Institut für Intermodale Transport- und Logistiksysteme

Automation Engineering (Automatisierungstechnik)

Lecture Notes

Jürgen Pannek

April 7, 2023

Jürgen Pannek
Institute for Intermodal Transport and Logistic Systems
Hermann-Blenck-Str. 42
38519 Braunschweig

FOREWORD

During winter term 2022/23 I give the lecture to the module *Automation Engineering (Automatisierungstechnik)* at the Technical University of Braunschweig. To structure the lecture and support my students in their learning process, I prepared these lecture notes. As it is the first edition, the notes are still incomplete and are updated in due course of the lecture itself. Moreover, I will integrate remarks and corrections throughout the summer term.

The aim of the module is to provide participating students with sound basic knowledge of terms and methods within the area of automation engineering. The students will be able to reproduce, describe and apply the latter and can explain modeling, classification, control and coupling of technical processes for basic examples. They are also able to analyze information handling in technical processes and information transfer. Students are capable to fundamentally determine the organizational, distribution and communication structures of automation systems for simple case studies. In addition, the students can describe basic aspects of modularization, standardization and automation. The students understand the digitization topics industrial internet, cloud and cyber-physical systems. As such, the students can reproduce the approaches of knowledge management, industrial big data and decision support.

To this end, we discuss

- Aim of automation engineering

- Basics, tasks and methods of automation

- Coupling and hierarchies of systems

- Information and information management

- Control, modularization and standardization in automation

- Digitalization for industrial internet, industrial could and CPS

■ Basics of knowledge management, industrial big data and decision support

within the lecture and support understanding and application within the tutorial and laboratory classes. The module itself is accredited with 5 credits.

During the preparation of the lecture, I utilized the books of Lunze [17] and Plenk [23] for automation engineering basics. Regarding digitalization, CPS and automation on the overall level, I used the books of Lai [15], Langemann and Turi [16] as well as Stjepandic, Sommer and Denkena [24].

Contents

List of Tables

List of Figures

List of Definitions and Theorems

CHAPTER 1

INTENSION, CONCEPT AND AIMS

Automation engineering deals with monitoring and control of systems or processes. Traditionally, the latter applies to physical activities such as mechanical processes like machines and robots or process technology such as bio- or chemical systems. In modern automation, however, also the automation of software processes plays an important role. Indeed, these physical and cyber worlds become more and more entangled. The development of ecosystems driven by AI methods for intelligent outcomes is just the latest of these aspects. In fact, such approaches are know from literature and practice even from the 1970s and were called expert systems.

1.1. Intension

As said before, automation engineering deals with monitoring and control. Its intention, on the other hand, goes back to improve or substitute a task. Criticism on automation typically addresses the point that automation substitutes labor force, which is not entirely true. While improvement or substitution of tasks indeed typically lead to reduction of required work force for the task itself, new tasks arise in integration, control, maintenance and so forth.

In history, automation engineering has gone through different stages:

- Approximately 800 B.C. the ancient Greek word *automaton* was first mentioned dealing with devices like automatically driving tripods or automated doors of temples. While these were implemented as feed forward, the first known feedback device is the water clock of Ctesibius (285 –222 B.C.). His invention remained to be the most accurate time measurement device till Huygens (1629 – 1695) invented the pendulum clock. Moreover, the first steam engine was developed by Hero of Alexandria (10 – 70). Other basic examples can be found, e.g. in the Middle East, China and Mayan cultures.

- The period we call First Industrial Revolution (1760 – 1840) started by the re-invention of the steam engine (respectively its governor) by Watt (1736 – 1819) with improvements by Siemens (1823 – 1883) and the programmable loom by Jacquard (1752 – 1834). The final step in this period was the development of a formal description for controlling a process by Maxwell (1831 – 1879). This period is also called mechanical revolution.

- Being able to design machines, the Second Industrial Revolution (1870 – 1915) was driven by development of electricity and electric devices. Here, the focus moved from single machines to efficient manufacturing methods such as production lines and the complete work split called Taylorism. The period is also called electrical revolution.

- The Third Industrial Revolution started around 1950 and focused on complex processes and digitalization. Considering automation, the programmable logic controller (PLC) was the major step to implement advanced control methods industrially. This technical invention was backed up by theoretical innovations from Kolmagorov (1903 – 1987) on complexity, Kalman (1930 – 2016) on observers and Bellman (1920 – 1984) on optimization. Additionally, the concept of a robot was introduced. As it it characterized by computers, this phase is also called digital revolution.

- Lately, the Fourth Industrial Revolution is postulated, which utilizes interconnected and self-functioning systems.

> **Remark 1.1**
> *Note that the above interpretation is focused on automation. Regarding other topics such as transport, capitalism or social development, other classifications are used.*

Nowadays, automation systems are components of enterprise resource planning (ERP) tools, still combining information for monitoring and control. Figure 1.1 shows a current default path of automation. It includes the steps taken in the Second Industrial Revolution (standardization and modularization) and the Third Industrial Revolution (automation and lean). We like to point out the the layers *integration*, *optimization* and *leading* also existed and were managed prior to the Forth Industrial Revolution. Yet now, the quantification of the five management tasks

- planning
- organizing
- staffing
- leading
- controlling

may be done much more accurately.

The first and overall question is why should one bother to deal with automation in the first place. According to the U.S. National Institute of Standards and Technology, there are two strategies for
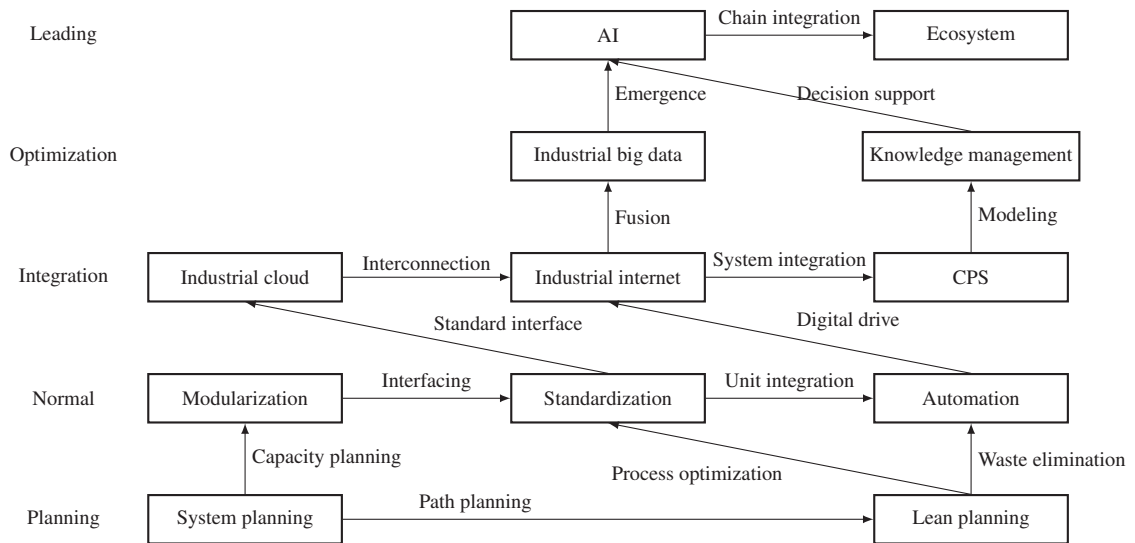
Figure 1.1.: Integrated implementation path of automation

companies to compete for market shares, the so called *cost leadership strategy* and the *differentiation strategy*. These strategies are derived from a company vision and address the market part, hence other strategies exist attending the remaining enterprise parts.

As illustrated in Figure 1.2, the strategies themselves can be broken down into the four goals agility, quality, productivity and sustainability.
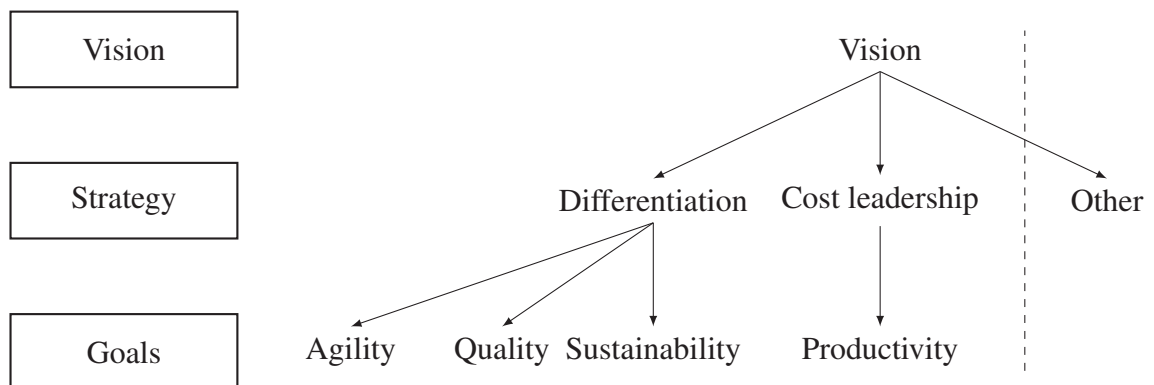
Figure 1.2.: Derivation of strategies and goals from vision

Remark 1.2

It is worth mentioning that strategies are very different around the globe. These are driven by available skills in the respective parts of the world. In the US where a large number of IT giant companies are situated, the National Manufacturing Innovation Network is working on strategies to apply AI and big data methods. The Japanese Industrial Value Chain promotes a CPS based

approach using its long history and strong background in robotics. The Chinese 2025 program makes use of its enormous work force to obtain cost leadership and push quality next. It is strongly connected to the German Industrie 4.0 program, which makes use of the high-end proficiency and quality of production and education of labor force. In contrast to both the Japanese and US programs, the Chinese and German programs start from production to AI, while the other start from AI to production.

In order to be successful, we need to balance these four goals. To this end, we have to define measures for each goal, which then allows us to define SMART projects (specific, measurable, ambitious, realistic, terminated) and avoid waste or failure in architecture and implementation. The following Table 1.1 give some examples of possible performance criteria.

Table 1.1.: Decomposition and measurement of key target capabilities

Strategy	Goals	Capability decomposition	Performance measurements
Cost leadership	Productivity	○ Production capacity	○ Product produced by machine/line/unit/factory per time period
		○ Overall equipment effectiveness	○ Availability · performance · quality
		○ Material and energy efficiency	○ Substance/energy used for specific output
		○ Artificial efficiency	○ Labor hours per unit
Differentiation	Agility	○ Response speed	○ Response time, transaction cycle
		○ On time delivery	○ Rate to complete and deliver
		○ Fault recovery	○ Rate of downtime during operation
	Quality	○ Product quality	○ Customer return/rejection
		○ Innovation	○ Innovation cycle time
			Continued on next page

Table 1.1 – continued from previous page

Strategy	Goals	Capability decomposition	Performance measurements
		○ Diversity	○ Product family and personalization
		○ Service	○ Customer's evaluation
	Sustainability	○ Product	○ Recyclability, energy efficiency, lifetime, reusability, manufacturability
		○ Process	○ Energy use, CO_2 balance
		○ Logistics	○ Transport energy

Having clarified the intension of automation, we next connect these to actual tasks.

1.2. Concept

Automation engineering focuses on systems or processes. Hereby, a process is the overall way a task is done while a system represents the connection of different interacting components to realize a given task. In many cases, these two terms can be used interchangeably. Formally, Figure 1.3 illustrates both terms.
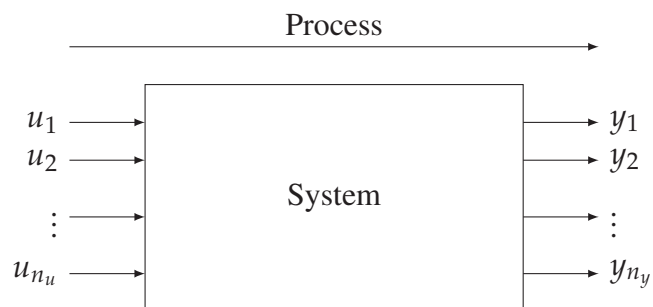
Figure 1.3.: Term of a system and a process

The interdependence of systems with their environment is given by so called *inputs and outputs*. In the literature [22], we see the following description for a system (translated from German):

> A system is a set of interrelated elements that are viewed as a whole in a particular context and considered as distinct from their environment.
>
> DIN IEC 60050-351 (2014)

Building on this description, a process is given as follows (translated from German):

> A process is the entirety of relations and interacting elements in a system through which matter, energy or information is transformed, transported or stored.
>
> DIN IEC 60050-351 (2014)

Within a process, we narrow down on elements within a system, which are connected via signals (translated from German):

> A signal is a physical quantity that conveys information about one or more variable quantities using one or more of its parameters.
>
> DIN IEC 60050-351 (2014)

More formally, we define the following:

Definition 1.3 (System and process).
Consider two sets $\mathcal{U}$ and $\mathcal{Y}$. Then a map $\Sigma : \mathcal{U} \to \mathcal{Y}$ is called a system and the application of this map to an input $\mathbf{u} \in \mathcal{U}$ to obtain an output $\mathbf{y} = \Sigma(\mathbf{u}) \in \mathcal{Y}$ is called a process.

The set $\mathcal{U}$ and $\mathcal{Y}$ are called input and output sets. An element from the input set $\mathbf{u} \in \mathcal{U}$ is called an input, which act from the environment to the system and is not dependent on the system itself or its properties. We distinguish between inputs, which are used to specifically manipulate (or control) the system, and inputs, which are not manipulated on purpose. We call the first ones *control or manipulation inputs*, and we refer to the second ones as *disturbance inputs*. An element from the output set $\mathbf{y} \in \mathcal{Y}$ is called an output. In contrast to an input, the output is generated by the system and influences the environment.

> **Link:** For further details on how to design inputs/controls such that system properties regarding outputs can be generated, we refer to the lectures *Control Engineering 1 & 2*.

In order to understand and utilize a system or process, we require a model of it. Unfortunately, models represent reality one–to–one to a certain extend only. So if we use a model, there may be deviations between model prediction and reality, especially if long time horizons are considered. The reason for that is due to the following: For a model, we always focus on those aspects we are interested in, and do not try to describe all of reality. Hence, the problem is split into two parts,

- the model, which describes what we are interested in, and

- the environment, which contains everything else.

Since we cannot tell anything about the environment (as it is not modeled), interactions between model and environment can only be interpreted as disturbances.

> **Link:** For further details on system and process modeling, identification and handling disturbances, we refer to the lecture *Systemics*.

To utilize the term system and get to the standard concept of automated systems sketched in Figure 1.4, we first need to clarify basic terms.
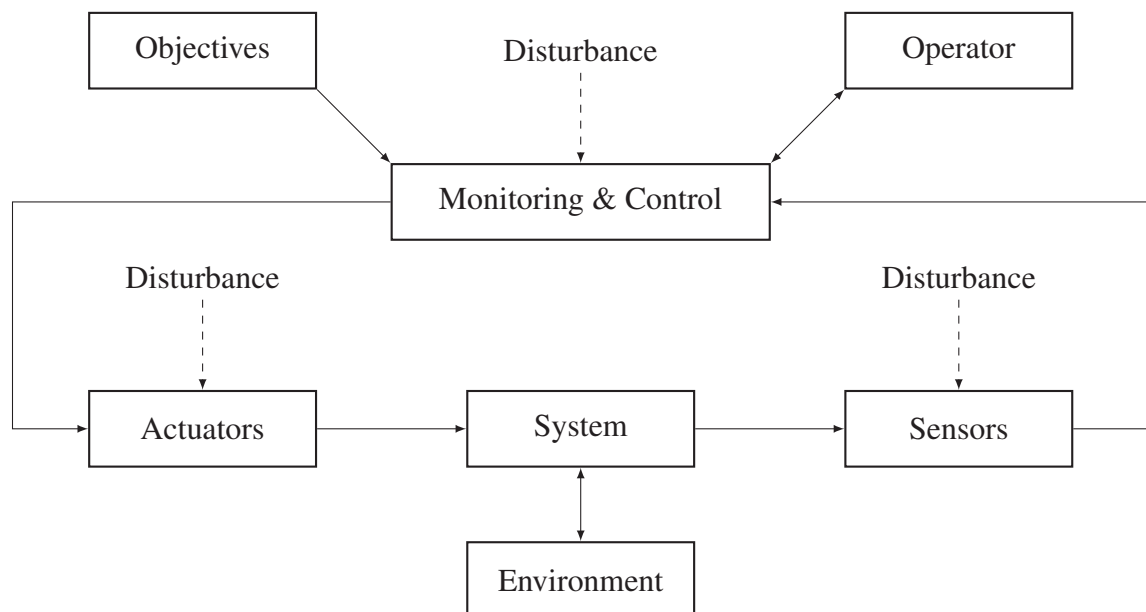
Figure 1.4.: Structure and process of automated systems

The first of these is the notion of time.

Definition 1.4 (Time set).

A *time set* $\mathcal{T}$ is a subgroup of $(\mathbb{R}, +)$.

Using the latter definition, we can use the concept of continuous time, discrete time and event time. Next, we introduce the so called *state of a system*.

Definition 1.5 (State).

Consider a system $\Sigma : \mathcal{U} \to \mathcal{Y}$. If the output $\mathbf{y}(t)$ uniquely depends on the history of inputs $\mathbf{u}(\tau)$ for $t_0 \leq \tau \leq t$ with $t_0, \tau, t \in \mathcal{T}$ and some $\mathbf{x}(t_0)$, then the variable $\mathbf{x}(t)$ is called state of the system and the corresponding set $\mathcal{X}$ is called state set.

In continuous time, that is $\mathcal{T} = \mathbb{R}$, we obtain the standard description of a state space system:

Definition 1.6 (State space – continuous time system).

Consider a system $\Sigma : \mathcal{U} \to \mathcal{Y}$ in continuous time $\mathcal{T} = \mathbb{R}$ satisfying the property from Definition 1.5. If $\mathcal{X}$ is a vector space, then we call it *state space* and refer to

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.1a}$$

$$\mathbf{y}(t) = h(\mathbf{x}(t), \mathbf{u}(t), t). \tag{1.1b}$$

as *continuous time system*. Moreover, $\mathbf{u}$, $\mathbf{y}$ and $\mathbf{x}$ are called *input*, *output* and *state* of the system.

Generally speaking, the (mathematical) description of models varies depending on the considered time, space and amplitude properties. Figure 1.5 provides a rough overview on these characteristics.

Remark 1.7

Regarding time, static models are characterized by the fact that inputs, outputs and measurements of the system are available. In contrast to that, continuous time models exhibit data streams being received continuously. Discrete time models differ from that by the availability of data, which is received at certain, not necessarily equidistant time instances. Last, event triggered models require issues to trigger receiving data.

Regarding space, models may vary from a simple connection to complex systems.

Regarding amplitude, models may differ regarding continuous spaces like e.g. mass and discrete spaces such as gear shifts.
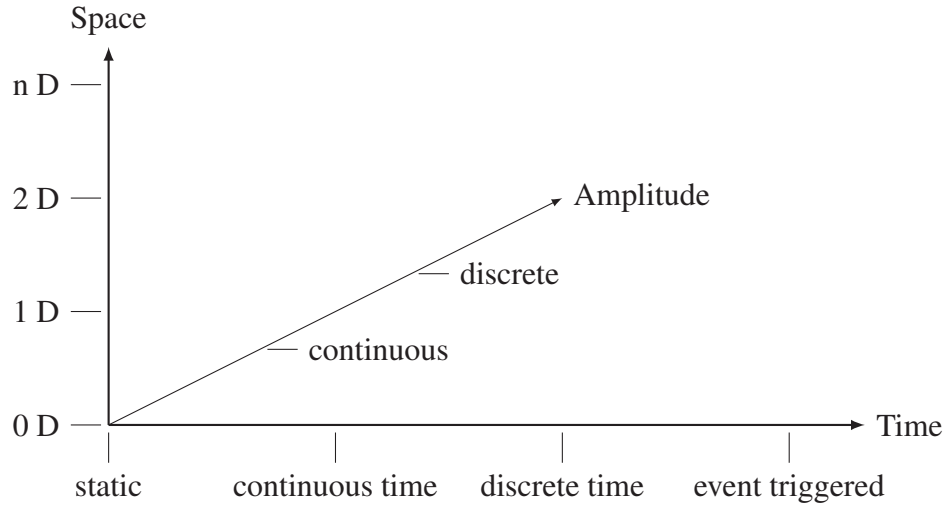
Figure 1.5.: Dimensions of model characteristics

As described before, the input to a system splits into two parts $\mathcal{U}_1, \mathcal{U}_2 \subset \mathcal{U}$ with $\mathcal{U}_1 \cup \mathcal{U}_2 = \mathcal{U}$ and $\mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset$, that is

1. the externally adjustable part $\mathbf{u} \in \mathcal{U}_1$ called control or actuator, and

2. the not influencable part $\mathbf{u} \in \mathcal{U}_2$ termed disturbance.

Since also sensors and actuators are modeled, there may exist unmodeled parts entering the overall scheme via these components. Similarly, any monitoring and control device based on a computer is also subject to idealization, e.g. via using floating point approximations of numbers. Hence, another source of disturbances exists via this component.

Remark 1.8
Formally, each block within Figure 1.4 is a system in the sense of Definition 1.3.

The last two components, which we did not discuss so far, are *monitoring and control* and *operator*. The component monitoring and control is a system representing the inverse of chain actuator – system – sensor, which is additionally modified by the objectives and interacts with operators. Note that similar to the system itself, also the monitoring and control system may exhibit an internal state. This block is typically implemented using computer or other devices such as PLCs. In the literature, the two following descriptions are found most regularly:

> A control is a process in a system, in which one or more variables as input variables influence other variables as output variables due to the laws peculiar to the system.
>
> DIN IEC 60050-351 (2014)

> A control is a process in which a variable, which is to be controlled, is continuously recorded, compared with another variable, the reference variable, and influenced in the sense of an adjustment to the reference variable.
>
> DIN IEC 60050-351 (2014)

The difference between these two descriptions is given by their purpose: While both aim to influence the output, the first one feeds the output from outside the system, but the second one uses an external reference to feed the output back to that reference. For this reason, the first one is called feed forward and the second one feedback. More formally:

Definition 1.9 (Feed forward).
If an input is defined by a function $\mathbf{u} : \mathcal{T} \to \mathcal{U}$, then we call it feed forward.

Definition 1.10 (Feedback).
If an input is defined by a function $\mathbf{u} : \mathcal{X} \to \mathcal{U}$, then we call it state feedback. If it is defined as a function $\mathbf{u} : \mathcal{Y} \to \mathcal{U}$, then it is called output feedback.

