky
1928-



1957

Taxonomy of Models

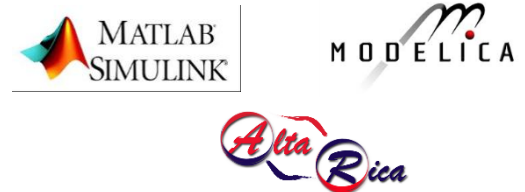
Models are designed at different level of abstraction, for different purposes and in different **modeling formalisms**.

Models to communicate
amongst stakeholders



Descriptive/pragmatic models,
written in *standardized notations*,
sometimes called *semi-formal*

Models to calculate
performance indicators:



Models to generate artefacts
(via code generation) or
physical components (via
additive manufacturing)



Predictive/formal models, that essentially
encode and organize (a certain type of)
mathematical equations

LECTURE 6. MODEL-BASED SYSTEMS ENGINEERING

SECTION 8. WRAP-UP

Wrap-Up

- Model-Based Systems Engineering aims at better understanding why systems under study are designed and operated, what are their interactions with their environment, and eventually what they consists of.

Main Difficulties of Designing a Systemic Digital Twin

Difficulty 1: Dive into the **solution** before stating the **problem**.

Take the time to state the problem is the only way to ensure that nothing has been forgotten and that all potential solutions have been considered.

Difficulty 2: **Overload** the analysis with useless considerations and/no being unable to define correctly the **hierarchy of concerns**.

The objective of a systemic digital twin is not to mimic accurately the system under study but to **make decisions** about its design and operation.

Do not hesitate to design simplified models as a means of communication and validation of your approach.

Difficulty 3: **Reason too abstractly** and/or **make overcommitting assumptions**.

There is a permanent risk of falling into “**abstract non-sense**” and to make assumptions that do not correspond to the “reality”. It is of primary importance to always go back to **concrete issues**, typically by designing **use cases**.

Difficulty 4: Being stuck due to the **lack of data**.

Wrap-Up

- A model should be seen a part of a **constructive** and **pragmatic proof** that it is possible to design, to build and to operate a system having some specified properties. These properties are formalized as **requirements**.
- The design a **systemic digital twin** is an **iterative process**.
- The Σ^{TM} **modeling framework** aims at designing systemic digital twins efficiently and accurately using the Σ^{TM} technology. It consists of **8 views** on the system:

