Universität Basel

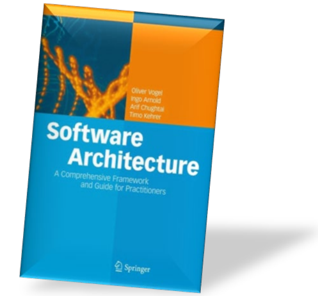
Software Architecture

Basic Terms and Concepts

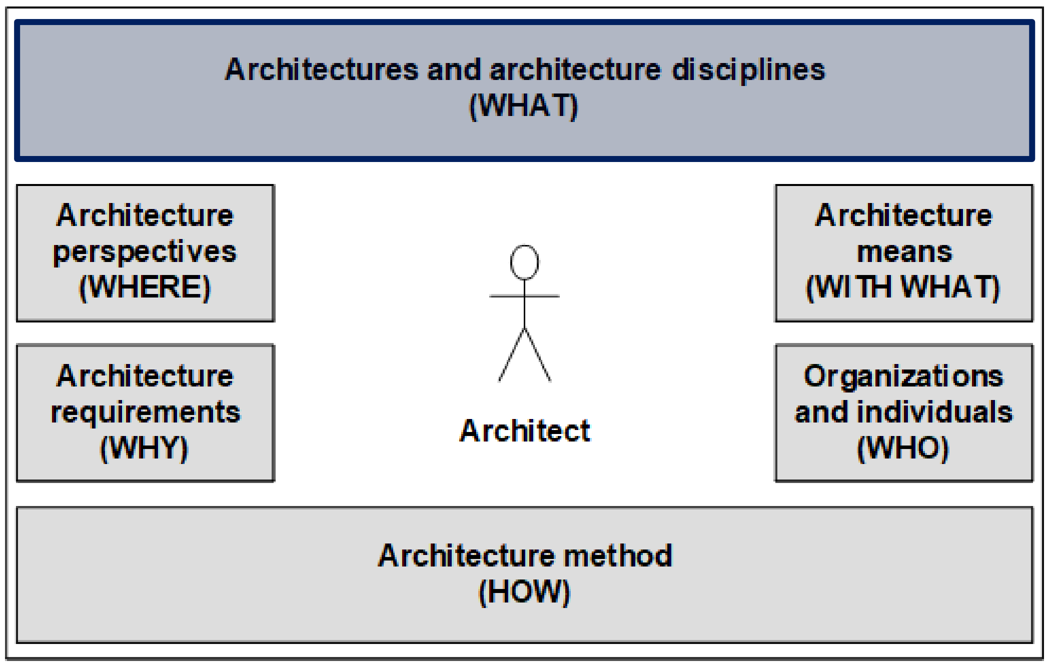
BSc

Ingo Arnold

Lecture Opening
Architecture Orientation Framework



Architecture WHAT lays a foundation for any architecture considerations by **introducing basic concepts and terms** as well as the context in which software architecture takes place.



Lecture Opening
Motivation

This lecture focuses on the topic of **architecture**. Before we dive deeper into this topic, it is important for the understanding of software architecture to be able to distinguish it from other **architecture disciplines** and to understand what contributions architecture disciplines make along the **enterprise value chain**.

We will distinguish architecture as a **strategic planning, portfolio & orientation discipline** from architecture as a **transformation & solution discipline**. Another feature in distinguishing different architecture disciplines is granularity of assets considered, transversality and horizon of consideration.

We will **sharpen the concept of Software Architecture**. This means that we will specify software architecture beyond the spectrum of other architecture disciplines. To do this, we will look at standard definitions and derive the core aspects of software architecture for us from these.

In addition, in the introductory lecture we will define **concepts and terms** that are fundamental to the understanding of architecture.

Lecture Opening
Learning Objectives

You …

- know the **context** in which architecture contributes along the enterprise value chain
- will be able to name the three central **architecture disciplines** and distinguish their respective contributions, as well as explain how they complement each other to form the overall contribution of an architecture function
- can name crucial components in the **definition of software architecture** and are able to explain them to your fellow students.
- know **basic architecture concepts** and terms

Universität Basel

Lecture Agenda

- **Classical Architecture**
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture
- Software Architecture

Classical Architecture
Overview

To approach architecture, we first take a look at **classical architecture**[1], whose starting point is the design of structures, buildings, cities, and similar urban conglomerates.

Classical architecture has traditionally been understood as both an **art** and a **science** concerned with the design and construction of buildings, and thus the entire process **from planning to realization**.