Last, the operator represents the human in the loop. The operator interacts with the monitoring and control system via a human-machine-interface (HMI), which provides two key functions for the human, that is information and manipulation. In the literature [11], we find the following description:

> A user interface are all components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system.
>
> ISO 9241 (2019)

After introducing all relevant components within the automation structure of Figure 1.4, we can now look into details regarding the aims of automation.

1.3. Aims

In particular, we need to understand what is meant by automation. In the literature [21], we see the following definition (translated from German):

> Automation is equipping a device so that it works as intended, in whole or in part, without human intervention.
>
> DIN 19233 (1998)

Within this definition, we observe several components:

1. Intend: Each automation requires one or more criteria to assess whether or not it works as intended, see Table 1.1 for derivation of such criteria.

2. Extend: Each automation requires a system boundary, that is which parts of a system or process are automated and which are not. Moreover, it specifically requires modeling of the interfaces between automated and non-automated parts. These can by physical like handovers from machines to storage or workers, or information to and from an operator, cf. Figure 1.4.

3. Attend: Each automation requires components to execute the monitoring and control tasks, that is sensors, actuators, communication and logic, cf. Figure 1.4 and Definitions 1.9, 1.10.

Remark 1.11
Note that extend is not restricted to the machine level, but is more commonly found on the integration, optimization and leading level in Figure 1.1.

Regarding intend, we are going to use the term of a key performance criterion, also called key performance indicator (KPI) throughout the lecture:

Definition 1.12 (Key performance criterion).
A key performance criterion is a function J, which measures defined information retrieved from the system against a standard.

Considering extend, we utilize the concept of constraints. The idea of constraints is to retrieve a set of conditions which must be uphold such that the system must be enabled to continue to exist, i.e. the system must not come into a stage in which it terminates itself. These circumstances are typically modeled via sets:

Definition 1.13 (Constraints).
We call a subset $\mathbb{X} \subset \mathcal{X}$ state constraint set and $\mathbb{U} \subset \mathcal{U}$ control constraint set.

Here, we like to note that each block within Figure 1.4 needs to provide features to ensure safety and security of the overall system for both accidental and malicious mistakes. More precisely:

> Safety is defined as the freedom from unacceptable risk of physical injury.
>
> IEC 61508 (2010)

As such, safety does not mean that a system or process needs to be monitored and control such that no risks exist. Instead, the standard [5] says that any safety-related system must work correctly or fail in a predictable (safe) way, i.e. actions need to be taken such that the probability of a safety-related system satisfactorily performing the required safety functions under all the stated conditions within a stated period of time can be guaranteed.
Security, on the other hand, deals with information security.

> Information security risks relate to the loss of confidentiality, integrity and availability of data within the scope of the information safety management system.
>
> ISO/IEC 27032 (2022)

Similar to safety, also the security standard [10] refers to risk management instead of risk absense and [13] focuses on internet/cyber security.

In the following chapters, we will introduce and discuss the fundamentals of the standard automation structure from Figure 1.4 along the automation path illustrated in Figure 1.1.

Part I.

Planing and specification

CHAPTER 2

SYSTEM PLANNING

Within this chapter, we will focus on the first layer of automation, the planning layer. At this initial level, everything starts with an idea on what shall be realized regarding automation. In the past, the typical mistake at that stage was to think in terms of replacement instead of in terms of renewal (or re-thinking). Most accurately, the following is found to be true:

> The first rule of any technology used in a business is that automation applied to an inefficient operation will magnify inefficiency.
>
> Bill Gates

Indeed, most automation cases realized digitalization in terms of process information, that is to gather information about a running process. So operational data and flow charts are available, which is a prerequisite for automation in the sense of Figure 1.1, but it is not sufficient. Coming back to the overall path of automation, Figure 2.1 displays the content of the present chapter. Here, we will discuss the perspectives and interest place upon planning. To this end, we combine systems/processes to a network, for which certain properties can be analyzed. The importance of these properties may be very different depending on the peer group assessing the network. Here, we first discuss these groups. Thereafter, we shift our focus to modeling concepts and respective description methods. These will serve us a digital model, which we will extend to a digital shadow and digital twin in the upcoming chapters.
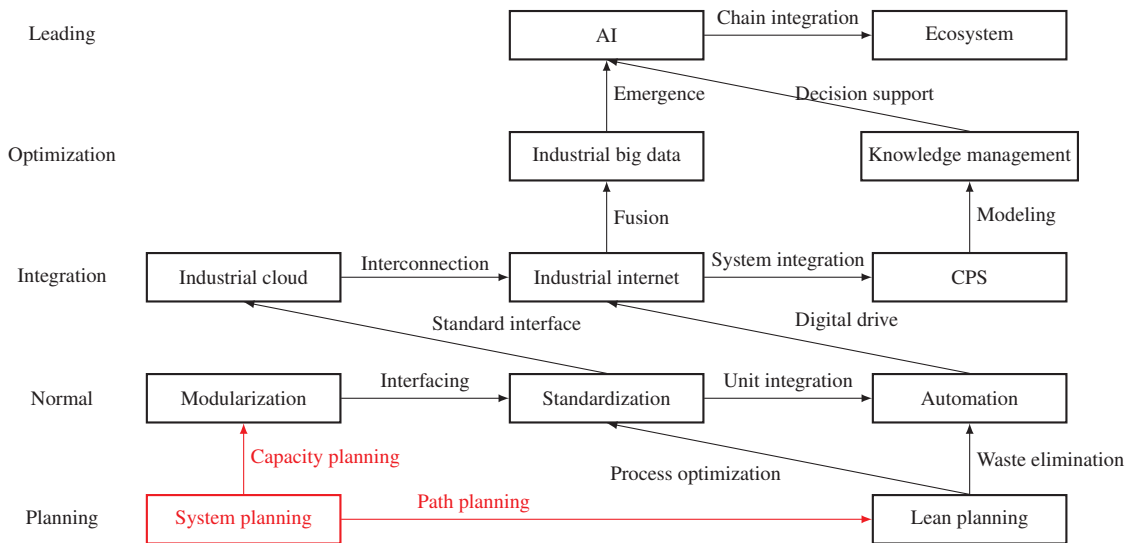
Figure 2.1.: Integrated implementation path of automation

2.1. Interested parties, perspectives and requirements

Automation follows the strategies of cost efficiency or differentiation, cf. Figure 1.2. And in fact, any of the following ideas are not sufficient and in some cases have even shown catastrophic results:

- Automation is not a purpose. Automation is a means to achieve or improve on KPIs, yet it shouldn't be implemented blindly, e.g. because other's introduce it.

- Automation is not an ideology. It should be carefully checked whether the system or process at hand can be improved or if automating a system/process improves connected systems/processes. One solution here is to consider connected systems/processes.

- Automation is not a whitewashing instrument. If KPIs are chosen or designed to improve a system or process which is economically, ecologically or socially unsuitable, it will cause a degradation and magnify inefficiency at the cost of marketing. In such a case, constraints are much more suitable than KPIs for automation.

Having discussed what is not sufficient, we consider necessary input for planning next. Here, we have to distinguish between interested parties and their perspectives. In the common literature, an interested party is most commonly referred to as a stakeholder, yet formally the norm [9] defines the following:

> An interested party can be a stakeholder, person or organization that can affect, be affected by, or perceive itself to be affected by a decision or activity.
>
> ISO 9001 (2015)

Hence, this gives us limits to what needs to be included in modeling a system or process, cf. Definition 1.3. Examples of interested parties can include (but are not limited to):

- suppliers
- customers
- partners
- employees

- investors
- owners
- bankers
- regulatory bodies

- unions
- competitors
- society
- opposing groups

The norm [9] states that one must consider those interested parties, which are relevant for the system/process. For these parties, requirements need to be identified and monitored or reviewed, which are given in [12].

> A requirement is defined as an expression, in the content of a document, that conveys objectively verifiable criteria to be fulfilled and from which no deviation is permitted if conformance with the document is to be claimed.
>
> ISO/IEC Directives 2 (2021)

Hence, the requirements provides us with an design space, which must be guaranteed. Within a model, this will be given by constraints, cf. Definition 1.13. In the trivial case, this design space is empty, i.e. requirements exclude one another. Otherwise, we have a certain degree of freedom for our task. This is exactly the part where perspectives of interested groups need to considered, i.e. where improvement/choice/optimization in terms of a KPI (Definition 1.12) is possible. Here, the norms falls short on a proper definition of what is usually termed *needs and expectations*, where need is used equivalently to requirement whereas expectations is used analogously to perspective. Here, we apply the following:

> An expectation is a strong believe that something will happen or be the case.

The following Table 2.1 provides an overview on the most common expectations.

Table 2.1.: Relevant perspectives and features in automation

Expectation	Feature
Performance and quality	Performance, time/space sampling
Input	Operational area, industrial sector, context, qualifications
Reliability (RAMS)	Dependability, accessibility, maintainability, safety
Physics	Composition, dimensions, interfaces, implementation, materials
Ecology	EM, climate, energy, geometry, stress
Legal	Norms, regulations, conventions, laws, admission
Economic	Life cycle cost, initial/running cost, disposal cost, return on invest

Due to these diverse perspectives, the respective modelers focus on their peer group and quite likely generate diverging models.

2.2. Concepts of modeling

There are several sources for divergence in models, which range from industrial sector specific objects to coding languages up to cultural diversity. Hence, of the overall tasks to be included in concepts of modeling is to provide a holistic view of the information model. The following representation may diverge as it is specific for interested parties.

Modeling itself as cognition method exhibits the components shown in Figure 2.2.

Here, the model can exist on various levels and degrees of detail. An example is given by the robot arm in Figure 2.3, which can be abstracted to its main physical components.

However, these components can be further detailed to satisfy a set of differential equations

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{y}(t) = h(\mathbf{x}(t), \mathbf{u}(t), t),$$
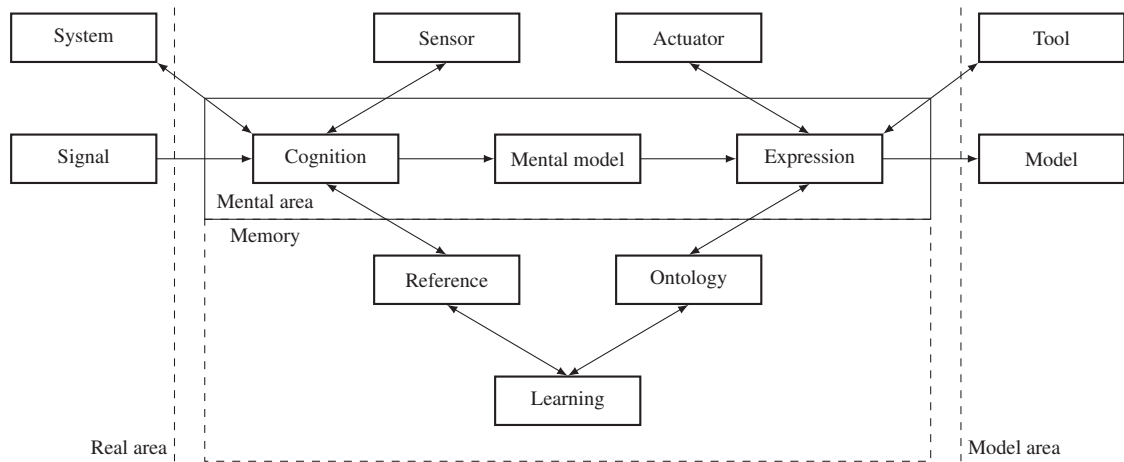
Figure 2.2.: Modeling as cognition method
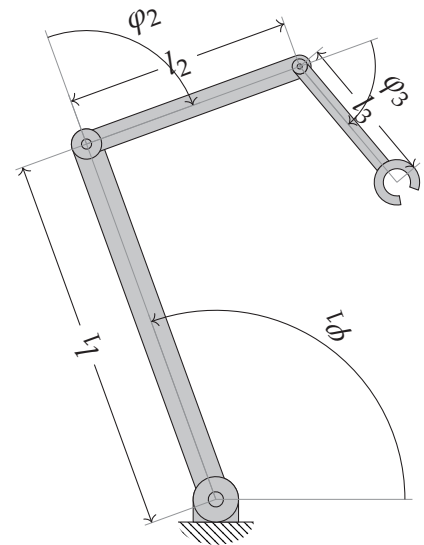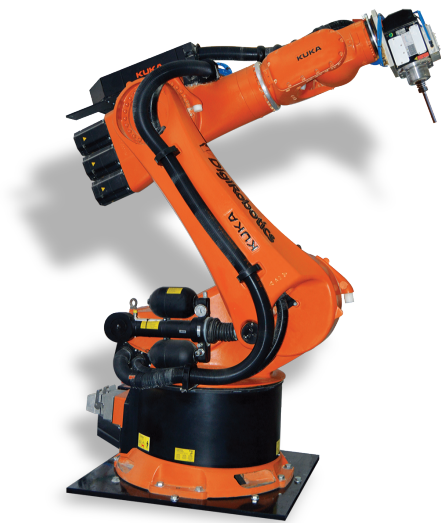
Figure 2.3.: Robot arm in real and model area

which need to be parametrized and identified. The level of detail therefore depends on the purpose of the model and how it shall be used, e.g. in planning.

In today's practice, there are two typical approaches for planning. The first is the typical classical engineering V-model, the second is referred to as agile. Both approaches include modeling in their respective phases. While these approaches differ fundamentally in their phases, they still operate on the same requirements regarding modeling.

Link: For further details on planning approaches, we refer to the lectures *Project management*.

Here, we start of with the process requirements, that is working principles within any modeling process. These requirements are a state-of-the-art list, which is commonly used as a convention and not as a definition.

Convention 2.1 (Process requirements of modeling)

During the modeling process, six principles need to be met:

1. Principle of Correctness: A model needs to present the facts correctly regarding structure and dynamics (semantics). Specific notation rules have to be considered (syntax).

2. Principle of Relevance: All relevant items have to be modeled. Non-relevant items have to be left out, i.e. the value of the model doesn't decline if these items are removed.

3. Principle of Cost vs. Benefit: The amount of effort to gather the data and produce the model must be balanced against the expected benefit.

4. Principle of Clarity: The model must be understandable and usable. The required knowledge for understanding the model should be as low as possible.

5. Principle of Comparability: A common approach to modeling ensures future comparability of different models that have been created independently from each other.

6. Principle of Systematic Structure: Models produced in different views should be capable of integration. Interfaces need to be designed to ensure interoperability.

Similarly, any concept of modeling should satisfy the following functional requirements:

Convention 2.2 (Functional requirements of modeling)

For any concept of modeling, the fitness of methods, tools and implementation need to be aligned.

1. Fitness for methods:

   - Consideration of modern software development methods
   - Consideration of development phase
   - Analytical/mathematical properties
   - Theoretical soundness and provability
   - Vertical and horizontal consistency
   - Composability and decomposability
   - Consideration of deterministic and stochastic properties

---

- ■ Graphical presentability
- ■ Ability for simulation
- ■ Testability, traceability and comprehensibility

2. Fitness for tools
   - ■ Portability
   - ■ Compatibility
   - ■ Usability

3. Fitness for implementation
   - ■ Viability in soft- and hardware
   - ■ Reverse engineering

Based on the latter, we now outline different description methods to approach the goal of a digital model.

2.3. Description methods

As indicated in the previous section, we need to avoid divergence of models at least on the layer of information. Depending on the purpose of the description, a variety of methods are applied. In the following Table 2.2, we structure these methods according to their usage. The sequence follows a top down idea, which is the typical sequence in deriving automation of a process.

Table 2.2.: List of description methods

Class	Example
Abstraction oriented	Verbal description, algebra, proposition logic, predicate logic
Structure oriented	Sequential logic system, combinatorial logic
Implementation oriented	Logic plan, function plan, contact diagram, structure diagram, timing diagram, instruction list, gantt chart
State oriented	Decision table, transition table, state diagram, state graph, Karnaugh-Veitch diagram
	Continued on next page

Table 2.2 – continued from previous page

Class	Example
Technology oriented	Flow chart, switching plan, computer aided design (CAD)
Method oriented	Network diagram, Nassi-Shneidermann diagram, unified modeling language (UML), structure-analysis-real-time (SA/RT) diagram
Mathematical	Boolean algebra, differential/difference equations, Markov chains

Note that the methods cannot be considered to stand by themselves, they require specification top down and connection bottum up. One of the fundamentals, which can be found in each class of Table 2.2, are networks, objects and ontologies. Within this lecture, we utilize networks and objects for our models. Both approaches are suitable for modern computer science concepts such as object orientation, allow for mathematical concepts such as discrete event triggered dynamical systems and integrate both compact description as well as horizontal and hierarchical structures, cf. Figure 1.5. More formally, we define the following.

Definition 2.3 (Object).
A set of objects $\mathcal{O} = \mathcal{A} \cup \mathcal{M}$ consists of definable parts called attributes $\mathcal{A}$, and reasoning parts called methods $\mathcal{M}$.

Remark 2.4
Note that Definition 2.3 does not require all parts or methods to be defined. This is in accordance with our concept of a model, for which certain parts are modeled and the remainder is considered as disturbance.

An example of an object is given by Figure 2.4, where we could define the three attributes robot arm, conveyor and package, and a possible method could be putting a package on the conveyor.

2.4. Petri Networks

In the general context of automation, an object may range from a parameter in a program to the program itself, the machine controlled by that program, a screw within the machine or up to the entire supply chain of a company and its surround economy.
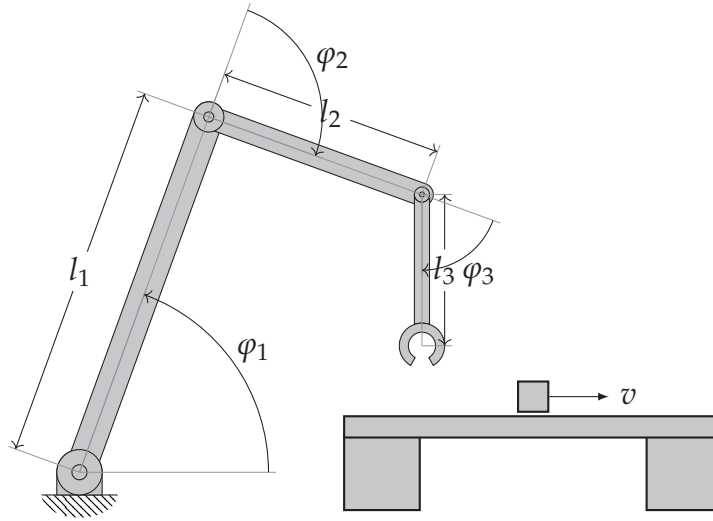
Figure 2.4.: Sketch of a robot arm

In computer science, one major feature indicated by this description is the so called inheritance. We distinguish between aggregation (one object consists of . . .) and abstraction (one object is a . . .). Moreover, elements of objects may be connected, which leads us to the definition of a network.

Definition 2.5 (Network).
Let $\mathcal{O} = \mathcal{A} \cup \mathcal{M}$ be a set of objects and $\mathcal{E} \subset \mathcal{A} \times \mathcal{M} \cup \mathcal{M} \times \mathcal{A}$ be a set of pairs of objects. Then we call $\mathcal{E}$ the set of edges and the tuple $\mathcal{N} = (\mathcal{O}, \mathcal{E})$ a network.
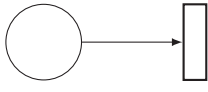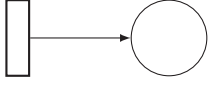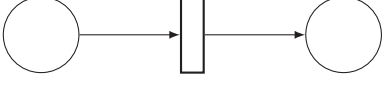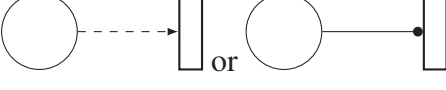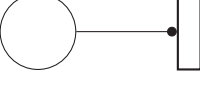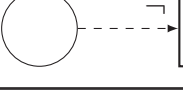
Remark 2.6
In the computer science or mathematics literature, a network is also called a graph, for which the set of objects is typically referred to as set of nodes.
In the process automation literature, attributes are also called places indicating the physical position of an object. Methods, on the other hand, are also called transitions, i.e. transportation from start to destination or modifications from initial to target property.

One such network, which can be used as a description method, is the so called Petri-Net. Within a Petri-Net, attributes are represented by circles, methods by blocks and edges by arrows. Moreover, the denomination extended in Table 2.3 is used.

Table 2.3.: List of Petri-Net symbols

Symbol	Meaning
	Attribute
	Method
	Edge
	Pre-edge
	Post-edge
	Relation
	Communication/test edge
	Inhibition edge

In this context, a pre-edge is a requirement for a method, while a post-edge is the result of a method. In Table 2.3, we drew both pre-edge and post-edge as a 1-1 connection, yet 1-n connections are possible as well.

Algebraically, the network or graph resulting from using the network notion stated above can be summarized in the so called incidence matrix.

Definition 2.7 (Incidence matrix).
For any network $\mathcal{N} = (\mathcal{O}, \mathcal{E})$, we call

$$\mathcal{I} = [\mathcal{I}_{jk}] \qquad \text{where } \mathcal{I}_{jk} := \begin{cases} 1 & \text{there exists an edge connecting } m_j \in \mathcal{M} \text{ and } a_k \in \mathcal{A} \\ -1 & \text{there exists an edge connecting } a_j \in \mathcal{A} \text{ and } m_k \in \mathcal{M} \\ 0 & \text{else} \end{cases}$$

$$(2.2)$$

incidence matrix of the network.

The incidence matrix not only provides us with a compact description of the network, it can also be used for computations. It can be used to identify, e.g., whether certain attributes are necessary/sufficient or may ever be reached. Similarly, if a method is a risk avoidance, it can be checked whether the risk case may ever occur.

Remark 2.8

In the context of an FMEA (failure mode and effects analysis), it is a common question to assess if and with what probability a risk case may occur. For such questions, a Petri-Net model can provide a direct answer.

Central for assessments are respective configurations. In principle, these configuration are nothing else than use cases of a system or process, for which we like to answer certain questions.

Definition 2.9 (Configuration).
Consider a network $\mathcal{N} = (\mathcal{O}, \mathcal{E})$ with $\mathcal{O} = \mathcal{A} \cup \mathcal{M}$. Then any subset $\mathcal{C} \subseteq \mathcal{A}$ is called a configuration. We call the tuple $(\mathcal{N}, \mathcal{C})$ an elementary network.

Hence, a configuration is a subnet within a network. As such, it interacts with the rest of the network, yet we are only interested in answers for this specific subset.

Remark 2.10

Loosely speaking, if the entire world would be represented as a network, then a configuration is a model of a process/system, which interacts with its surroundings and is disturbed by it.

Before we can properly formulate the mentioned questions, we first need some more technical detail. For a Petri-Net, we extend the notion of a configuration by introducing markings and multiplicities. Markings can be interpreted as units assigned to an attribute, like motors waiting to be installed into a powertrain. Multiplicities are the required number of units for a method, i.e. now many motors are required to integrate a powertrain.