[1] [Vogel, Arnold et al 2011]

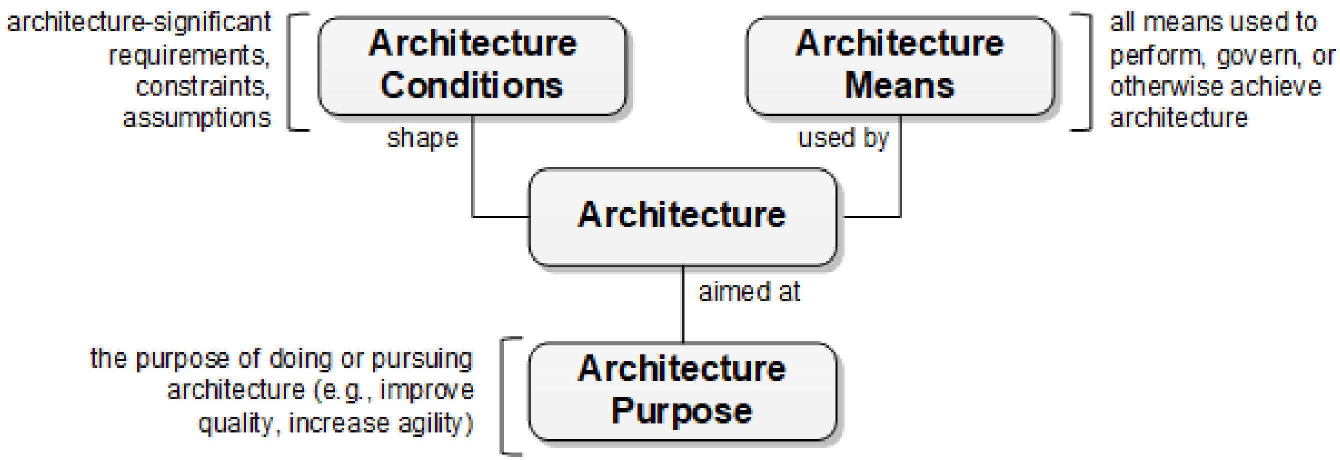
Universität Basel

Classical Architecture

Architecture conditions, means, and purpose

Classical architecture is concerned with the **ordering structure of parts** of an intended **whole**.

In doing so, architecture pursues an **architecture purpose** that seeks to address **architecture conditions** (e.g., the desire for functional, affordable housing) using given **architecture means** (e.g., building materials, tools, techniques, and methods).



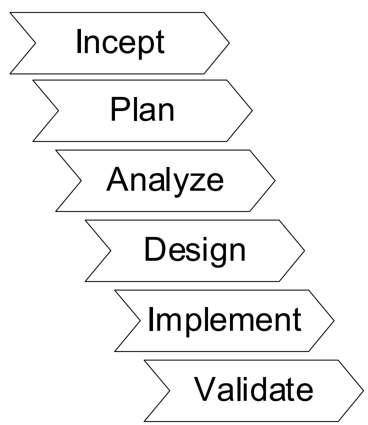
Universität Basel

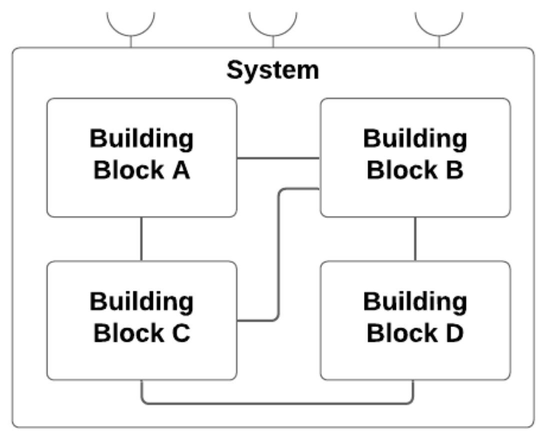
Classical Architecture
Process and process result

The classical concept of architecture encompasses both the systematic **process of architecture** planning, design, and implementation and the **result of that architecture process**.

Architecture = Architecture Process + Architecture Process Result

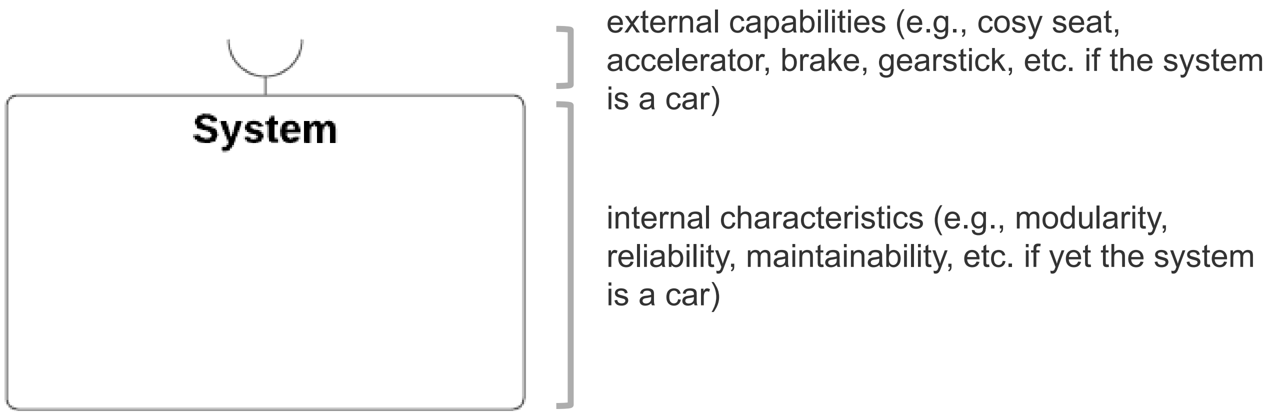
architecture **is** the process of governing and performing architecture **plus** the results and outcomes produced by that process





Classical Architecture
Purpose

Architecture designs **systems so that they have desired external and internal capabilities**, characteristics, or properties – such that the systems can fulfill their purpose.