Definition 2.11 (Petri-Net).
Consider a network $\mathcal{N} = (\mathcal{O}, \mathcal{E})$ with $\mathcal{O} = \mathcal{A} \cup \mathcal{M}$. Moreover, let

$$\mathcal{C}_{\mathcal{A}} : \mathcal{A} \to \mathbb{N}_0^{\mathcal{A}}, \qquad \mathcal{C}_{\mathcal{A}}(a) = \#\left(a \in \mathcal{C}_{\mathcal{A}}(a)\right) \quad \forall a \in \mathcal{A} \tag{2.3}$$

$$\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{N}_0^{\mathcal{E}}, \qquad \mathcal{C}_{\mathcal{E}}(e) = \#\left(e \in \mathcal{C}_{\mathcal{E}}(e)\right) \quad \forall e \in \mathcal{E} \tag{2.4}$$

be multisets. Then the triple $(\mathcal{N}, \mathcal{C}_{\mathcal{A}}, \mathcal{C}_{\mathcal{E}})$ is called a Petri-Net. The maps $\mathcal{C}_{\mathcal{A}}$ and $\mathcal{C}_{\mathcal{E}}$ are called marking and multiplicity.

Hence, markings can be considered as storage, whereas multiplicities represent performance requirements for methods/tasks. Here, we already obtain the first interesting questions:

- reachability: Can all markings be set? Which markings can be set?

- coverability: Can specific markings be set?

Continuing, we are interested in how the system/process evolves over time, and what the requirements for such a flow are. This leads us to the so called flow relation.

Definition 2.12 (Flow relation).
Given a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$. Then we call the set

$$\mathcal{F} := \{e \in \mathcal{E} \mid \mathcal{C}_\mathcal{E}(e) > 0\} \tag{2.5}$$

a flow relation.

The flow relation says that certain values of attributes are required in order for edges to be executable. This points into the direction of the question

- liveness: Is a process/system deadlock free, can all attributes be marked and unmarked?

- consistency: Will all markings be set uniquely?

- boundedness: Will all markings stay bounded?

Regarding these questions, special attention must be paid to inhibitor edges. In this context, a flow is inhibited if the inhibited attribute is marked. This directly leads us to the requirements and results of methods.

Definition 2.13 (Preset and postset).
Consider a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ with flow relation $\mathcal{F}$. For any edge $e \in \mathcal{F}$, we call the set

$$\bullet m := \{a \in \mathcal{A} \mid \mathcal{C}_\mathcal{E}(a, m) > 0\} \tag{2.6}$$

preset of a method $m \in \mathcal{M}$ and the set

$$m\bullet := \{a \in \mathcal{A} \mid \mathcal{C}_\mathcal{E}(m, a) > 0\} \tag{2.7}$$

postset of a method $m \in \mathcal{M}$.

Preset and postset can be interpreted as input and output of a method, cf. Figure 1.3 where the system corresponds to a method block in the Petri-Net.

Remark 2.14

As stated before, inhibitor edges are somewhat different. These edges enter the set definition (2.6) with a negation, which can be simplified to $C_{\mathcal{E}}(a, m) = 0$ if $e = (a, m)$ is an inhibitor edge.

Following the input/output idea, a method can only be executed if all requirements are satisfied. This allows us to check whether or not a method will ever by applied (e.g., in an FMEA). Apart from that, the presets identify the required chain of methods/attributes, which are elementary for application of the method under consideration. In the context of Petri-Nets, application is typically referred to as firing.

Definition 2.15 (Enabling and firing).
Given a Petri-Net $(\mathcal{N}, C_{\mathcal{A}}, C_{\mathcal{E}})$. Then a method $m \in \mathcal{M}$ is enabled if

$$\forall a \in \bullet m : \ C_{\mathcal{A}}(a) \geq C_{\mathcal{E}}(a, m). \tag{2.8}$$

Firing a method $m \in \mathcal{M}$ is possible if m is enabled and leads to updating the markings according to

$$C_{\mathcal{A}}(a) = \begin{cases} C_{\mathcal{A}}(a) - C_{\mathcal{E}}(a, m), & \forall a \in \bullet m \\ C_{\mathcal{A}}(a) + C_{\mathcal{E}}(m, a), & \forall a \in m \bullet \\ C_{\mathcal{A}}(a), & \text{else.} \end{cases} \tag{2.9}$$

Remark 2.16

Sometimes a method is called enabled if additionally to (2.8) the postset is clear, i.e.

$$\forall a \in m \bullet : \ C_{\mathcal{A}}(a) = 0.$$

What makes Petri-Nets particularly interesting is the fact that we can check what is going to happen when methods may continually fire in arbitrary order. Note that at no point, we mentioned anything about time, sequence, order etc. Hence, the stated questions can be analyzed without knowledge of time. This holds particularly true for paths, circuits and deadlocks.

Definition 2.17 (Path and circuit).

Consider a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ with flow relation $\mathcal{F}$. Then we call a sequence $a_1, \ldots, a_n \in \mathcal{A}$ a path if the sequence is finite, nonempty and the respective edges are elements of the flow relation $\mathcal{F}$.

A path is called a circuit if $a_1 = a_n$ and additionally from $a_j = a_k$ we have $j = k$ for all $1 < j, k < n$.

Paths and circuits can directly be obtained by the incidence matrix for Petri-Nets. Here, it is common to separate between the preset and postset incidence matrix. The preset one actually allows to see circuits, which may get lost if the complete incidence matrix is computed.

Definition 2.18 (Petri-Net incidence matrix).

For any network Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$, we call $\mathcal{I} = \mathcal{I}^- + \mathcal{I}^+$ with

$$\mathcal{I}^- = [\mathcal{I}_{jk}] \qquad \text{where } \mathcal{I}_{jk} := \begin{cases} -\mathcal{C}_\mathcal{E}(a_j, m_k) & a_j \in \bullet m_k \\ 0 & \text{else} \end{cases} \tag{2.10}$$

$$\mathcal{I}^+ = [\mathcal{I}_{jk}] \qquad \text{where } \mathcal{I}_{jk} := \begin{cases} \mathcal{C}_\mathcal{E}(m_k, a_j) & a_j \in m_k \bullet \\ 0 & \text{else} \end{cases} \tag{2.11}$$

incidence matrix of the Petri-Net.

The incidence matrix is a central tool for computing properties of networks and initial markings. To qualify such a network, we adapt the definition of a key performance indicator (Definition 1.12) to the network setting.

Definition 2.19 (Cost function for networks).

Consider a network $(\mathcal{O}, \mathcal{E})$. Then we call $J : \mathcal{I} \times \mathcal{I}^2 \to \mathbb{R}_0^+$ cost function.

While Definition 2.19 is generic, the following indicators specify the idea of qualifying a network in practical terms.

2.4.1. Reachability and coverability

Regarding our question on uniqueness, it is already obvious that the resulting marking depends on the sequence of firing. Hence, uniqueness cannot be expected. Still for the properties reachability and coverability, we can use the concept of markings to derive the following answer:

Algorithm 2.20 (Coverability and reachability)

Consider a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ together with an initial marking $c_0 \in \mathbb{N}_0^\mathcal{A}$ and a terminal marking $c \in \mathbb{N}_0^\mathcal{A}$ to be given.

1. For each $a \in \mathcal{A}$ with $c_0(a) > 0$

    Set $\mathcal{R}(c_0) := \mathcal{R}(c_0) \cup \{a\}$.

2. Do

    a) For each method $m \in \mathcal{M}$

        i. If m is fireable (2.8)

            A. Update marking $\mathcal{C}_\mathcal{A}$ according to (2.9)

            B. Set $\mathcal{R}(c_0) := \mathcal{R}(c_0) \cup \{a\}$ for all $a \in m\bullet$

    b) If $\mathcal{C}_\mathcal{A} \geq c$, then c is coverable and stop.

    while there existed a method m, which was fireable

3. If $\mathcal{C}_\mathcal{A} \ngeq c$, then c is not coverable.

Within Algorithm 2.20, the set $\mathcal{R}(c_0)$ refers to those attributes, which can be set given the initial marking c_0.

Definition 2.21 (Reachability set).

For a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ with given initial marking $c_0 \in \mathbb{N}_0^\mathcal{A}$ we call $\mathcal{R}$ reachability set of the Petri-Net if it is maximal and for each attribute $a \in \mathcal{R}$ there exists a path within the flow relation $\mathcal{F}$. We call $\mathcal{R}(c_0)$ reachability set of the initial marking if it is the maximal set of attributes a such that there exists a path within the flow relation $\mathcal{F}$ and each method along this path is enabled.

Remark 2.22

Note that the reachability set depends on the initial marking. Moreover, not the entire coverability search must be executed to check if a specific attribute can be reached.

A short way to compute whether or not a marking is coverable is given by the incidence matrix.

Theorem 2.23 (Coverability).

Consider a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ together with an initial marking $c_0 \in \mathbb{N}_0^\mathcal{A}$ and a terminal marking $c \in \mathbb{N}_0^\mathcal{A}$ to be given. Then c is coverable iff

$$\exists x \in \mathbb{N}_0^\mathcal{E} : \ c = c_0 + \mathcal{I} \cdot x. \tag{2.12}$$

From equation (2.12) we directly obtain that a marking is coverable if

$$x = \mathcal{I}^{-1} (c - c_0)$$

is solvable. The result corresponds a sequence of methods $m \in \mathcal{M}$ which need to be fired resulting in a path of intermediate attributes.

To further answer the question which attributes can be set, i.e. what is reachability set of a initial marking, we directly obtain that it is identical to the maximal coverable set. Hence, we have the following:

Theorem 2.24 (Reachability).

For a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ and an initial marking $c_0 \in \mathbb{N}_0^\mathcal{A}$, the reachability set is given by

$$\max_{c \in \mathbb{N}_0^\mathcal{A}} c : \ \exists x \in \mathbb{N}_0^\mathcal{E} : \ x = \mathcal{I}^{-1} (c - c_0). \tag{2.13}$$

2.4.2. Liveness

Having answered the questions regarding reachability and coverability, we next address liveness.

Definition 2.25 (Liveness).

Consider a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ and an initial marking $c_0 \in \mathbb{N}_0^\mathcal{A}$. Then we call a method $m \in \mathcal{M}$

- dead if no path exists such that m can be fired,

- quasi-live if m can be fired at least once in a path,

- live if it is quasi-live for every marking $c \subset \mathcal{R}(c_0)$.

While the first case is self-speaking, the other two cases show a fundamental difference. While quasi-live means that the method may be executed but is potentially not executed, live means that

the Petri-Net is deadlock free independent from the choice of the path/firing sequence. The latter requires that the following two cases can at least be avoided such that the system does not get stuck:

Theorem 2.26 (Liveness conflicts).
Consider a Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$. *Then the network is live if no directional conflicts exists, that is*

$$\bullet m_1 \cap \bullet m_2 = \emptyset, \tag{2.14}$$

$$m_1 \bullet \cap m_2 \bullet = \emptyset. \tag{2.15}$$

Remark 2.27
Note that conditions (2.14), (2.15) *are necessary only.*

A conflict in predecessor attributes, i.e. violation of (2.14), may exist if an attribute is a requirement for more than one method, yet its marking is to low to satisfy the multiplicities of the acquiring methods. An example is a working part, which is necessary for different workshops, or a tire which can be mounted to different cars.

Similarly, a conflict in successor attributes, that is violation of (2.15), may exist if firing one method results in blocking an attribute for another method. An example is a storage area filled by two lines. If the output of one line causes the limit of the storage area to be reached, then the output of the second line cannot be processed.

2.4.3. Safeness

Last, we address boundedness. The idea of boundedness is to guarantee that a network is safe, i.e. there will be no overflows independent from the chosen firing sequence. For ease of exposition, we consider the most simple case of 1-boundedness here:

Definition 2.28 (Safeness / 1-boundedness).
A Petri-Net $(\mathcal{N}, \mathcal{C}_\mathcal{A}, \mathcal{C}_\mathcal{E})$ with initial marking $c_0 \in \mathbb{N}_0^\mathcal{A}$ is 1-bounded or safe if

$$\forall c \in \mathcal{R}(c_0) : \; c(a) \leq 1 \; \forall a \in \mathcal{A}. \tag{2.16}$$

Basically, this definition says that each attribute may at most have one marking. We then directly obtain the following:

Theorem 2.29 (Sufficient conditions for a safeness).
A Petri-Net $(\mathcal{N}, \mathcal{C}_{\mathcal{A}}, \mathcal{C}_{\mathcal{E}})$ is safe iff c_0 has at most one unit element. A live network is safe iff c_0 has exactly one unit element.

While Theorem 2.29 states sufficient conditions, also necessary conditions are available:

Theorem 2.30 (Necessary conditions for safeness).
Given a Petri-Net $(\mathcal{N}, \mathcal{C}_{\mathcal{A}}, \mathcal{C}_{\mathcal{E}})$, there exists a safe and live marking iff $\mathcal{N}$ is strongly connected.

Safeness is one of the major objectives, for which Petri-Nets are used in practical applications. Yet, respective algorithms require an indeep connection between the safety property and the incidence matrix, which is outside the scope of this lecture. For further details, we refer to the work of Murata [18].

So far, we have not considered time in our modeling. Even the firing sequence used within Petri-Nets was neither fixed nor connected to time. Yet, time will play a vital role in the remainder of the lecture. We will put special emphasis on the real-time requirement of systems, that is computations shall be fast enough to not cause any delay and avoid unused capacities in the execution of the system/process.

CHAPTER 3

SEPARATION

In the past chapter, we discussed networks to represent systems and processes. Networks are a very general tool to indicate connection and evaluate systemic properties such as reachability, liveness and safeness. In the setting used here, it is particularly difficult to add time. While the latter can be done, it may not be in the best interest of the user to do so as the description becomes more complex. Instead, the idea we discuss in this chapter is to further reduce complexity via a divide–and–conquer strategy. Reconsidering our overall path of automation, Figure 3.1 indicates the focus of the present chapter.
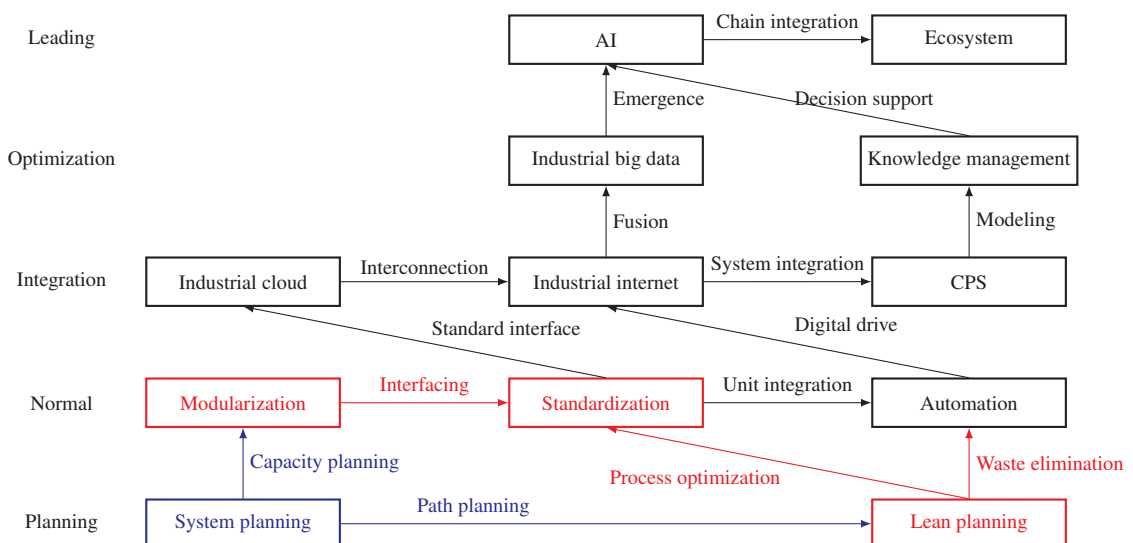
Figure 3.1.: Integrated implementation path of automation

To this end, we first discuss modularization, i.e. how to specifically cut down the system/process into smaller ones. In a second step, we consider the interfaces between these modules and how

these can be standardized. Note that an interface can be with respect to information, energy, goods and people. Both standardization and modularization need to be considered from a global perspective to generate efficiency and avoid waste. These aims stand at the core of lean planning, which we discuss at the end of this chapter.

3.1. Modularization

As outlined, the aim of modularization is to cut the system/process into smaller parts. We will use the network description derived in the previous chapter to identify objects, which are *similar* to one another. In network theory, these are also called clusters/communities and similarity is based on topology, i.e. connectedness.
Modularization bears a variety of advantages:

- Complexity: If only a module needs to be treated, the overall system complexity can be reduced drastically resulting in lower cost and faster development cycles.

- Replacement: If modules can be exchanged, cross usage (like cross products) can be applied to assure quality or accelerate technical iteration/innovation.

- Diversity: As modules may be substituted, this leads to divers/customizable solutions.

However, compared with integrated architecture, on a management level modular architectures often lack overall coordination and performance optimization. Hence, its weak link is often found in the interface between modules.
In order to modularize a system/process, we first introduce the concept of a cluster, which is basically an intersection free coverage of the network by subsets of it. More formally:

Definition 3.1 (Clustering/Modularization).
Consider a network $(\mathcal{O}, \mathcal{E})$. Then we call $S = \{S_1, \ldots, S_{\#S}\}$ a clustering or modularization if

$$S_j \neq \emptyset \; \forall j \in \{1, \ldots, \#S\}$$
$$S_j \cap S_k = \emptyset \; \forall j, k \in \{1, \ldots, \#S\} \tag{3.1}$$
$$\bigcup_{j=1}^{\#S} \mathcal{O}_j = \mathcal{O}.$$

In order to derive clusters, there are several measures of quality of a cluster, which will be our key performance criterion, recall Definition 1.12. Here, we focus on the main feature of modularity.

Definition 3.2 (Modularity measure).
Given a network $(\mathcal{O}, \mathcal{E})$ let S be a clustering of it. Then we call

$$J_{\text{Modularity}}(S) := \sum_{j=1}^{\#S} \left(P(S_j \mid S_j) - P(S_j \mid \mathcal{O})^2 \right) \qquad (3.2)$$

the modularity of a clustering S where $P(S_j \mid S_j)$ denotes the probability of intra-cluster edges in cluster S_j, that is

$$P(S_j \mid S_j) := \frac{\# \left\{ (e_1, e_2) \mid e_1 \in S_j, e_2 \in S_j \right\}}{\#\mathcal{E}} \qquad (3.3)$$

and $P(S_j \mid \mathcal{O})$ the probability of either an intra-cluster edge in cluster S_j or of an inter-cluster edge incident on cluster S_j, i.e.

$$P(S_j \mid \mathcal{O}) := \frac{\# \left\{ (e_1, e_2) \mid e_1 \in S_j, e_2 \in \mathcal{O} \right\}}{\#\mathcal{E}} \qquad (3.4)$$

Modularity is a value (or score), which can be used to assess a clustering. In terms of modularity, we are interested to identify a clustering S such that the links between clusters are minimized, that is each cluster S_j is more or less self-contained. The idea behind self-containedness is to allow us to treat such a cluster as standalone or *embedded*, i.e. to treat it separately from the remaining network. Hence, a clustering S is considered to be good if the value of a given metric $J(S)$ is maximized (and at best 1).

Task 3.3

Consider the network given in Figure 3.2. Compute modularity of the clustering $S = \{\{a, b, c\}, \{d, e, f, g\}, \{h, i\}\}$.
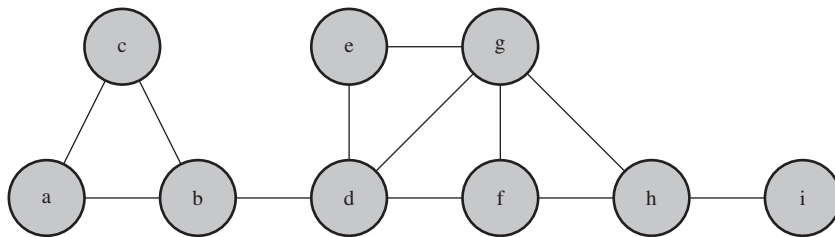
Figure 3.2.: Cluster of a network

Solution to Task 3.3: We obtain

$$J_{\text{Modularity}}(S) = \left(\frac{6}{24} - \left(\frac{7}{24}\right)^2\right) + \left(\frac{10}{24} - \left(\frac{13}{24}\right)^2\right) + \left(\frac{2}{24} - \left(\frac{4}{24}\right)^2\right) \approx 0.34.$$

A different measure is the so called conductance, which is a measure to check whether a network has a bottleneck.

Definition 3.4 (Conductance).

Given a network $(\mathcal{O}, \mathcal{E})$ let S be a clustering of it. Then we call

$$J_{\text{Conductance}}(S_j) := \frac{Q(S_j, \overline{S}_j)}{\min\{Q(S_j, \mathcal{O}), Q(\overline{S}_j, \mathcal{O})\}} \tag{3.5}$$

conductance of a cluster S_j where $\overline{S}_j = \mathcal{O} \setminus S_j$ and

$$Q(S_j, \overline{S}_j) = \#\left\{(e_1, e_2) \mid e_1 \in S_j, e_2 \in \overline{S}_j\right\}$$

denotes the quantity of edges leaving the cluster S_j,

$$Q(S_j, \mathcal{O}) = \#\left\{(e_1, e_2) \mid e_1 \in S_j, e_2 \in \mathcal{O}\right\}$$

denotes the quantity of edges within and leaving the cluster S_j, and

$$Q(\overline{S}_j, \mathcal{O}) = \#\left\{(e_1, e_2) \mid e_1 \in \overline{S}_j, e_2 \in \mathcal{O}\right\}$$

denotes the quantity of edges outside and entering the cluster S_j. Moreover, we call

$$J_{\text{Conductance}}(S) := 1 - \frac{1}{\#S} \sum_{j=1}^{\#S} J_{\text{Conductance}}(S_j) \tag{3.6}$$

conductance of a clustering S.

The idea of detecting bottlenecks is totally different from separability/embeddedness. Here, we are interested to identify those cuts, which put a limit on the flow of goods/information/people/energy within the network. To similarly being able to assess the network in terms of required flow, conductance is inverted in Equation (3.6), again resulting in a maximization problem with aim 1.

Task 3.5

Reonsider the network given in Figure 3.2. Compute conductance of the clustering $S = \{\{a,b,c\},\{d,e,f,g\},\{h,i\}\}$.

Solution to Task 3.5: For $S_1 = \{a,b,c\}$, $S_2 = \{d,e,f,g\}$, $S_3 = \{h,i\}$, we obtain

$$J_{\text{Conductance}}(S_1) = \frac{1}{7}, \qquad J_{\text{Conductance}}(S_2) = \frac{3}{11}, \qquad J_{\text{Conductance}}(S_3) = \frac{2}{4}$$

and

$$J_{\text{Conductance}}(S) = 1 - \frac{1}{3} \cdot \left(\frac{1}{7} + \frac{3}{11} + \frac{2}{4}\right) \approx 0.69.$$

Last, one also typically considers coverage of a network as a marker for modularization. Here, coverage is different from coverability in Chapter 2, where we considered that certain attributes can be set. Coverage describes the property of subsets within a network.

Definition 3.6 (Coverage).
Given a network $\mathcal{N} = (\mathcal{O}, \mathcal{E})$ and a clustering S of it, the quotient

$$J_{\text{Coverage}}(S) := \frac{\sum\limits_{j=1}^{\#S} Q(S_j, S_j)}{\#\mathcal{E}} \tag{3.7}$$

is called coverage of a cluster S_j where

$$Q(S_j, S_j) = \#\left\{(e_1, e_2) \mid e_1 \in S_j, e_2 \in S_j\right\}$$

denotes the quantity of edges inside of cluster S_j.

As such, coverage tell us which percentage of edges are contained within the clustering, and which percentage are neglected. Hence, the aim is to maximize this number to 1. Yet over-optimization of this metric is not useful, the optimal result in that case is simply the cluster to be identical to the set of objects $\mathcal{O}$.

Task 3.7

Reonsider the network given in Figure 3.2. Compute coverage of the clustering $S =$
$\{\{a,b,c\},\{d,e,f,g\},\{h,i\}\}$.

Solution to Task 3.7: We obtain

$$J_{\text{Conductance}}(S) = \frac{6+10+2}{24} = 0.75.$$

Remark 3.8

In the literature, additional metrics like information recovery or normalized mutual information are considered as well, which are outside the scope of this lecture, cf., e.g., [2].

One method to derive a clustering, which (approximately) maximizes any of the above metrics, is the so called Louvain algrithm. The idea of this method is to approximate a maximum by sequentially checking whether a metric is improved by removing/inserting objects from one cluster into another. As this is a greedy approach, the solution will only be locally optimal [1].

Algorithm 3.9 (Louvain algorithm for clustering)
Consider a network $(\mathcal{O},\mathcal{E})$.

1. For $j = 1,\ldots,\#\mathcal{O}$ set $S_j = \{\mathcal{O}_j\}$

2. Compute metric (modularity (3.2) or conductance (3.6) or coverage (3.7))

3. Do

    a) For $j = 1,\ldots,\#\mathcal{O}$

        i. For each edge (j,k) or (k,j) of object $\mathcal{O}_j$ with $k \in \mathcal{O}_k$, $\mathcal{O}_j \neq \mathcal{O}_k$

            ■ Remove $\mathcal{O}_j$ from its cluster and add it to cluster of $\mathcal{O}_k$

            ■ Recompute metric

            ■ If metric not improved, revert move of $\mathcal{O}_j$

        ii. Remove empty clusters

    while metric is improved

Having a possibility to cluster a system/process at hand, the question is which clustering is appropriate regarding strategy and goals of the system/process.

3.2. Standardization

As we have seen in Table 1.1, there are two main strategies, cost leadership and differentiation. Regarding cost leadership, standardization aims to develop components, subassemblies, processes, structures, types, units, goods, services or programs which can be interchanged. Reconsidering the previous Section 3.1, this corresponds to applying modularity. In a network sense, it is the opposite of customization, which is the aim of the differentiation strategy and corresponds to coverage.

> **Remark 3.10**
> *Note that in the context of customization, alternatives are treated as different set of sub-networks, which all need to be present in the network.*

Since standardization can only provide a cost advantage if options are dropped, both strategies work against one another.

To derive a meaningful standardization, an essential component is the interface definition of modules. In order for modules to be used flexibly, the interfaces should be as independent as possible of special customer requirements and the respective module. The standardization procedure typically consists of the three steps shown in Figure 3.3.
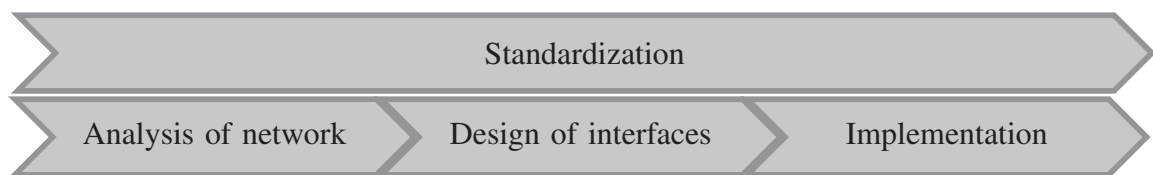
Figure 3.3.: Structure of the standardization process

In the previous Section 3.1 we discussed ideas and methods how to modularize a network. In this section, we discuss the proper integration of the criteria and requirements put to a modularization to be optimized/satisfied, which is termed standardization.

> **Definition 3.11** (Standardization).
> Consider a network $(\mathcal{O}, \mathcal{E})$. We call a clustering S a standardization if
>
> - S is optimal with respect to a defined metric $J(\cdot)$, and
>
> - S satisfies all constraints defined by requirements.

Task 3.12

Reconsider the network from Figure 3.2. Suppose we require object f to be held singular in one cluster. Plot the standardization by adapting the clustering S from Task 3.3 respectively.

Solution to Task 3.12: Results are given in Figure 3.4 with

$$J_{\text{Modularity}}(S) = \left(\frac{6}{24} - \left(\frac{7}{24} \right)^2 \right) + \left(\frac{6}{24} - \left(\frac{10}{24} \right)^2 \right)$$
$$+ \left(\frac{0}{24} - \left(\frac{3}{24} \right)^2 \right) + \left(\frac{2}{24} - \left(\frac{4}{24} \right)^2 \right) \approx 0.24$$
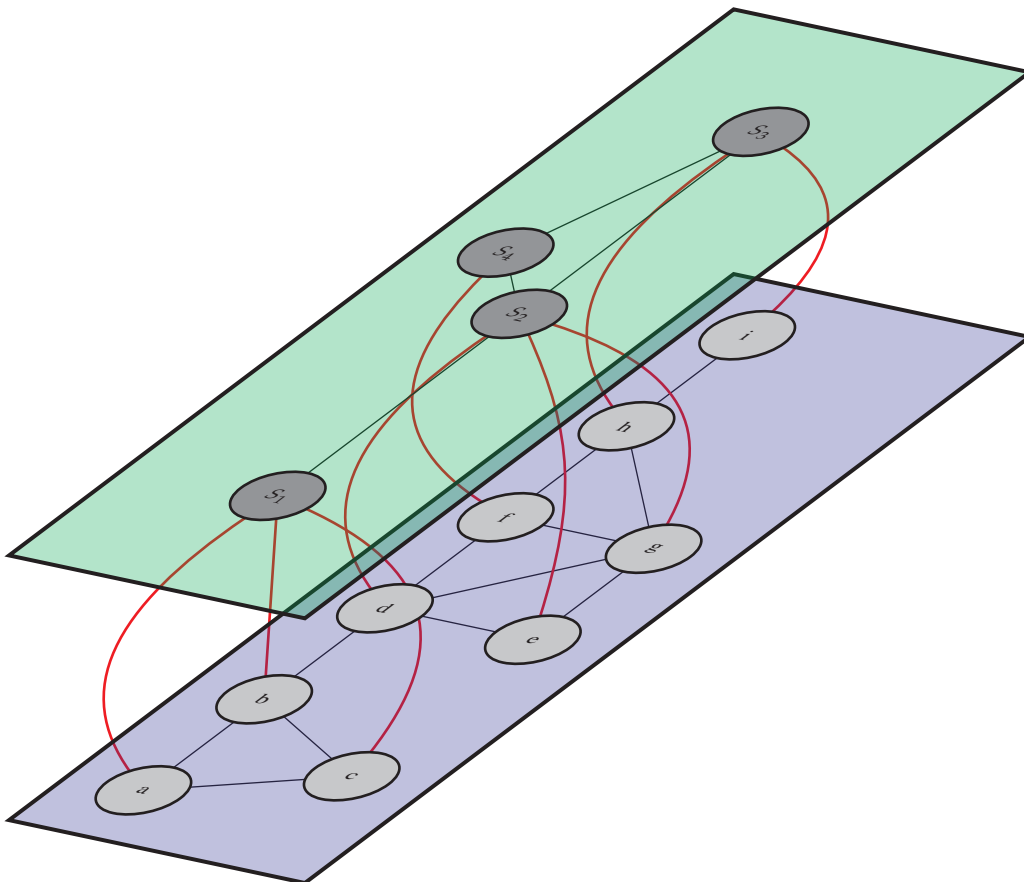
Figure 3.4.: Multi-layer Cluster map using standardization

> **Remark 3.13**
>
> *Note that constraints can also be put on edges and are not restricted to objects.*

The main idea we can take from Task 3.12 is that requirements (as they represent restrictions) will lead to a reduction in optimality. Fixed interfaces are requirements, hence special attention needs to be paid on how these interfaces are defined. Considering the definition of a requirement from [12], we can formalize the following:

> **Definition 3.14** (Requirement).
> Suppose a network $(\mathcal{O}, \mathcal{E})$ to be given. Then the conditions
>
> $$\bigcup_{j \in \mathcal{I}} \mathcal{O}_j \in S_k \qquad \text{(Union)} \qquad (3.8)$$
>
> $$\mathcal{O}_i \in S_j \cup \mathcal{O}_k \in S_l \qquad \text{(Split)} \qquad (3.9)$$
>
> $$(e_1, e_2) \in \mathcal{E} : (e_1, e_2) \in S_j \qquad \text{(Containment)} \qquad (3.10)$$
>
> $$(e_1, e_2) \in \mathcal{E} : e_1 \in S_j \wedge e_2 \in S_k \qquad \text{(Interface)} \qquad (3.11)$$
>
> can be used to impose requirements on a clustering S. The set of requirements is denoted by $\mathcal{R}$.

Within the Louvain Algorithm 3.9, these conditions can be integrated easily in the greedy strategy by checking whether or not a requirement needs to be enforced.

> **Algorithm 3.15** (Louvain algorithm for clustering with requirements)
> Consider a network $(\mathcal{O}, \mathcal{E})$ and requirements $\mathcal{R}$.
>
> 1. For $j = 1, \ldots, \#\mathcal{O}$ set $S_j = \{\mathcal{O}_j\}$
>
> 2. Compute metric (modularity (3.2) or conductance (3.6) or coverage (3.7))
>
> 3. Do
>
>   a) For $j = 1, \ldots, \#\mathcal{O}$
>
>   i. For each edge (j, k) or (k, j) of object $\mathcal{O}_j$ with $k \in \mathcal{O}_k$, $\mathcal{O}_j \neq \mathcal{O}_k$
>
>   ▪ If $\mathcal{O}_j$ can be removed from S_j and added to S_k according to $\mathcal{R}$
>
>   ▪ Remove $\mathcal{O}_j$ from its cluster and add it to cluster of $\mathcal{O}_k$
>
>   ▪ Recompute metric
>
>   ▪ If metric not improved, revert move of $\mathcal{O}_j$

> ii. Remove empty clusters
>
> while metric is improved

Remark 3.16

Note that internal interfaces can be set freely and should be designed to have no impact on optimality of the clustering given a defined metric. External interfaces on the other hand are subject to negotiation. A quantification of optimality degradation is given by so called Lagrangian variables, also called shadow prices, which are beyond the scope of this lecture. For details, we refer to [20].

Still, standardization (of modules) offers a number of advantages for both users and manufacturer, cf. Table 3.1.

Table 3.1.: Advantages of standardization for users and manufacturers

User	**Manufacturer**
• Becomes more transparent and easier to use	• Fulfills requirements of existing users
• Reduces error sources by using proven modules	• Allows maintenance and expanding competitiveness
• Minimizes effort for creation and commissioning	• Increases engineering efficiency
• Simplifies diagnosis and troubleshooting	• Simplifies management of component variants (flexibility)
• Clarifies documentation of modules and behavior	• Allows division of tasks into work packages
• Defines interfaces	• Allows virtual commissioning (digital twin)

On the other hand, disadvantages arise in particular for the differentiation strategy. In production, the latter is also called mass customization. In particular, we see the following:

Table 3.2.: Disadvantages of standardization for users and manufacturers

User	Manufacturer
• Increases wait time from order placement to shipment	• Renders forecasting of trends/spikes difficult
• Inhibits innovation outside standard ranges	• Increases cost to maintain variety of machinery/tools
• Affects flow of supply chains with partners	• Makes build-up of stock impossible
• Renders tracking of orders/projects difficult	• Increases complexity for returned components

Requirements for standardization may arise from different sources and are sector specific. Examples for sector specific norms are ISO/IEC 81346 [7] for industrial plant or IEC 61512 [4] for batch processes. Here, we focused on the general setting of requirements and how these impact on planning.

Link: For further details on standardization in production, we refer to the lecture *Computer Integrated Manufacturing*.

In the following Chapters 4 and 5, we additionally consider the information (IEC 61131 [6]) and control aspect (DIN IEC 60050-351 [22]). Before coming to that, we next consider waste within planning and how waste may be identified.

3.3. Lean planning

So far, we have seen the aims of modularization, conductance, coverage and standardization. While all of these aspects are fundamental for dividing and conquering a specific work task, the idea of lean / lean planning is to question the work task itself. In particular, in lean planning we aim to reduce waste. Thinking in terms of networks (Definition 2.5) and key performance indicators (Definition 2.19), waste can be defined as follows:

Definition 3.17 (Waste).
Given a network $(\mathcal{O}, \mathcal{E})$ together with a key performance criterion $J : \mathcal{I} \times \mathcal{I}^2 \to \mathbb{R}_0^+$. We call

> any object $\mathcal{O}_j$ or edge $\mathcal{E}_k$ waste if the condition
>
> $$J(\mathcal{O} \setminus \mathcal{O}_j, \mathcal{E} \setminus \mathcal{E}_k) \geq J(\mathcal{O}, \mathcal{E}) \tag{3.12}$$
>
> holds.

On the planning level, our task is to quantify elements with respect to their impact on the considered key performance indicator. Here, the latter definition says that an object or an edge exhibits a neutral or negative contribution towards the key performance criterion. If these objects or edges are not necessary within the network, they should be removed.

Still, lean planning features a lot more ideas including:

- Align all tasks with costumer needs

- Focus on strengths

- Optimize processes/networks

- Improve quality continuously

- Implement costumer centric company principle

- Strengthen self responsibility, empowerment and team work

- Implement decentralized and costumer focused structures

- Live leadership as service for coworkers

- Implement open information and feedback processes

- Improve culture and attitude within the company

Most of the latter are qualitative and not engineering related issues. In order to lead a company/process, it is still necessary to be familiar with these tasks but outside the scope of this lecture.

> **Link:** For further details on lean planning considering management and human resource tasks, we refer to the lecture *Operations Management*.

While waste can easily be quantified for a given setting/network, finding an improved one is much more involved and will be content of Chapter 7 focusing on optimization and leading.

CHAPTER 4

INFORMATION AND COMMUNICATION

Coming back to the network of systems/processes, which we discussed in Chapter 2, we come to notice that the links between systems/processes may consist of matter, energy or information (cf. Definition of process in [22]). As our aim is automation, within the present chapter we will focus on the latter part of information, which is required to steer the system. The question on how the information can be used will be addressed in the following Chapter 5.

In practice, such a network can consist of many different types different programs, computers, communication lines down to different machines, sensors and actuators. The intention of the present chapter is to show how all these different objects with their potentially different languages can be combined. To this end, we impose the idea of modularization and apply it to communication. Reconsidering our overall path of automation, Figure 4.1 indicates the focus of the present chapter.

We first address what should be understood by the term *information*. Thereafter, we move towards different transmission networks and communication structures before actually modularizing the information transfer itself. Thereafter, we swiftly address the ideas of channel access and message formats. This structure lays the foundation for the idea of actually controlling the system/process/network at hand in the following chapter.

4.1. Information processing

As outlined, information is one of the linking components in systems/processes and also in networks. Yet, in Chapter 1, we only discussed the term signal from DIN IEC 60050-351 (2014) [22]. Recall the latter:
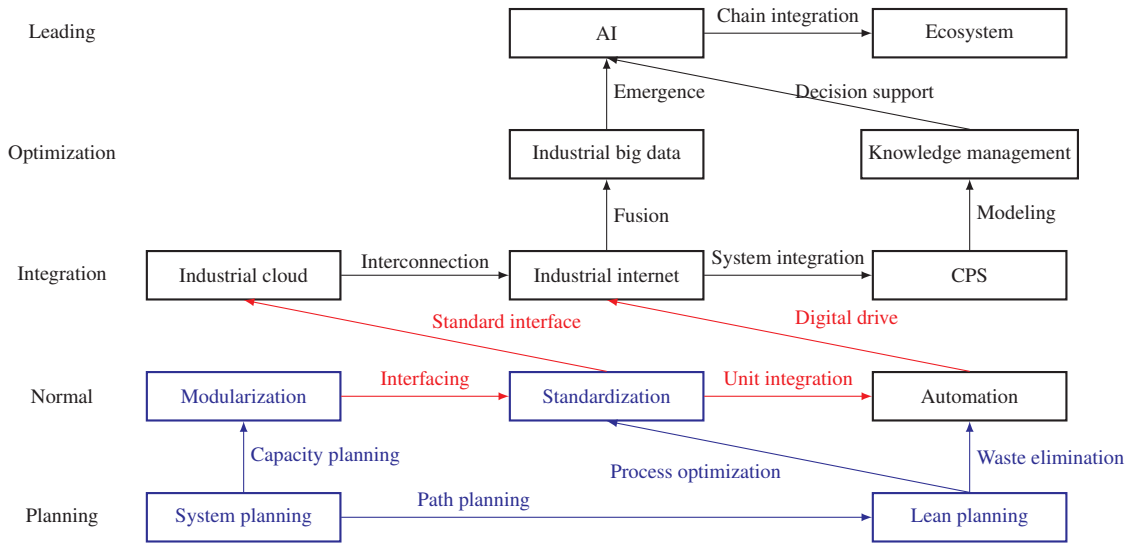
Figure 4.1.: Information in the integrated implementation path of automation

> A signal is a physical quantity that conveys information about one or more variable quantities using one or more of its parameters.
>
> DIN IEC 60050-351 (2014)

Hence, a signal is only a means of transport like an edge in a network. Regarding information, there exists no formal definition and the interpretation of the word is different in various fields. Common properties of information are reduction of uncertainty, relevance, redundancy freeness and newness.

In practice, information is gained by knowledge of context and respective processing of primary signal, i.e. raw data from objects such as sensors. Examples of data streams are given in Figure 4.2 and 4.3. As we can see, not only the means of transfer but also the context of data streams in application may vary significantly.

In general, information is typically something which is extracted from data by knowledge of the task at hand. As such, information stands between data and knowledge.

Definition 4.1 (Signal processing).

Reconsider a system

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{y}(t) = h(\mathbf{x}(t), \mathbf{u}(t), t)$$

(1.1)

with output $\mathbf{y}$. Then any map/method operating on the output is called signal processing.

GPS: Global Position System
OBD-II: Second On-board Diagnostic

ABS: Anti-skid Brake System
SRS: Supplemental Restraint System
EBA: Emergency Brake Assist

Ethernet   CAN-High
FlexRay   LIN
CAN   CAN-Low
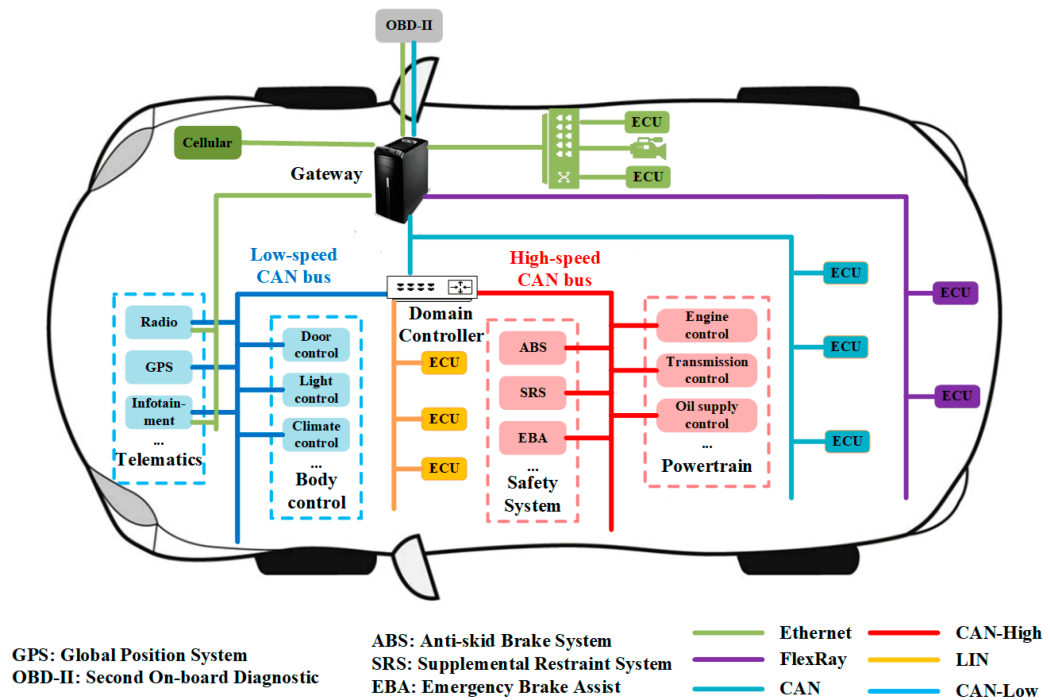
Figure 4.2.: Examples of data streams within a car
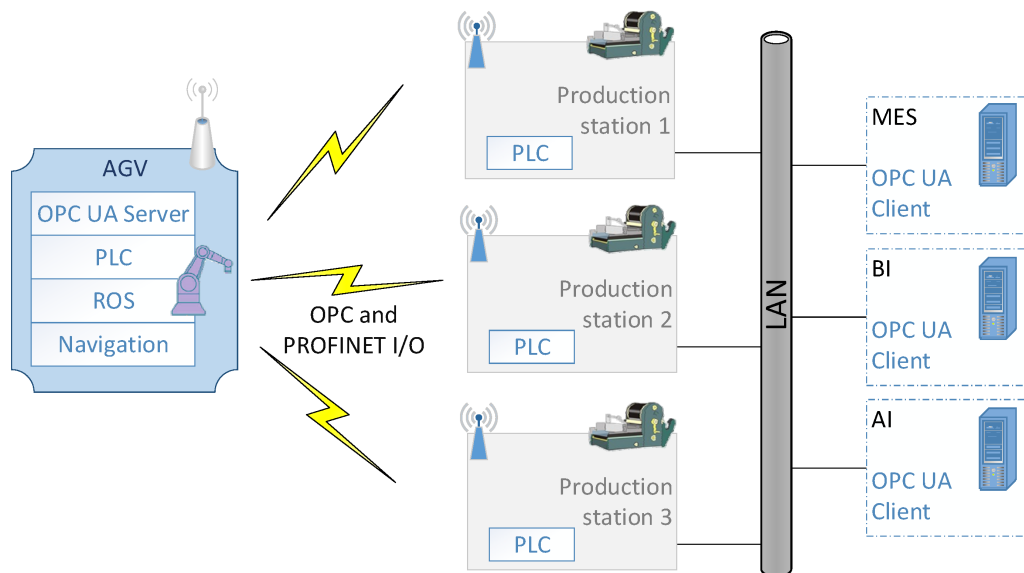
Figure 4.3.: Examples of data streams for a production system

In applications, signal processing consists of several steps and depends on what form of information shall be used. The basic step is always to take a measurement with some kind of sensing equipment. Using mechanics only, such a signal can already be used to take action on the process. This was the standard approach in the first industrial revolution, e.g. via the steam engine

governor, see again Chapter 1. The mechanical signal can also be transduced, i.e. pushed into an electric form. This allows for monitoring processes as in the second industrial revolution. Third step is to apply a converter and shift the signal into digital form, which already allows to apply logic units to it (third industrial revolution). Last, coding reveals state information, which can be included in higher computing tasks. The following Table 4.1 captures these steps.

Table 4.1.: Steps of signal processing

Attribute	Method	Attribute
State of system/process	Sensing	Measurement signal (arbitrary form)
Measurement signal	Transducing	Electric continuous signal
Electric continuous signal	Converting	Electric digital form
Electric digital signal	Coding	System/process state information

Task 4.2

An example of such a signal processing is a gyro, a device nowadays found in almost any smartphone or motion platforms. Assign the objects amplifier, stator, A/D converter and Kalman filter in the right sequence for a signal processing unit.

Solution to Task 4.2: The gyro sensor is built up as stator → amplifier → A/D converter → Kalman filter.

Remark 4.3

Note that the reverse path from gathered system/process information to influencing the system/process is equivalent but inverted in its sequence.

In todays practice, a variety of sensors can be found, which to some extend already integrate all of the steps above.

The choice of sensor depends on the deepness of integration of the process in higher management levels. If a control system shall be able to access not only a particular machine, a particular system/process but a vaster range, then information regarding that object needs to be available to the control. This directly leads us to the question on how information can be transmitted.
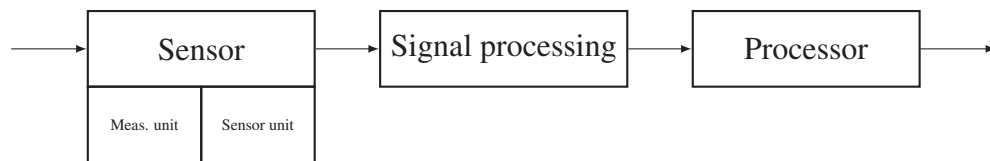
Figure 4.4.: Standard sensor configuration
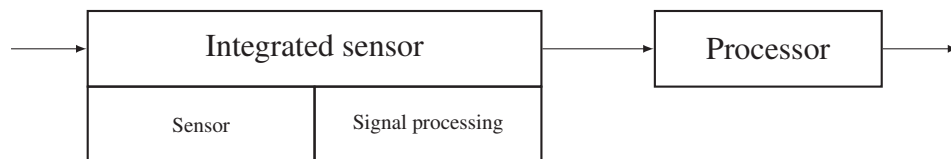
Figure 4.5.: Integrated sensor
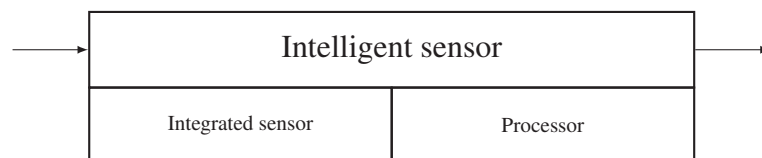
Figure 4.6.: Intelligent sensor

4.2. Transmission networks and communication structures

In the previous Chapter 2, we have already seen how object can be connected by edges in a network. While in that chapter we thought of networks more in a physical way, i.e. goods to be transported or tasks to be done, networks for information focus on the transmission of data/information.

Remark 4.4

Data and information are regularly used equivalently although they are not. This confusion stems from the problem that information gained from a single system/process, i.e. on a low level, may only be primary data for the integration of systems/processes, i.e. on a higher level. Hence, the same object is information for a control engineer on operational (machine) level but data for a planner on tactical (layout) level, or information for a planner but data for a data scientist on strategic (leading) level.

By definition of a network (cf. Definition 2.5), the connection of objects by edges depends on the application to be modeled. There exist two trivial cases:

1. If the network exhibits zero edges, then the network is completely disconnected.

2. If for each pair of distinct objects $\mathcal{O}_j$, $\mathcal{O}_k$ with $\mathcal{O}_j \neq \mathcal{O}_k$ there exists an edge $e = (\mathcal{O}_j, \mathcal{O}_k)$, then the network is fully connected.

In between these two extreme cases, there exists a wide variety of possible topologies, which exhibit certain advantages and disadvantages when it comes to transmission of data/information. Prototypes of such topologies are given in Figure 4.7.

(a) Line   (b) Bus

(c) Ring   (d) Star   (e) Meshed   (f) Bus
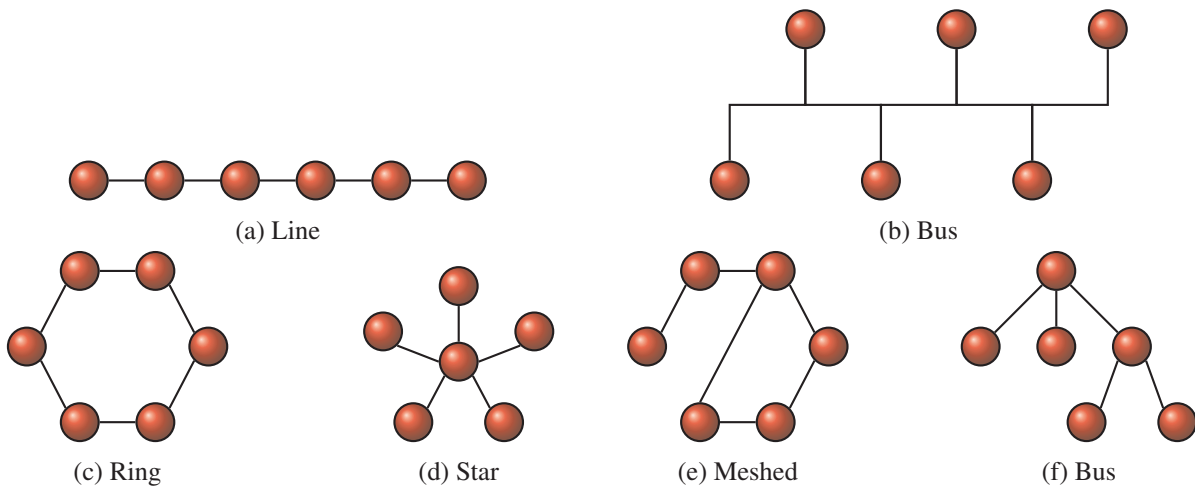
Figure 4.7.: Different network topologies

Focusing on the practically relevant topologies, we obtain the (dis-)advantages shown in Table 4.2.

Table 4.2.: Properties of communication topologies

Topology	Advantages	Disadvantages
Bus	• Scalable • No additional network components	• Not secure
Star	• Easily reducable • Secure • No routing	• Large number of lines • SPOF
Ring	• Easily extendable	• Not secure
		Continued on next page

Table 4.2 – continued from previous page

Topology	Advantages	Disadvantages
	• No additional network components	• Each object is SPOF • Long transmission times

For the physical realization of such networks, a wide range of tethered and wireless options are available. These are referred to as media.

Definition 4.5 (Network medium).
Consider a network $(\mathcal{O}, \mathcal{E})$. Then we call the physical possibility to realize edges $e \in \mathcal{E}$ network medium $\mathcal{N}_M$.

Here, we only address the different media and do not go into details for respective options. Still, even on a fundamental level, the transmission media show quite different behavior and exhibit respective pros and cons. The following Table 4.3 additionally links the media to the communication topologies.

Table 4.3.: Properties of communication media

Medium	Advantages	Disadvantages	Topology	Example
UTP[1]	Low costs Simple patching	Not secure Low throughput Low range	Bus, ring, star	ISDN Ethernet
S/STP[2]	Simple patching Medium throughput	High costs Not secure	Star	Ethernet
Coaxial	High throughput	Difficult patching	Bus	BNC (Car)
				Continued on next page

---

[1] Unshielded twisted pair such as CAT3 to CAT5
[2] Screend shielded twisted pair such as CAT6/7

Table 4.3 – continued from previous page

Medium	Advantages	Disadvantages	Topology	Example
	High range	Medium costs		F-plug (TV)
Fiber optic	High throughput	High costs	Star, ring	LC (Router)
	High range	Difficult patching		Soundbar
		Not bus-compatible		
Wireless	No cables	High costs	Free	LTE/5G
	No patching	Not secure		802.11/15/16

As one can already see from the different available media and topologies, the applied technologies we showed in examples in Table 4.3 may diverge quickly. To avoid such a divergence, a standard was developed to ensure that all technologies and media can be used interchangeably.

Remark 4.6

Note that not switching between modules for communication may require additional components, especially since some media are not compatible with some topologies.

4.3. Open system interconnection

The so called OSI (open system interconnection) reference model was developed to allow for a standardization of communication, which provides advantages for both users and developers/-manufacturers of communication components. As we learned in Chapter 3, such a standardization comes at the price of optimality. In the case of information/communication, the degradation typically comes from a non-application specific communication pattern, which arises by the reference model.

The reference model consists of seven layers [8], which control certain aspects of communication depicted in Figure 4.8.
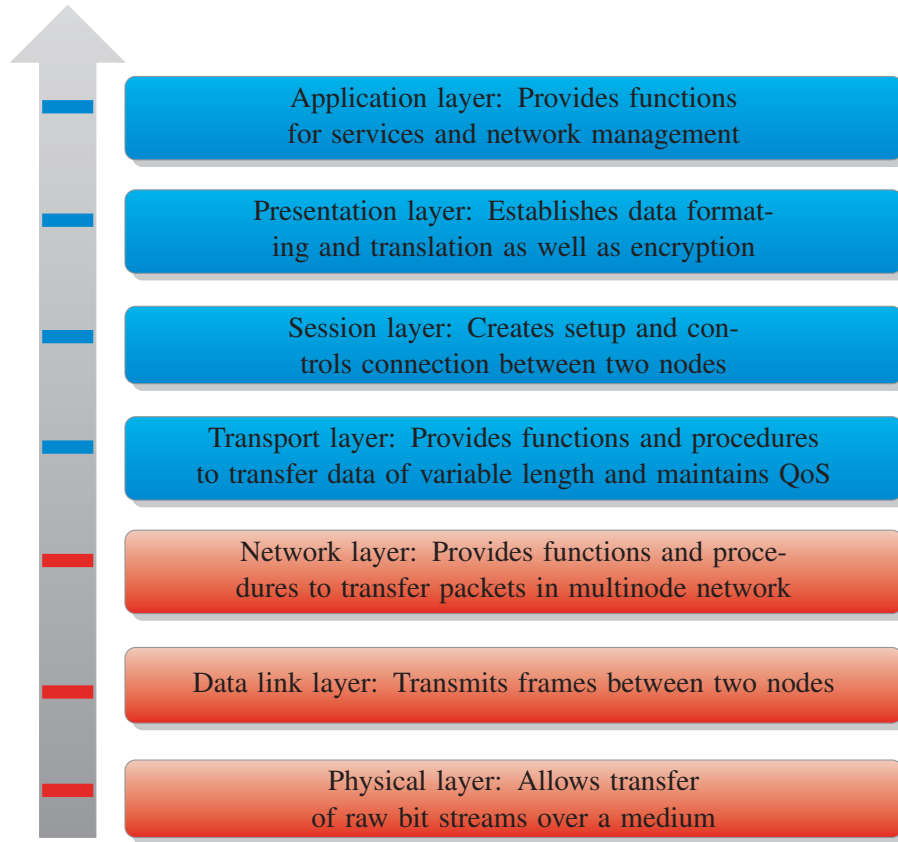
Figure 4.8.: Structure of the standardization process

As such, the OSI reference model is a network, which upholds certain rules.

Definition 4.7 (Layer).

Consider a network $(\mathcal{O}, \mathcal{E})$ with clustering S. Then we call the elements $\mathcal{L}_j \in S$ layers if

$$\mathcal{L}_j \cap \mathcal{L}_k = \emptyset \tag{4.1}$$

$$\forall \mathcal{O}_{j_1} \in \mathcal{L}_k, k < \#S : \exists \mathcal{O}_{j_2} \in \mathcal{L}_{k+1} \wedge (\mathcal{O}_{j_1}, \mathcal{O}_{j_2}) \in \mathcal{E} \tag{4.2}$$

$$\forall \mathcal{O}_{j_1} \in \mathcal{L}_k, k > 1 : \exists \mathcal{O}_{j_2} \in \mathcal{L}_{k-1} \wedge (\mathcal{O}_{j_1}, \mathcal{O}_{j_2}) \in \mathcal{E} \tag{4.3}$$

$$\forall \mathcal{O}_{j_1} \in \mathcal{L}_k : \nexists (\mathcal{O}_{j_1}, \mathcal{O}_{j_2}) \in \mathcal{E} : \mathcal{O}_{j_2} \notin \mathcal{L}_{k-1} \cup \mathcal{L}_{k+1} \tag{4.4}$$

$$\forall \mathcal{O}_{j_1} \in \mathcal{L}_k : \nexists \mathcal{O}_{j_2} \in \mathcal{L}_k \cap (\mathcal{O}_{j_1}, \mathcal{O}_{j_2}) \in \mathcal{E}. \tag{4.5}$$

Layers are thereby linked only to their predecessors and successors.

Remark 4.8

Note that the definition for layers is intended for a separable node (physically or virtually) acting by itself. In a network sense, we still talk about these nodes as objects.

The OSI network links different nodes, hence there exists an layer structure in each of the nodes.

Definition 4.9 (OSI network).

Consider a network $(\mathcal{O}, \mathcal{E})$. Then we call it OSI network if

- each object $\mathcal{O}_j \in \mathcal{O}$ is itself a network consisting of 7 layers in the sense of Definition 4.7 and Figure 4.8,

- for each edge $(\mathcal{O}_j, \mathcal{O}_k) \in \mathcal{E}$ there exist edges $(\mathcal{O}_{j_z}, \mathcal{O}_{k_z})$, $z = 1, \dots, 7$ connecting the layers, and

- for each communication process including more than two objects only layers 1-3 exist for intermediate objects.

Remark 4.10

The last condition in Definition 4.9 explicitly deals with the communication process, not the network. In fact, Definition 4.9 shows us two networks, one for the communication, and one within the objects for the respective layers.

For developers of programs, the first three layers are relevant as they can be adapted to application specifications, whereas the lower four layers address the transport of information. In terms of connectivity, the first five levels consider multihop and multi-destination problems, whereas the lower two layers implement point-to-point communication, cf. Table 4.4.

The top four layers within the OSI reference are so called host layers and deal with issues of the host. The lower three layers, on the other hand, are called media layers and focus on specifics of the respective media. In Figure 4.8, these layers are indicated by the different colors.

Table 4.4.: Layers in the OSI reference - **P**lease **D**o **N**ot **T**hrow **S**alami **P**izza **A**way

Layer	Classification	Connection	Examples
Application			DHCP, FTP, DNS
Presentation	Application oriented		MQTT, LDAP,
Session		Multihop	HTTP/S
Transport			TCP, UDP
			Continued on next page

Table 4.4 – continued from previous page

Layer	Classification	Connection	Examples
Network			IP, IPX
Data link	Transport oriented		802.11, MAC
Physical		Point-2-point	ARCNET

Here, we want to highlight some of the key properties of the respective layers:

1. The physical layer provides „physical" means for communication, that is mechanical, electrical, optical or other components to transmit bits. Hence, only 0 and 1 can be transmitted, which induces some kind of coding.

2. On the data link layer, we are already dealing with frames, which are sets of bits combined to be transmitted. The aim is to guarantee a most error free transmission and ability to access the medium. To this end, checks for successful transmission are made on the receiving end, yet no corrections are considered.

3. The network layer deal with packets of frames and aims to choose the right way to the destination while already avoiding congestion on the medium. To this end, this layer considers also different ways to reach the target, which is called routing.

4. The transport layer slices the information to be transmitted into packets and serves as internal router by assigning a port for communication. By assigning a port, this layer also converts data into technology dependent formats for the underlying layers.

5. The session layer organizes and synchronizes information transmission between two systems. As such, it supervises each connection and validates access permission.

6. The presentation layer converts the system depending representation of data into a independent form, i.e. allows for syntactic abstraction, but also for encryption or compression.

7. Last, the application layer provides access to functions of lower layers and serves as mediator.

While the OSI reference model tells us, how communication aspects shall be split hierarchically for each unit, it does not specify how communication channels shall be accessed physically, logically or timewise.

4.4. Network access

Coming from the OSI reference model, we now want to transmit signals between objects. To this end, we can utilize different communication structures and physical means of transmission as discussed in Section 4.2 and combine these with the OSI reference model from Section 4.3. Our aim is to get rules for sending data via a transmission medium, that is if and how access to the medium can be structured.

Here, we talk about network access in the following sense:

Definition 4.11 (Network access).

Consider a network medium $\mathcal{N}_M$. We call any set of rules to access the network medium *network access $\mathcal{N}_A$*.

In practice, we have to distinguish two forms of network access, that is deterministic and probabilistic methods.

Definition 4.12 (Deterministic network access).

Consider a network $(\mathcal{O}, \mathcal{E})$ together a network medium $\mathcal{N}_M$ and network access $\mathcal{N}_A$. If

- $\mathcal{N}_A$ is time-based or fixed for all $\mathcal{O}$ and

- the transmission and response time are bounded and known,

then we call $\mathcal{N}_A$ a deterministic network access method.

Definition 4.13 (Probabilistic network access).

Consider a network $(\mathcal{O}, \mathcal{E})$ together a network medium $\mathcal{N}_M$ and network access $\mathcal{N}_A$. If $\mathcal{N}_A$ is event-based then we call it a probabilistic network access method.

Why are these two forms relevant: The deterministic network access is only applicable if the objects themselves and the network medium can be characterized (more or less) completely. Such an implementation makes sense if the overall process will not be changed and no modifications will be made. This is typically the case for production lines. Examples of such network access methods are given in Table 4.5.

Table 4.5.: Deterministic network access methods

Method	Advantages	Disadvantages
Master/Slave	Master periodically asks slaves whether they want to access the network medium.	
	Simple organization	Max. latency proportional to number of objects
	Guaranteed response time	No communcation if master fails
Token passing	Token is handed over to net object in ring topology.	
	High performance capability	Long delays in case of errors
		Required surveillance of token
TDMA[3]	Each object has one/multiple time slots for network access in a cyclic period.	
	Short and constant cycle time	Required synchronization of time
	Low overhead	

In contrast to deterministic methods, probabilistic methods are more suitable for changing environments, changing number of objects and modifications to applications which result in different response times. The idea of these methods is to constantly listen to the network medium and start network access if the medium is available. While being very flexible, such methods also show structural shortcomings. In the following Table 4.6, we characterize a few of these methods.

Table 4.6.: Probabilistic network access methods

Method	Advantages	Disadvantages
CSMA[4]	Easily extendable	Not realtime capable
		Continued on next page

---

[3]Time Division Multiple Access
[4]Carrier Sense Multiple Access

Table 4.6 – continued from previous page

Method	Advantages	Disadvantages
CSMA-CD[5]	No coordination required	Requires permanent listening
	Detect collisions of packages via data matching, resends after object specific waiting time.	
	Short latency in low load case	Long waiting times in high load case
CSMA-CA[6]	Avoids collisions by priority rules.	
	Allocation of time slots possible	Required fixed rules

Apart from the network access, also the format of a transmission itself is of interest. As the format differs depending on the network medium and the used protocol, we will not go into detail but focus on the big picture along the OSI reference model. Figure 4.9 provides an overview on how data is split and appended by network and transmission information to be ready for transmission/-transport using the OSI layers.

As an example, Figure 4.10 shows a frame in for a CAN bus including the CAN PWM graphs on binary level.

---

[5]Carrier Sense Multiple Access - Collision Detection
[6]Carrier Sense Multiple Access - Collision Avoidance
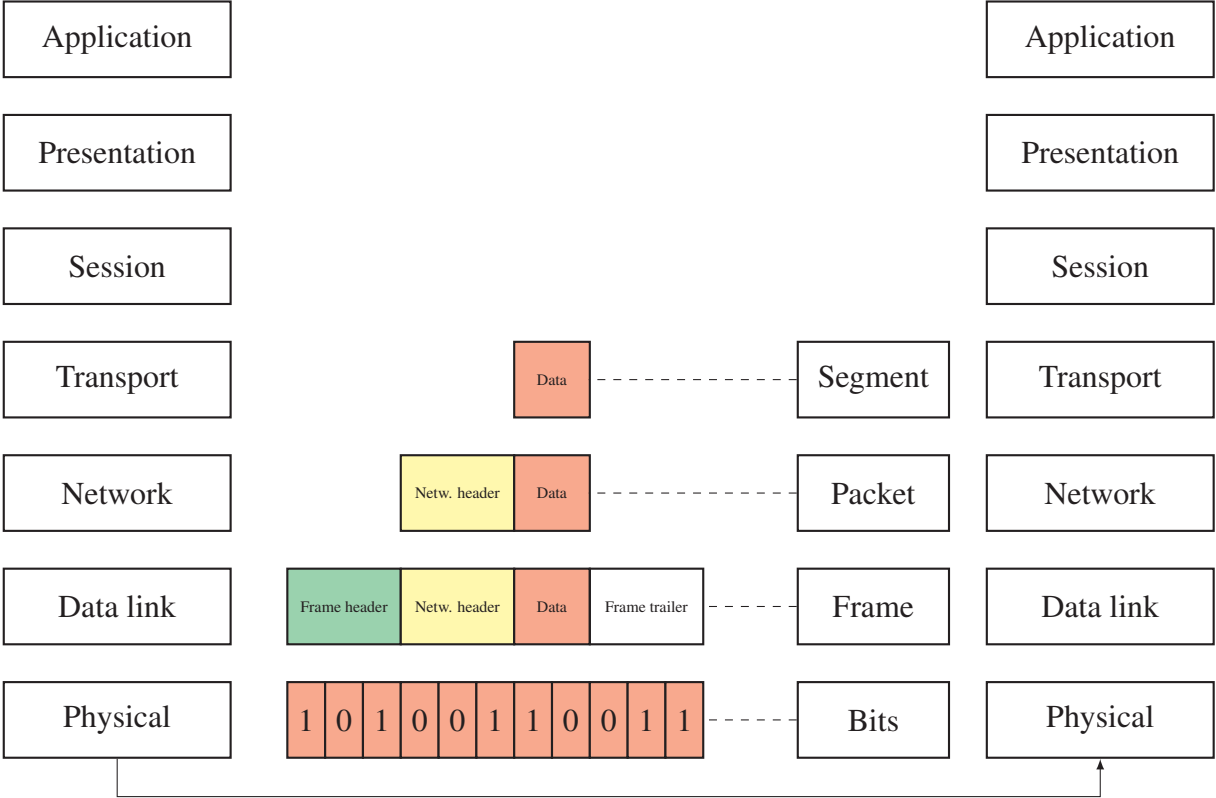
Figure 4.9.: Transforming network communication to bits
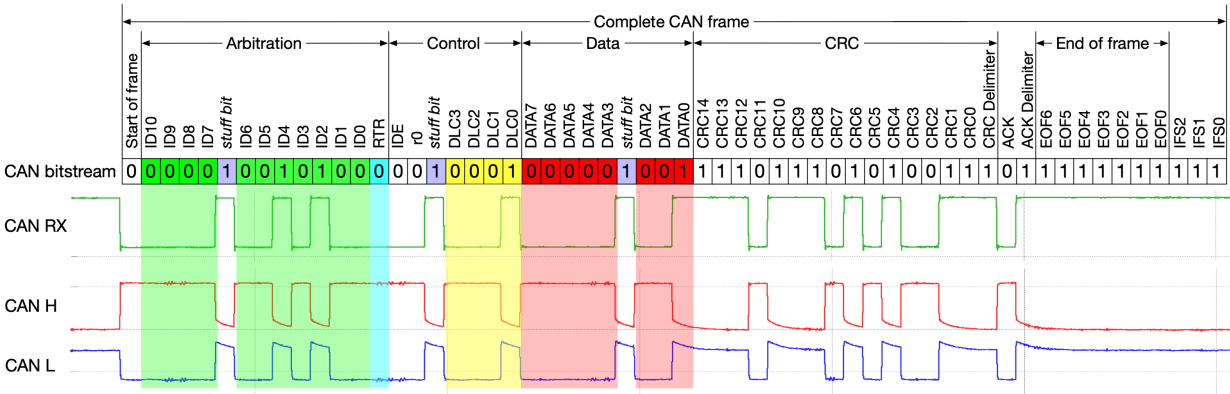
Figure 4.10.: CAN bus frame [25]

CHAPTER 5

CONTROL

In Chapter 1 we introduced the concepts of feedforward and feedback control. These two very different and also very basic concepts form the backbone of automation and automation systems. In particular, they link the higher level tasks of integration, optimization and leading to the normalization and planning levels, cf. Figure 5.1.
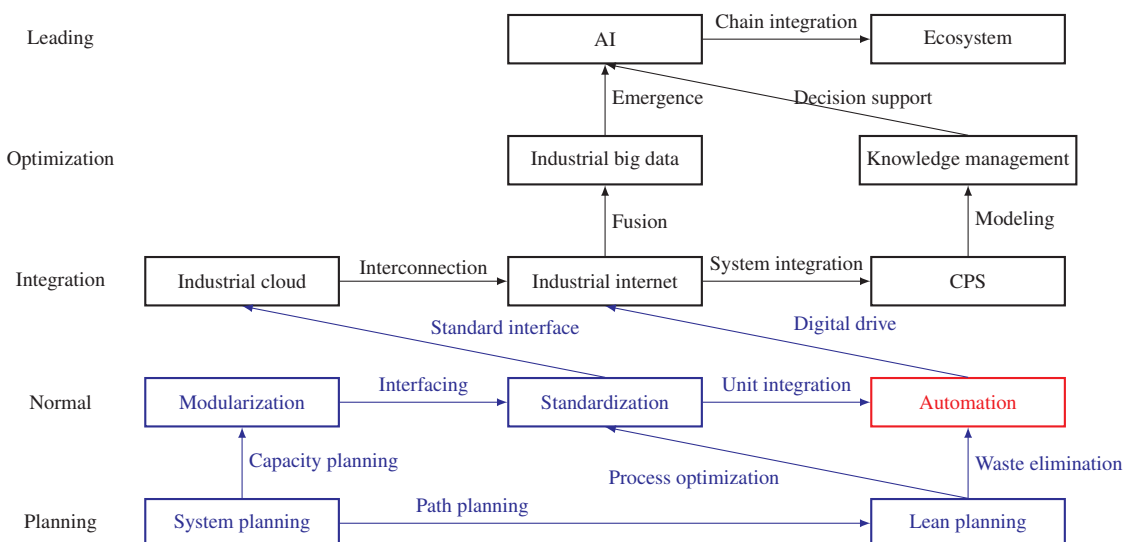
Figure 5.1.: Information in the integrated implementation path of automation

Within this chapter, we provide a short overview of feedback using the most commonly applied PID controller. Since the latter is only capable to treat one input and one output, we extend the setting to precontrol and prefilter where additional information from a feedforward can be integrated into the control loop.

In a large scale system/process, not only one output will have to be processed and not only one in-

put will have to be computed. These problems are referred to as multi-input-multi-output MIMO systems, which will be our entry window to combine from highlevel tasks down to machine level in the upcoming chapters.

5.1. PID control

In Definitions 1.9 and 1.10 we introduced two concepts of control, the so called feed forward and feedback. Roughly speaking, a feed forward is a plan that certain actions shall be taken at certain time instances. Therefore, it is a mapping from time to input action. In contrast to that, feedback is a plan that certain actions shall be taken if certain outputs are obtained. Hence, a feedback is a mapping from output to input. In order to be properly defined, recall our definition of a nonlinear control system

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{y}(t) = h(\mathbf{x}(t), \mathbf{u}(t), t). \tag{1.1}$$

To derive a feedback for the latter, we require a reference:

Definition 5.1 (Reference).
Given a system (1.1), we call $\mathbf{w} : \mathcal{T} \to \mathcal{Y}$ a reference.

Figures 5.2 and 5.3 sketch the respective paths of information.
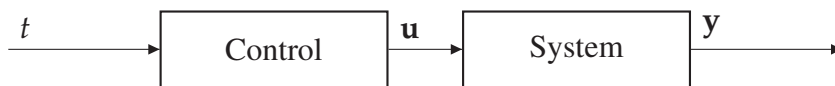
Figure 5.2.: Simple feed forward
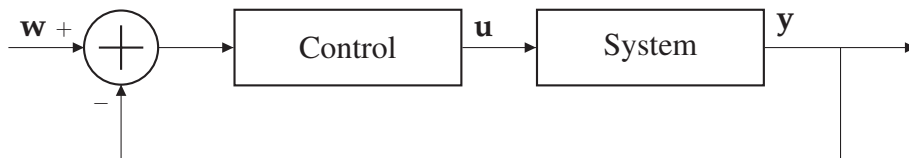
Figure 5.3.: Simple feedback

In practical applications, both concepts are used and also typically integrated into one another. Yet, they show different properties.

Task 5.2
Discuss the concepts of feed forward and feedback in driving situations. Highlight disadvantages.

Solution to Task 5.2:

- A feed forward is similar to reading a street map and deriving a route how to get from a starting to a destination point.

- A feedback is accelerating and steering a vehicle based on its current state given by measurements/perception and its environment.

The downsides for this particular example are :

- By reading a map, we have no information on the actual road usage, e.g., whether a vehicle breaks right in front of ours.

- By driving only, we don't know about route alternatives, e.g., we may drive into a dead end road.

On a more generic view, the concepts exhibit the advantages and disadvantages given in Table 5.1.

Table 5.1.: Advantages and disadvantages of feed forward and feedback

Method	Advantages	Disadvantages
Feed forward	Integrates external knowledge Plans ahead via simulation Addresses KPIs	May show infeasible solutions Requires good model
Feedback	Guarantees stable behavior Reacts to circumstances Addresses system properties	May end in unwanted operating points Requires theoretical insight

While feedbacks are more difficult in theory, they typically operate on system level. For this

reason, we first consider feedback before integrating feed forward into such loops in the following Section 5.2.

PID control — proportional integral derivative is probably the most common controller used in industrial applications. Formally, a PID control is given by the following:

Definition 5.3 (PID control).

Consider a system (1.1). Then we call $\mathbf{u} : \mathcal{Y} \to \mathcal{U}$ given by

$$\mathbf{u}(t) = K_P \cdot (\mathbf{w}(t) - \mathbf{y}(t)) + \int_{t_0}^{t} K_I \cdot (\mathbf{w}(\tau) - \mathbf{y}(\tau)) \ d\tau + \frac{\partial}{\partial t} \left(K_D \cdot (\mathbf{w}(t) - \mathbf{y}(t)) \right) \quad (5.1)$$

PID controller. The parameters K_P, K_I and K_D are called proportional, integral and derivative gains.

A PID controller — like any other controller — is introduced to enforce a so called stability property for operating points. Operating points are desired points for a system/process, e.g., temperatures at which a chemical process works best, or speeds at which an aircraft utilizes the least gas per kilometer. More general:

Definition 5.4 (Operating point).

Consider system (1.1). Then the pairs $(\mathbf{x}^\star, \mathbf{u}^\star)$ satisfying

$$f(\mathbf{x}^\star, \mathbf{u}^\star) = 0 \quad (5.2)$$

are called *operating points* of the system. If (5.2) holds true for any $\mathbf{u}^\star$, then the operating point is called strong or robust operating point.

As stated before, the result/property we are most interested in is stability. Stability can basically be described as if the system/process is at an operating point, then it stays there, and if it is far way, then solutions are driven to the operating point. Utilizing Definition 5.4 we can introduce two concepts of stability and asymptotic stability, robustness and controllability. These concepts depend on the interpretation of $\mathbf{u}$ as an external control or a disturbance.

Definition 5.5 (Stability and Controllability).

For a system (1.1) we call $\mathbf{x}^\star$

- *strongly* or *robustly stable* operating point if, for each $\varepsilon > 0$, there exists a real number

$\delta = \delta(\varepsilon) > 0$ such that for all $\mathbf{u}$ we have

$$\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq \delta \implies \|\mathbf{x}(t) - \mathbf{x}^\star\| \leq \varepsilon \qquad \forall t \geq 0 \qquad (5.3)$$

- *strongly* or *robustly asymptotically stable* operating point if it is stable and there exists a positive real constant r such that for all $\mathbf{u}$

$$\lim_{t \to \infty} \|\mathbf{x}(t) - \mathbf{x}^\star\| = 0 \qquad (5.4)$$

holds for all $\mathbf{x}_0$ satisfying $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq r$. If additionally r can be chosen arbitrary large, then $\mathbf{x}^\star$ is called *globally strongly* or *robustly asymptotically stable*.

- *weakly stable* or *controllable* operating point if, for each $\varepsilon > 0$, there exists a real number $\delta = \delta(\varepsilon) > 0$ such that for each $\mathbf{x}_0$ there exists a control $\mathbf{u}$ guaranteeing

$$\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq \delta \implies \|\mathbf{x}(t) - \mathbf{x}^\star\| \leq \varepsilon \qquad \forall t \geq 0. \qquad (5.5)$$

- *weakly asymptotically stable* or *asymptotically controllable* operating point if there exists a control $\mathbf{u}$ depending on $\mathbf{x}_0$ such that (5.5) holds and there exists a positive constant r such that

$$\lim_{t \to \infty} \|\mathbf{x}(t) - \mathbf{x}^\star\| = 0 \qquad \forall \|\mathbf{x}_0 - \mathbf{x}^\star\| \leq r. \qquad (5.6)$$

If additionally r can be chosen arbitrary large, then $\mathbf{x}^\star$ is called *globally asymptotically stable*.

Task 5.6

Draw solutions of systems for each of the cases in Definition 5.5.

Remark 5.7 (BIBO stability)

In some books, the concept of strong/robust stability is also termed BIBO (bounded input bounded output) stability.

Note that for strongly asymptotically stable control systems the control does not affect the stability property but can be used to improve the performance of the system. The concept of weak stability, on the other hand, naturally leads to the question how to compute a control law such that $\mathbf{x}^\star$ is weakly stable, and, in particular, how to characterize the quality of a control law.

5.2. Prefilter and precontrol

As outlined before, our path of automation (Figure 5.1) requires us to integrate higher and lower level. Basically, a plan on a high level is a feed forward, while a working routine on a lower level is a feedback. In the following, we consider a fundamental approach to integrate a feed forward and a feedback into one loop. To this end, two structures are possible, cf. Figures 5.4 and 5.5 respectively.
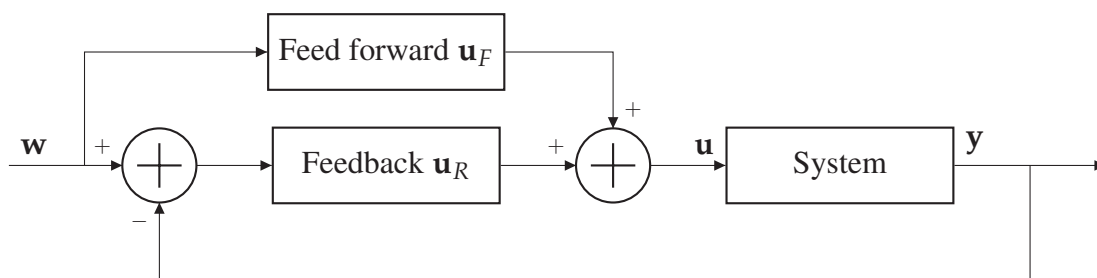
Figure 5.4.: Structure of a precontrol
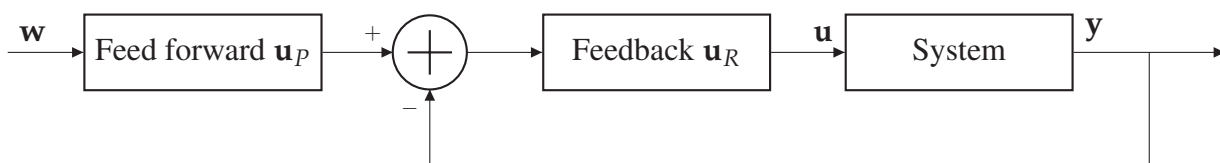
Figure 5.5.: Structure of a prefilter

Remark 5.8

We like to stress that as the feed forward does not interact with the feedback, the stability properties of the closed loop remain unchanged. Hence, the structures above allow us to foster the advantages of both the feed forward and the feedback concepts.

The structures of the precontrol from Figure 5.4 and of the prefilter from Figure 5.5 allow to simultaneously treat reference tracking via the feedback and leading via the feed forward. The design of a precontrol is done with the following two steps:

Algorithm 5.9 (Design precontrol)

Consider a control system as illustrated in Figure 5.4.

(1) Design the feed forward $\mathbf{u}_F(t)$ to be (approximately) the inverse of the open loop system (1.1).

(2) Design the feedback $\mathbf{u}_R(\mathbf{y})$ such that desired properties such as stability are guaranteed as good as possible.

The idea to design a prefilter is exactly the other way around:

Algorithm 5.10 (Design prefilter)

Consider a control system as illustrated in Figure 5.5.

(1) Design the feedback $\mathbf{u}_R(\mathbf{y})$ such that desired properties such as stability are guaranteed as good as possible.

(2) Design the feed forward $\mathbf{u}_P(t)$ to be (approximately) the inverse of the closed loop system (1.1) with feedback $\mathbf{u}_R(\mathbf{y})$.

Remark 5.11

As a function, the prefilter is a map $\mathbf{u}_P : \mathcal{T} \to \mathcal{Y}$ whereas the precontrol maps $\mathbf{u}_F : \mathcal{T} \to \mathcal{U}$. Hence, these function live in different arenas of the control problem task.

Note that the precontrol — in contrast to the prefilter — does not depend on the closed loop. Consequently, there is no need to adapt it in case the closed loop is optimized or changed after the design process is finished.

Despite this difference, the following equivalency result holds:

Theorem 5.12 (Equivalency precontrol and prefilter).

Suppose a system (1.1) *to be given. Then*

■ *for every precontrol defined via Algorithm 5.9 there exists a prefilter, which exhibits an identical closed loop, and*

■ *for every prefilter defined using Algorithm 5.10 there exists a precontrol, which exhibits an identical closed loop.*

Note that the idea of integrating a feed forward into a feedback directly corresponds to the integration, optimization and leading level being connected to the normalization and planning level. Still, prefilter and precontrol are limited to single pear applications.

Recalling the structure of an automated system/process introduced in Chapter 1, cf. Figure 5.6, we now have all the means to address a system/process with one input and one output.
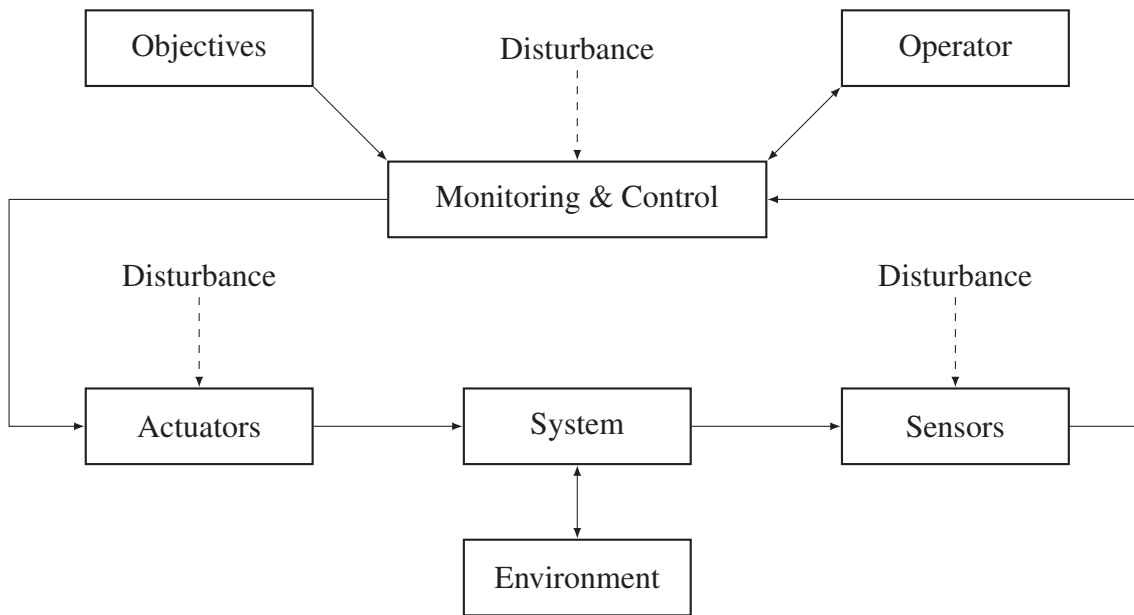
Figure 5.6.: Structure and process of automated systems

As the intention for automation engineering and in particular concepts like Industrie 4.0 or Industrial IoT is to work on a possibly global scale, we will address how large scale application can be treated using centralized and decentralized ideas in the upcoming Chapter 6 before accessing the distributed problem range in Chapter 7.

Part II.

Integration, optimization and leading

CHAPTER 6

NETWORKING

Having understood the components of networks, communication and control, we saw how one system/process can be designed, modeled, standardized and controlled. Within the present chapter, we address the issue of aligning multiple systems/processes, i.e. we move from a single system to a multi system setting. While we technically already considered such a setting in our planning Chapter 2 using networks, we left out the components communication and information as well as control and coordination. In practice, specifically coordination poses problems. Considering our path of automation, Figure 6.1 shows the components we consider in this chapter.

Figure 6.1.: Information in the integrated implementation path of automation

To show the problems of interconnection and system integration, we highlight traditional solutions, which are based on decoupling of systems/processes. While being simple to use, the

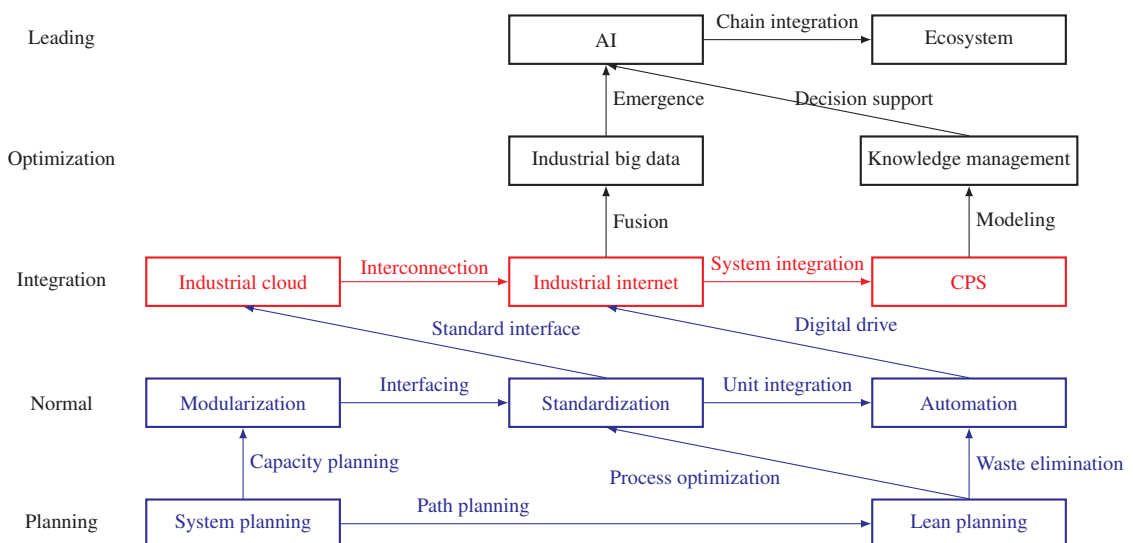state based decoupling induces stocks, the time based decoupling induces delays, and the control based decoupling induces disturbances. Hence, these solutions induce fall short in the sense of lean management.

To avoid decoupling, the approach using cyber physical systems can be used. These systems are a digital version of the respective physical systems/processes, which can be fed using data or information from the physical part to represent, monitor or interact with the physical system/process. The digital version is not bound to the restriction of the physical one, in particular regarding computing power and information access. In this chapter, we discuss the respective technologies of industrial cloud, internet and cyber physical system, which will be the basis for optimization in the following and concluding Chapter 7.

6.1. Decoupling

In order to understand why new technologies like industrial cloud, industrial internet and cyber physical systems can have a major impact on automation, we must first understand how interconnection and system integration is working as of today. The basic idea of (typically) any engineering task is to keep things as simple as possible. Applying the latter to interconnections and system integration means that we should try to separate and split up systems/processes as much as possible such that no coupling between them exists.

More formally, we define the following:

Definition 6.1 (MIMO system).
Consider a system
$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{y}(t) = h(\mathbf{x}(t), \mathbf{u}(t), t). \tag{1.1}$$

Then we call the system to be multi input multi output (MIMO) if

$$\left\| \frac{\partial^2 \mathbf{y}}{\partial a_j \partial a_k} \right\| \geq \theta \tag{6.1}$$

for some $a_j, a_k \in \{y_1, \ldots, y_{n_y}, u_1, \ldots, u_{n_u}, t\}$ with $\theta \in \mathbb{R}^+$.

The threshold parameter θ indicates the degree of coupling of the inputs and outputs. Hence, if there exists no pair a_j, a_k for which (6.1) holds true, then all components can be treated independently. Still one has to keep in mind that the threshold allows for certain small impacts, hence the control of the independently designed systems should be capable to suppress disturbances emanating from other systems.

Before coming to approaches for decoupling, we want to highlight that coupling occur on various levels and variables. Examples for strongly coupled outputs are

- pressure and temperature for steamers,

- position, velocity and force for robot arms, or

- produced parts in serial production lines.

Coupled inputs may be

- position control of multiple drives for robots,

- roll and yaw angle control for flying curves with an aircraft, or

- temperature, pH measurement and biomass distribution for bio reactors.

Within this section, we briefly discuss ideas on how to use states, time and control to decouple a system/process and indicate limitations of such approaches.

6.1.1. Decoupling of states and outputs

The first and very basic idea of decoupling uses states. If two states are connected, we can only observe this in the respective outputs y_j, y_k.
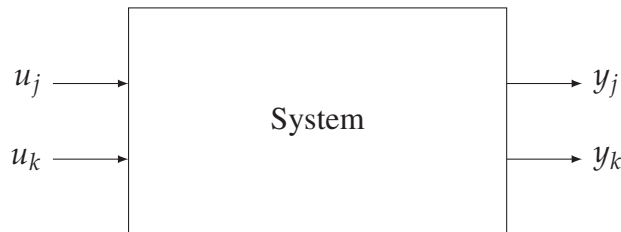
Figure 6.2.: MIMO system with two inputs and two outputs

From (6.1), we directly obtain

$$\left\| \frac{\partial^2 \mathbf{y}}{\partial y_j \partial y_k} \right\| \geq \theta. \tag{6.2}$$

The idea to decouple these two outputs is to add a new state

$$x_{jk}(t) := y_j(t) - y_k(t)$$

to the system eliminating the coupling. While introducing the new state solves the pure coupling problem at least mathematically, the downside is that the new state sums up all coupling problems. In practice, the new state is basically a storage area, e.g. for material, energy, people or information. Hence, such an approach is good to compensate for fluctuations, yet not for system/process instabilities. Unfortunately, such a storage area can only be used to a certain extend, i.e. until the storage capacity is reached.

Table 6.1.: Advantages and disadvantages of output decoupling

Advantage	Disadvantage
✓ Simple separation	✗ Induces storage
✓ Small additional load	✗ Unable to treat instabilities

6.1.2. Decoupling of time

An alternative to decoupling via states/outputs is decoupling via time. Similarly, we have

$$\left\| \frac{\partial^2 \mathbf{y}}{\partial y_j \partial y_k} \right\| \geq \theta,$$

yet now we utilize a time difference to redefine

$$y_j(t) = \eta(t) \cdot y_j(t)$$
$$y_k(t) = \eta(t - \delta) \cdot y_j(t)$$

for some $\delta > 0$. Here, $\eta(\cdot)$ is the Heaviside function

$$\eta(t) = \begin{cases} 0, & t < 0 \\ \text{undefined}, & t = 0 \\ 1, & t > 0 \end{cases}$$

representing a unit jump. While mathematically eliminating the coupling at a specific point in time t, the result of a time decoupling is that system/process steps are executed consecutively. In practice, such an approach is widely used to avoid parallel tasks to be controlled, e.g., driving and welding with a robot arm.

Table 6.2.: Advantages and disadvantages of time decoupling

Advantage	Disadvantage
✓ Simple separation	✗ Induces time delays
✓ No additional load	✗ Requires program adaptation

6.1.3. Decoupling of control

Regarding coupling of controls, a decoupling is more involved. Zooming into the setting of Figure 6.2, there are two different structures resembling feed forward and feed back connectivity.

Definition 6.2 (P canonical structure).

Consider a system with two inputs and two outputs. If the coupling of inputs to outputs exhibits a feed forward structure as shown in Figure 6.3a where P_{jk} is the transfer function of input u_k to output y_j for all j and k, then we call it P canonically structured.

Definition 6.3 (V canonical structure).

Consider a system with two inputs and two outputs. If the coupling of inputs to outputs exhibits a feedback structure as shown in Figure 6.3b where V_{jk} are the respective transfer functions for all j and k, then we call it V canonically structured.

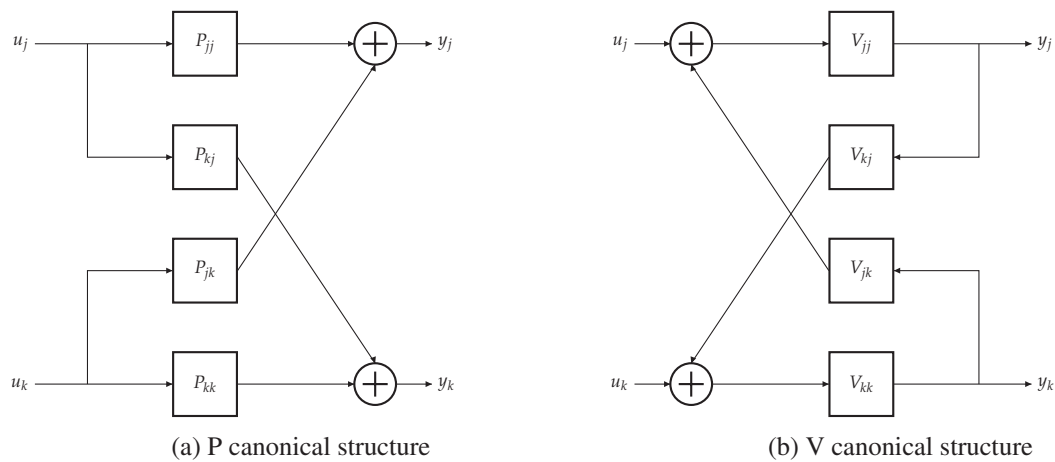(a) P canonical structure   (b) V canonical structure

Figure 6.3.: Canonical structures of MIMO systems with two inputs and two outputs

Both structures are often found in practice showing the properties given in Table 6.3.

Table 6.3.: Properties of P and V canonical structure

P canonical structure		V canonical structure	
✓	Direct correspondence to transfer matrix	✗	Required transformation of transfer matrix
✗	Typically no connection to modeling	✓	Direct derivation via modeling
✓	Easy to treat	✗	Difficult to treat
✗	Typically no equivalent of P_{jk}, P_{kj} in real system	✓	Equivalent of V_{jk}, V_{kj} in real system
✗	Physical interpretation questionable	✓	Physical interpretation given

As the P canonical structure is more easily treatable via control methods, one typically transforms V canonical systems to P canonical structure. To this end, we have

Theorem 6.4 (Equivalence P and V canonical structure).
Consider two systems with two inputs and two outputs to be given. Suppose one system is in P canonical structure and one in V canonical structure. If

$$\begin{bmatrix} P_{jj} & P_{jk} \\ P_{kj} & P_{kk} \end{bmatrix} = \begin{bmatrix} 1 & -V_{jj} \cdot V_{jk} \\ -V_{kk} \cdot V_{kj} & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} V_{jj} & 0 \\ 0 & V_{kk} \end{bmatrix} \tag{6.3}$$

holds, then both systems are equivalent.

Regarding decoupling, we define the following:

Definition 6.5 (Decoupling control).
Consider a system with two inputs and two outputs in P canonical structure. If the control exhibits the structure given in Figure 6.4 where S_{jk} is the transfer function of input u_k to output y_j for all j and k, then we call it decoupling control.

The idea of decoupling control is a special case of disturbance rejection, i.e. we eliminate or at least reduce the impact of the systems on one another, which allows us to apply standard methods for the decoupled circuits.

Within Figure 6.4, there are four controllers which need to be designed. While designing, the intention is that

- R_{jj} shall control y_j using u_j (main system S_{jj}),
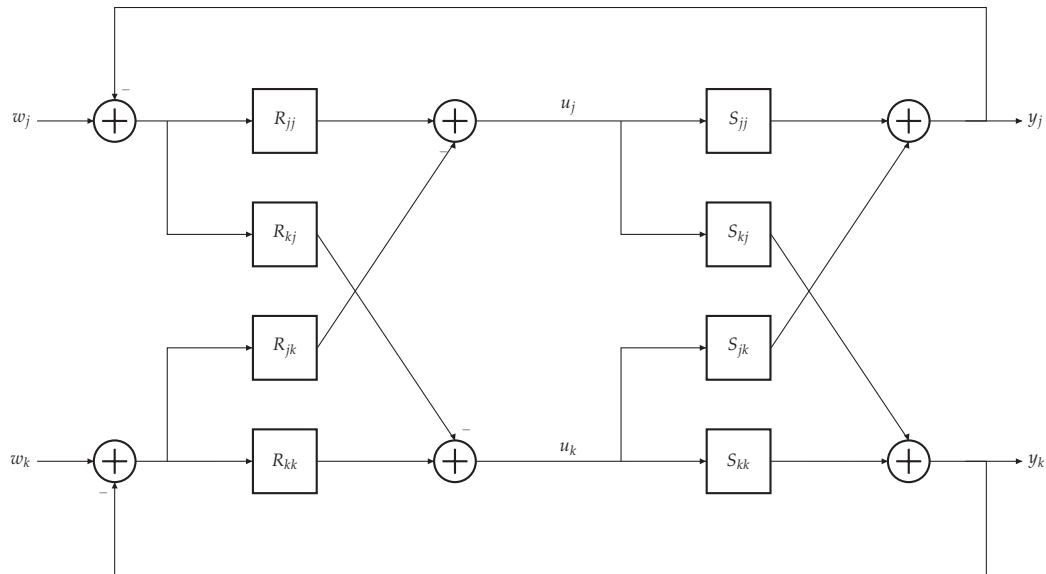
Figure 6.4.: Decoupling structure of MIMO system with P canonical structure

- R_{jk} shall eliminate the impact of u_k on y_j (coupling system S_{jk}),

- R_{kj} shall eliminate the impact of u_j on y_k (coupling system S_{kj}), and

- R_{kk} shall control y_k using u_k (main system S_{kk}).

We now focus on eliminating the impact of the second system on the first, cf. Figure 6.5.
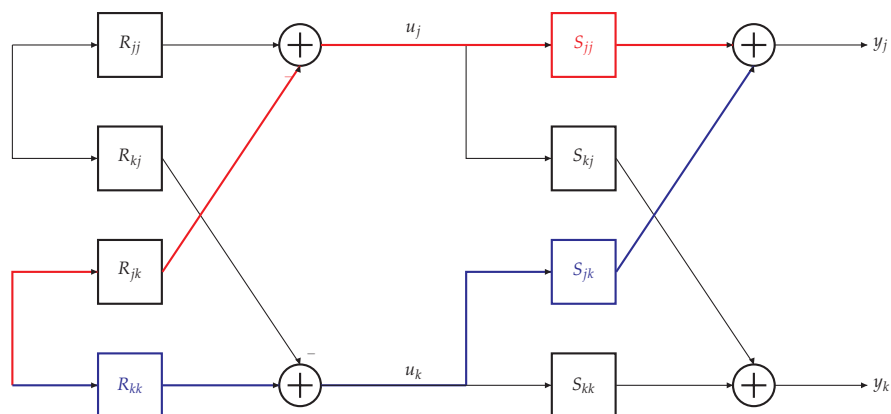
Figure 6.5.: Elimination of coupling

In order to eliminate one another, the blue and red paths in Figure 6.5 need to be identical. Hence, we directly obtain

> **Theorem 6.6** (Decoupling condition).
>
> *Consider a MIMO system with two inputs and two outputs in P canonical structure subject to a decoupling control. If the conditions*
>
> $$R_{jk} = R_{kk} \cdot \frac{S_{jk}}{S_{jj}} \tag{6.4}$$
>
> $$R_{kj} = R_{jj} \cdot \frac{S_{kj}}{S_{kk}} \tag{6.5}$$
>
> *hold, then the system is decoupled.*

It is of particular importance that even in the case of ideal decoupling, each system depends on both the main and the decoupling control. As a consequence, if we want to adapt the main controller in a later stage of the development, the decoupling controller needs to be adapted as well.

Table 6.4.: Advantages and disadvantages of control decoupling

Advantage	Disadvantage
✓ Keeps for MIMO structure	✗ Computationally involved
✓ Standardized P and V structures	✗ Structure limited in usage or derivation
✓ Allows for independent design	✗ Require additional controllers
✓ Allows for basic methods	✗ Requires specific decoupling structure

The beauty of the latter three approaches is that systems/processes can be treated completely separately. On the downside, these approaches are the cause of technically unnecessary stocks, time delays and disturbances. In the following section, we address ideas to avoid these shortcomings.

6.2. Digital twin

In the context of networking, one idea to completely avoid decoupling is to remodel the system/process on a software layer where it is possible to tackle the overall system/process. The big advantage of such an idea in contrast to decoupling is that all aspects of a system/process can be handled on a software level. To this end, a digital representation is required. Here, we follow the notation of the Industrial Internet Consortium [14] and the National Institute of Standards and Technology [19]. The building blocks of such a representation are so called data elements

Definition 6.7 (Data element).
A data element is a basic unit of information built on standard structures haveing a unique meaning and distinct units or values.

Using such elements, a digital representation can be formed:

Definition 6.8 (Digital representation).
Consider a system/process to be given. A digital representation is a data element representing a set of properties of the system/process.

While a digital representation is typically used to represent a physical entity, we like to point out that the above definition can also be applied for non-physical entities, e.g. to capture software behavior on an abstracted level. In the past, all of the latter were done document centric. Within this lecture, we consider a model centric view, which is commonly found in systems engineering but – at least currently – not too often in industrial practice.

Definition 6.9 (Systems modeling language (SysML)).
Consider a system/process. Then we call a set of

- requirements,

- behaviors given by

  - activity diagram

  - sequence diagram,

  - state machine diagram, and

  - use case diagram

- structures given by

  - block definition diagram,

  - internal block diagram,

  - parametric diagram, and

  - package diagram

a model according to the systems modeling language.

The diagrams within SysML follow a specific structure, which is closely connected to the unified modeling language (UML), cf. Figure 6.6 for details.
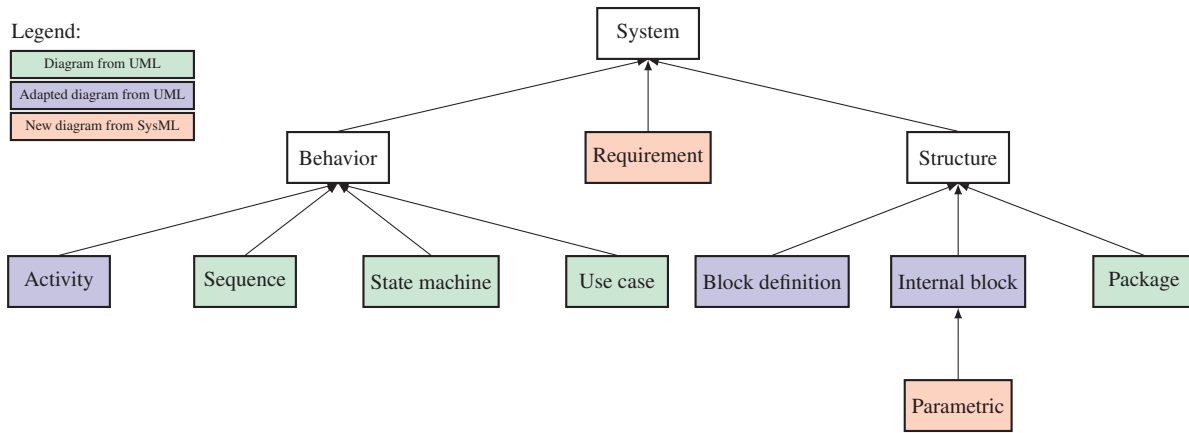
Figure 6.6.: Diagram taxonomy for systems (according to SysML)

Using this digital representation, we can define a digital twin.

Definition 6.10 (Digital model/shadow/twin).

Suppose a system/process with inputs and outputs, a digital representation of the same system/process and communication possibility between both to be given.

- If there exists at least a manual data flow from the system/process to the digital representation, then we call the digital respresentation a digital model.

- If there exists at least an automated data flow from the system/process to the digital representation, then we call the digital respresentation a digital shadow.

- If there exists a bidirectional automated data flow between the system/process and the digital representation, then we call the digital representation a digital twin.

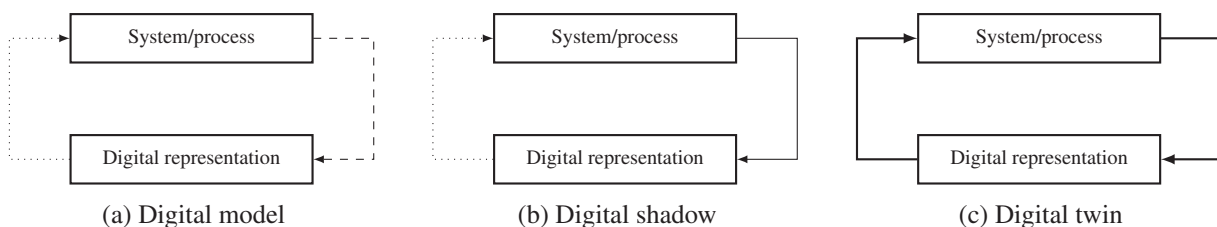(a) Digital model (b) Digital shadow (c) Digital twin

Figure 6.7.: Comprehend the difference between digital model/shadow/twin

Using Definition 6.10, we see that the difference between the three forms exists in the interaction structure. For the digital model, a data transfer is done manually from real system/process to the digital one. The intention of such a structure is to obtain insights into the system/process

behavior and its properties. Its applications are on the strategic layer. Using an automated (and possible real time capable) data stream, the digital shadow can be used for monitoring purposes. Due to the automated nature, it is a reporting tool and can be used for planning. Last, the automated backwards flow to the physical system/process allows the digital twin to be applied on the operational layer. Figure 6.8 illustrates these points.
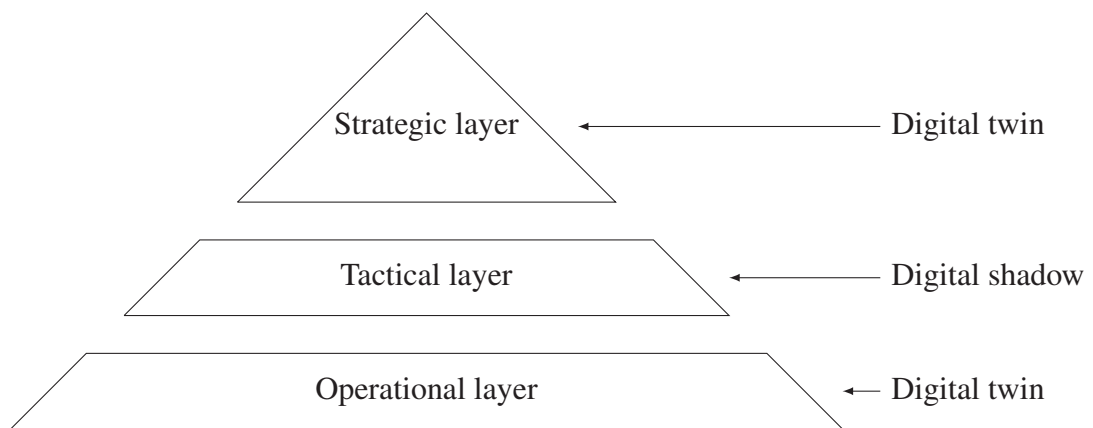

Figure 6.8.: Working layers for digital representations

> **Remark 6.11**
> *Note that a representation can be given in many forms such as models or networks, which we discussed in the lecture, but may also consist in lookup tables, databases or other digital types, but may also take a physical form like analog computers.*

Hence, not all digital forms can or should be applied for all management tasks but instead are specific to the respective management layer. These layers traditionally work on abstractions and aggregations of the physical system/process, i.e. a machine operator deals with specifics of the machine, a shift supervisor deals with a much larger working area, a plant manager oversees the performance of the plant, and a supply chain manager deals with intercompany transitions.

> **Remark 6.12**
> *As the last example indicates, the abstraction level is not directly proportional to the possibility of interaction with the system at hand.*

Apart from being able to consider the overall system/process, a digital version shows two additional advantages: For one, a digital version is easier to manipulate and assess in a virtual environment that the physical system in the real world. This allows for cost-effective exploration

of the behavior of the system under testing conditions. Secondly, the data from these experiments can be used to improve the system/process itself, e.g., maintenance, design, robustness etc. Hence, the application fields for digital representatives span (but are not limited to)

- Model validation with real data

- Decision support

- Identification of waste

- Optimization of overall performance

- Prediction of changes within the system/process

- Exploration of new application and revenue streams

Taking a more generic look at Figure 6.8, the idea of layered systems is to generate levels of abstraction of the system/process, which allows to zoom in on specific tasks at machine level, but also to see the big picture of requirements, interfaces, system design, analysis/tradeoff and tests put to the overall system. Here, we like to note that the model centric view provides a structure, which can be followed to properly derive a digital representation, which is suitable for the task it is created for. Additionally, such an approach allows to integrate generalizations in both directions, that is to generate KPIs and model abstractions for higher layers or to integrate realtime communication and control down to machine level.

> **Remark 6.13**
> *The big advantage of the latter approach is its integration property. It doesn't matter where the starting point of a digital representative is set, the approach allows full and complete integration of properties as well as traceability of properties through the diagrams.*

Table 6.5.: Advantages and disadvantages of SysML based digital representations

Advantage	Disadvantage
✓ Represent overall system/process	✗ Requires full taxonomy of SysML
✓ Allows full integration	✗ Induces costs to rework document centric view
✓ Allows aggregation and specification	
✓ Allows assessment in virtual space	

6.3. Cyber physical systems

Our starting point for a digital twin structure are so called cyber physical systems. In the literature [19], the term connects to time-sensitive functions with varying degree of interaction, which we already consolidated in our definition of a digital twin.

Definition 6.14 (Cyber physical system).

Consider a physical system/process together with KPIs and requirements to be given. Then we call a digital twin a cyber physical system, if the bidirectional flow is realtime capable to enforce the requirements and address the KPIs.

Different from a digital twin, a cyber physical system for one is a digital representation of a physical system/process. Secondly, a cyber physical system must contain a control unit and be able to access the sensors and actuators of the physical unit in realtime. And thirdly, the imposed control action shall improve given KPIs while upholding system/process requirements.

To integrate the latter into one particular problem, we first need to specify KPI in the setting of our system dynamics

$$
\begin{aligned}
\dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \\
\mathbf{y}(t) &= h(\mathbf{x}(t), \mathbf{u}(t), t).
\end{aligned}
\tag{1.1}
$$

Focusing on the state space, we typically speak of cost functions. These combined information on state and input of the system to quantify performance of the control.

Definition 6.15 (Cost function).

We call a key performance criterion given by a function $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ a *cost function*.

The value of a key performance criterion reveals a snapshot only, i.e. the evaluation at one time instant $t \in \mathcal{T}$. To obtain the performance, we need to evaluate it over the operating period of the system. Since by doing so we define a function of a function, this is referred to as a functional.

Definition 6.16 (Cost functional).

Consider a key performance criterion $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. Then we call

$$
J(\mathbf{x}_0, \mathbf{u}) := \int_0^\infty \ell(\mathbf{x}(t, \mathbf{x}_0, \mathbf{u}), \mathbf{u}(t)) dt
\tag{6.6}
$$

cost functional.

Remark 6.17

Note that the latter definition can be used across operational, tactical and strategic level whereas Definition 2.19 is applicable for planning, i.e. the strategic level only.

Integrating these components allows us to quantify not only operating points (Definition 5.4), but also the transients from the current state of the system to such an operating point.

Definition 6.18 (Optimal control problem).

Consider a system (1.1) and a cost functional (6.6). Then we call

$$\min \ J(\mathbf{x}_0, \mathbf{u}) \quad \text{over all } \mathbf{u} \in \mathcal{U} \tag{6.7}$$
$$\text{subject to } \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{x}(t) \in \mathbb{X}, \quad \mathbf{u}(t) \in \mathbb{U}$$

an *optimal control problem.*

Remark 6.19

Note that solving an optimal control problem belongs to control engineering. It is an essential part of automation engineering, yet only a part. Automation refers to address a problem so that it needs less human interaction. To this end, technology is used to plan and change existing devices and monitor the performance of the resulting device.

In order to solve problem (6.7) as a CPS on various levels, we require not only a control logic, but also computing, storage, communication and planning/modeling components, cf. Figure 6.9 for a generic sketch.

The terms in Figure 6.9 specify generic components only. In practice, there are several possibilities and methods, which can be chosen for each of these components, cf. Figure 6.10 for some of the current buzzwords.

Figure 6.10 sketches the indication that cyber physical systems give rise to big data, and thus promote the spread of optimization and AI methods. Here, the following interconnections can be observed:

- While the primary function of the Internet of Things is to automatically collect data, cyber physical systems map the dynamics to a digital representative to capture the system/process status and act accordingly.

- CPS maps the interactive operation and internal evolution of complex objects in virtual space to capture the required properties of the physical system.
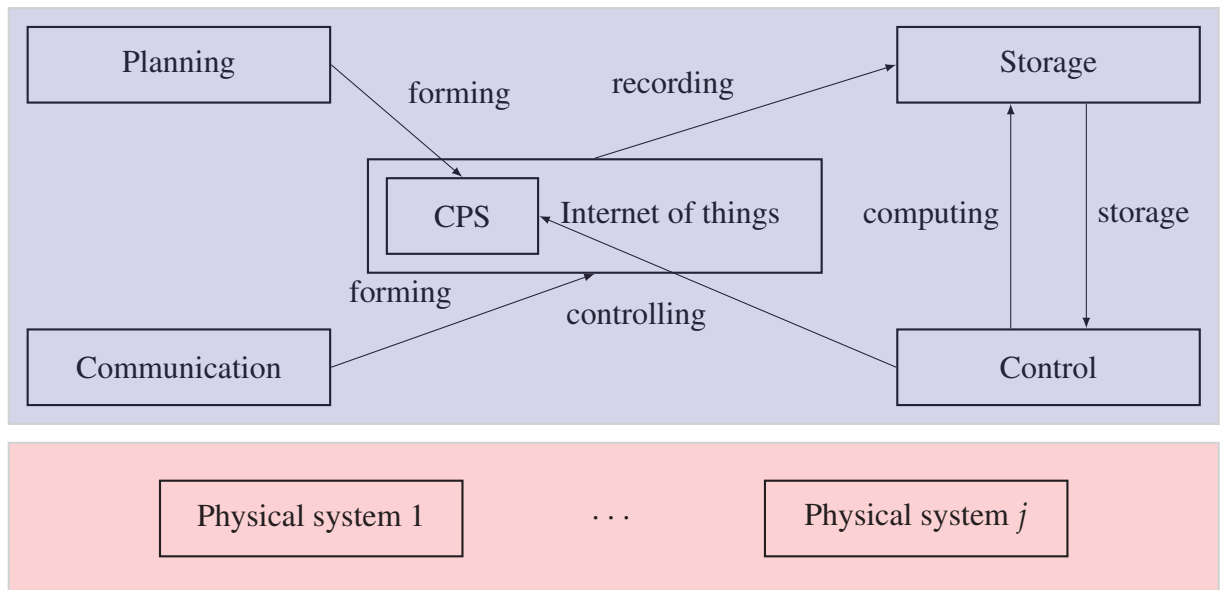
Figure 6.9.: Generic sketch of a CPS structure

- By collecting data, large storage basis arise which form Big Data. Combined with control, the data can be utilized to derive information regarding the physical system/process.

- To assess big data, new computing mechanisms like cloud computing or AI methods arise working on the derivation of information and secondly the derivation of controls.

> **Remark 6.20**
> *Although being initially connected, developments of data driven and AI based methods have become self sustained research fields.*

Cyber physical systems are a means of interaction between the physical and the digital system. CPS themselves can be structured in five levels, cf. Figure 6.11.

On the very basis, we used CPS to make tacit knowledge of humans, machines, methods and ideas explicit, and then embed this knowledge into software, software into chips, chips into hardware and hardware into objects (physical devices).

Table 6.6.: Advantages and disadvantages of cyber physical systems

Advantage	Disadvantage
✓ Represent overall system/process	✗ Requires realtime computation
✓ Allows for layers of methods	✗ Requires realtime communication
	Continued on next page

Table 6.6 – continued from previous page

Advantage	Disadvantage
✓ Maps digital representation physically	✗ May require big data analytics

6.4. Industrial cloud platform

The idea of a cloud platform is to provide costumers access to services such as computing, storage, software or data, which are available anytime and anywhere. To properly characterize the latter, we first have to define basic terms of this concept. Again, we follow the denomination of the Industrial Internet Consortium [14] and the National Institute of Standards and Technology [19]. As the cloud platform shall provide access, some kind of an interface is required. More generic:

Definition 6.21 (Interface).
Consider a system/process and a respective digital representative to be given. An interface is a named set of operations to read and set data of the system/process or its digital representative.

An interface therefore allows interaction of both the system and its cyber version with the outside, but also allows interaction between both internally. Based on interfaces, the access can be utilized to fulfill, e.g., KPIs or constraints.

Definition 6.22 (Service).
A service is a distinct part of the functionality that is provided through interfaces.

The latter term allows us to apply services not only to cyber physical systems, which rely on digital twins, but also to digital shadows and digital models. Hence, the application range of services is not limited by the requirement of realtime connectivity and feeding back information. One way to provide such services is via clouds. Regarding the implementation of the latter, both self-hosted, managed, hybrid or community systems can be used, which again depend on requirements put on the service.

Definition 6.23 (Industrial cloud platform).
Suppose a system/process and a digital representative to be given. Then an industrial cloud platform is a service for the digital representative, which provides an elastic execution environment of resources involving multiple stakeholders and providing a metered service at multiple granularities for a specific level of quality.
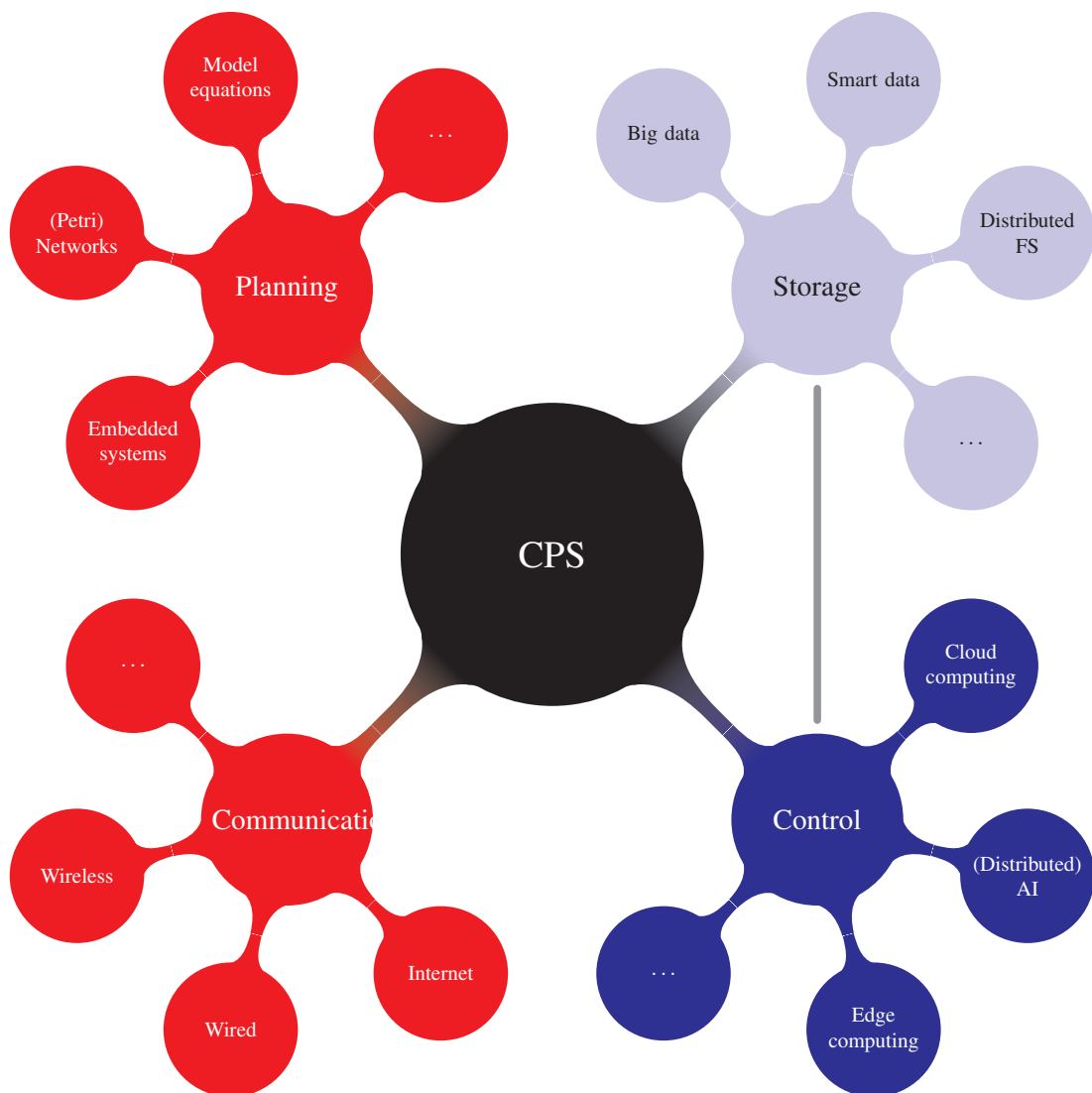
Figure 6.10.: Possibilities for CPS components

> **Remark 6.24**
> *Here, we utilized the European formulation [3], which explicitly integrates quality of service and metering. In contrast to that, the US version addresses IT configurability regarding resources.*

Hence, an industrial cloud platform is the means to provide a service, but additionally addresses

- cloud technologies as a core,

- user-centered solutions as it is designed for multiple stakeholders,

- knowledge support by specific level of quality, and

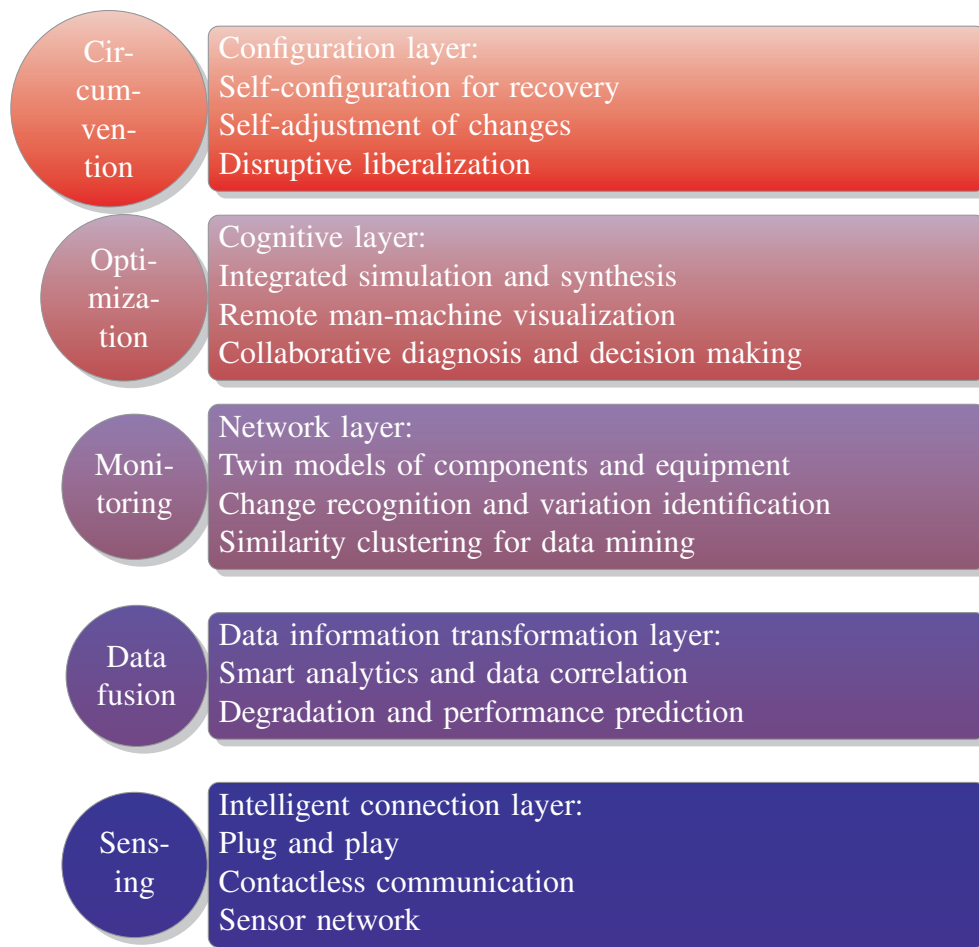- collaboration options by means of multiple granularities.

Figure 6.11.: Structure levels of cyber physical systems

There are several classes of services as indicated in Figure 6.12, which can be addressed using such an option. These services utilize the characteristics of industrial clouds and can be realized via one of the deployment possibilities mentioned before.

Here, we subdivide these classes according to the usage of software and hardware, which leads to the following:

Corollary 6.25 (SaaS, IaaS, PaaS)

Consider a system/process, its digital representative and an industrial cloud platform to be given. Then providing

- *access to delivery or licensing of software is a service called Software-as-a-Service,*

- *utilization of hardware such as compute, storage and networking resources is a service called Infrastructure-as-a-Service, and*

- *access to maintained soft- and hardware is a service called Platform-as-a-Service.*
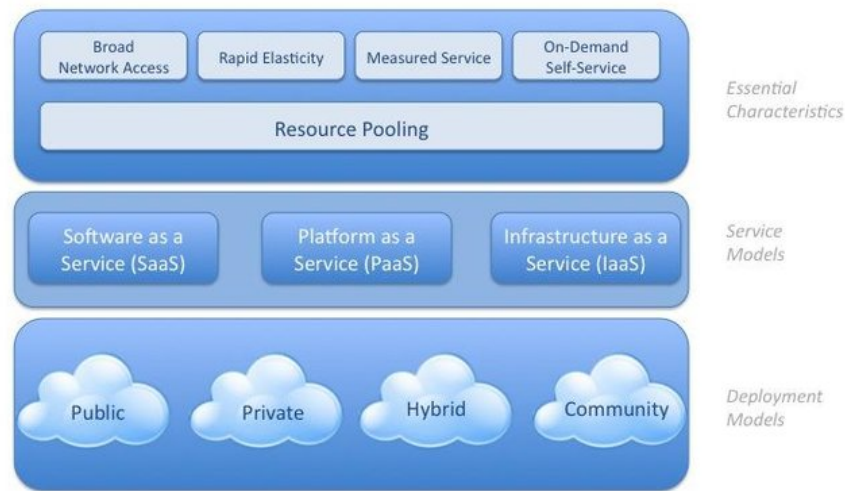
Figure 6.12.: Visual model of industrial cloud

> **Remark 6.26**
> *Note that the latter services are not independent from a digital representative and of the system/process.*

In any of the above cases, the users of such services can focus on their main task while outsourcing buildup and maintenance of the service components and being able to scale the latter freely.

Table 6.7.: Advantages and disadvantages of industrial cloud

Advantage	Disadvantage
✓ Separates core from auxiliary tasks	✗ Depends on use case
✓ Utilizes shared resources	✗ Depends on security

6.5. Industrial internet

Industrial clouds are core components of systems/processes, which are to be modeled, monitored or controlled as a whole. As such, any realization focuses on particular applications. This means that each implementation is driven by one use case. From Corollary 6.25, we have already seen that services based on industrial clouds are not limited to specific use cases, but may allow reutilization of components.

The industrial internet is the generalization of this reutilization idea. It aims to abstract from the use cases and to realize the modeling and reuse of industrial technology, experience and knowledge. The goal of the industrial internet is to finally form a ecology with resource enrichment and collaborative participation.

Definition 6.27 (Industrial internet).
Suppose the set of all industrial clouds to be given and any interfaces to the underlying systems/processes to be removed. Then any element of the powerset is called industrial internet.

Note that by definition, the industrial internet is not unique but depends on the choice of combined industrial clouds. The combination of clouds additionally stresses the necessity of standardization of systems/processes, their digital representations and all of their components.

The downside of such an idea is that the industrial internet is completely decoupled from all systems/processes, from which the respective data or digital representative originated.

While being almost trivial, the following is the foundation of a vast number of companies acting in the field of digitalization:

Corollary 6.28
Every industrial internet is a service.

Table 6.8.: Advantages and disadvantages of industrial internet

Advantage	Disadvantage
✓ Allows aggregation of use cases	✗ Looses connection to reality
✓ Reutilizes methods and data	✗ Misses uniqueness of service

BIBLIOGRAPHY

[1] BLONDEL, V.D. ; GUILLAUME, J.-L. ; LAMBIOTTE, R. ; LEFEBVRE, E.: Fast unfolding of communities in large networks. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008 (2008), Nr. 10, S. P10008. http://dx.doi.org/10.1088/1742-5468/2008/10/P10008. – DOI 10.1088/1742–5468/2008/10/P10008

[2] EMMONS, S. ; KOBOUROV, S. ; GALLANT, M. ; BÖRNER, K.: Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. In: *PLOS ONE* 11 (2016), S. 1–18. http://dx.doi.org/10.1371/journal.pone.0159161. – DOI 10.1371/journal.pone.0159161

[3] EXPERT GROUP ON CLOUD COMPUTING: *The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010*. Commission of the European Communities, 2010

[4] INTERNATIONAL ELECTROTECHNICAL COMMISSION: *IEC 61512-4:2009 Batch control*. IEC, 2009

[5] INTERNATIONAL ELECTROTECHNICAL COMMISSION: *IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems*. IEC, 2010

[6] INTERNATIONAL ELECTROTECHNICAL COMMISSION: *IEC 61131:2013 Programmable controllers*. IEC, 2013

[7] INTERNATIONAL ELECTROTECHNICAL COMMISSION: *ISO/IEC 81346:2022 Industrial systems, installations and equipment and industrial products – structuring principles and reference designations*. IEC, 2022

[8] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO 7498:1994 Information Technology - Open Systems Interconnection - Basis Reference Model*. Beuth, 1994

[9] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO 9001:2015 Quality management systems - Requirements*. Beuth, 2015

[10] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO/IEC 27001:2017 Information technology - Security techniques - Information security management systems - Requirements*. ISO, 2017

[11] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO 9241:2019 Ergonomics of human-system interaction*. ISO, 2019

[12] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO/IEC Directives, Part 2*. International Organization for Standardization, 2021

[13] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO/IEC DIS 27032:2022 Cybersecurity - Guidelines for Internet security*. ISO, 2022

[14] KARMARKAR, A. ; BUCHHEIT, M.: *The Industrial Internet of Things Volume G8: Vocabulary*. Industrial Internet Consortium, 2017

[15] LAI, C.: *Intelligent Manufacturing*. Springer, 2022

[16] LANGMANN, C. ; TURI, D.: *Robotic process automation – Digitalisierung und Automatisierung von Prozessen*. Springer, 2020

[17] LUNZE, J.: *Automatisierungstechnik*. 5. Auflage. DeGruyter, 2020

[18] MURATA, T.: Petri Nets: Properties, Analysis and Applications. In: *Proceedings of the IEEE* 77 (1989), Nr. 4, S. 541–580

[19] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *Framework for Cyber-Physical Systems*. National Institute of Standards and Technology, 2015

[20] NEUMANN, K. ; MORLOCK, M.: *Operations Research*. Hanser, 2002

[21] NORMUNG E. V., Deutsches I.: *DIN V 19233:1998 Control technology - Process automation - Automation with process computer systems, definitions*. Beuth, 1998

[22] NORMUNG E. V., Deutsches I.: *DIN IEC 60050-351 Internationales Elektrotechnisches Wörterbuch Teil 351: Leittechnik (IEC 60050-351:2014-09)*. Beuth, 2014

[23] PLENK, V.: *Grundlagen der Automatisierungstechnik kompakt*. Springer, 2019

[24] STJEPANDIC, J. ; SOMMER, M. ; DENKENA, B.: *DigiTwin: An approach for production process optimization in a built environment*. Springer, 2022

[25] WIKIPEDIA: *CAN bus.* https://en.wikipedia.org/wiki/CAN_bus, 2023. – Accessed: 2023-01-03

Jürgen Pannek
Institute for Intermodal Transport and Logistic Systems
Hermann-Blenck-Str. 42
38519 Braunschweig