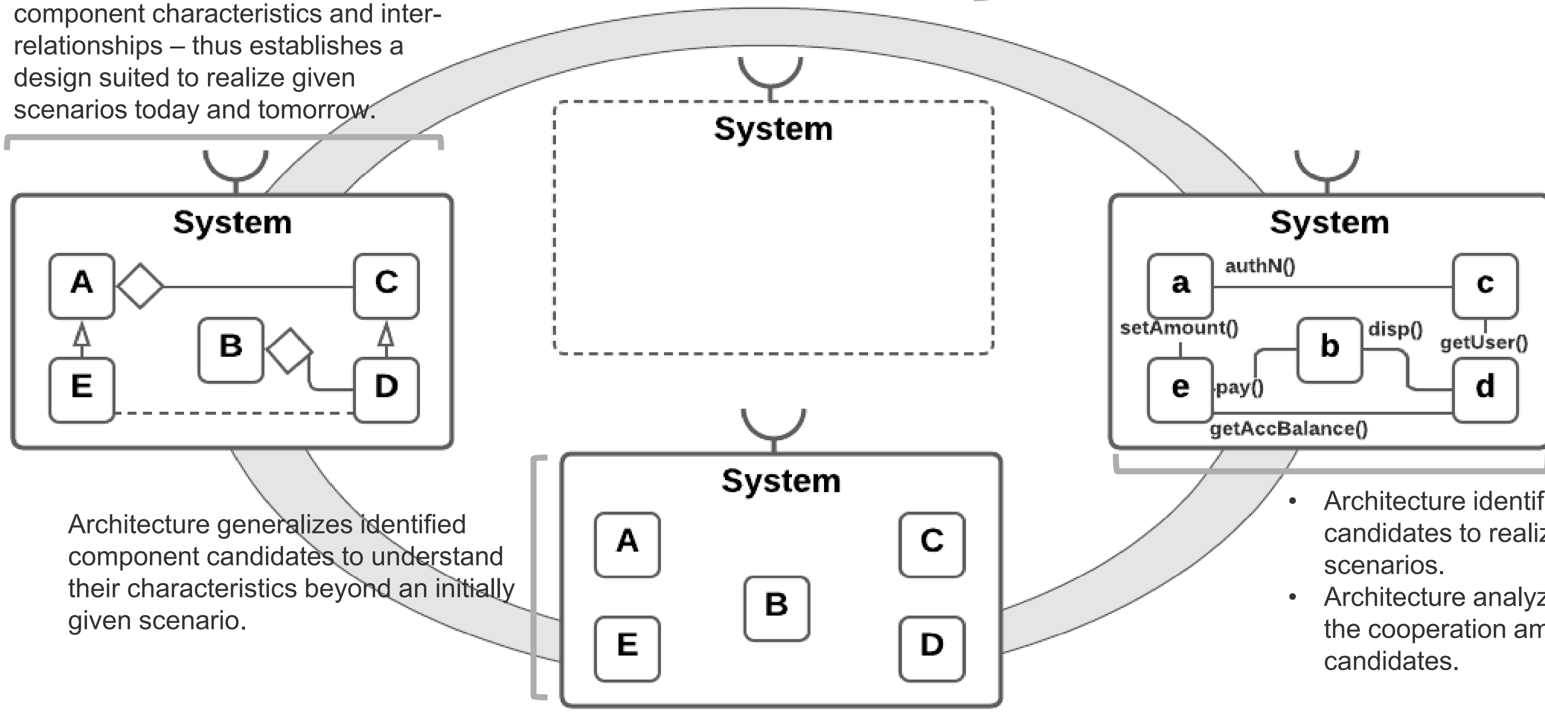
Classical Architecture
Purpose

Architecture designs **systems so that they have desired external and internal capabilities**, characteristics, or properties – such that the systems can fulfill their purpose.

1. user arrives at ATM
2. ATM authenticates user
3. user selecst amount to withdraw
4. amount is dispensed in cash by ATM
5. user takes cash and leaves the ATM

Architecture analyzes the scenarios a system is supposed to realize and fulfill.

Architecture identifies general component characteristics and inter-relationships – thus establishes a design suited to realize given scenarios today and tomorrow.

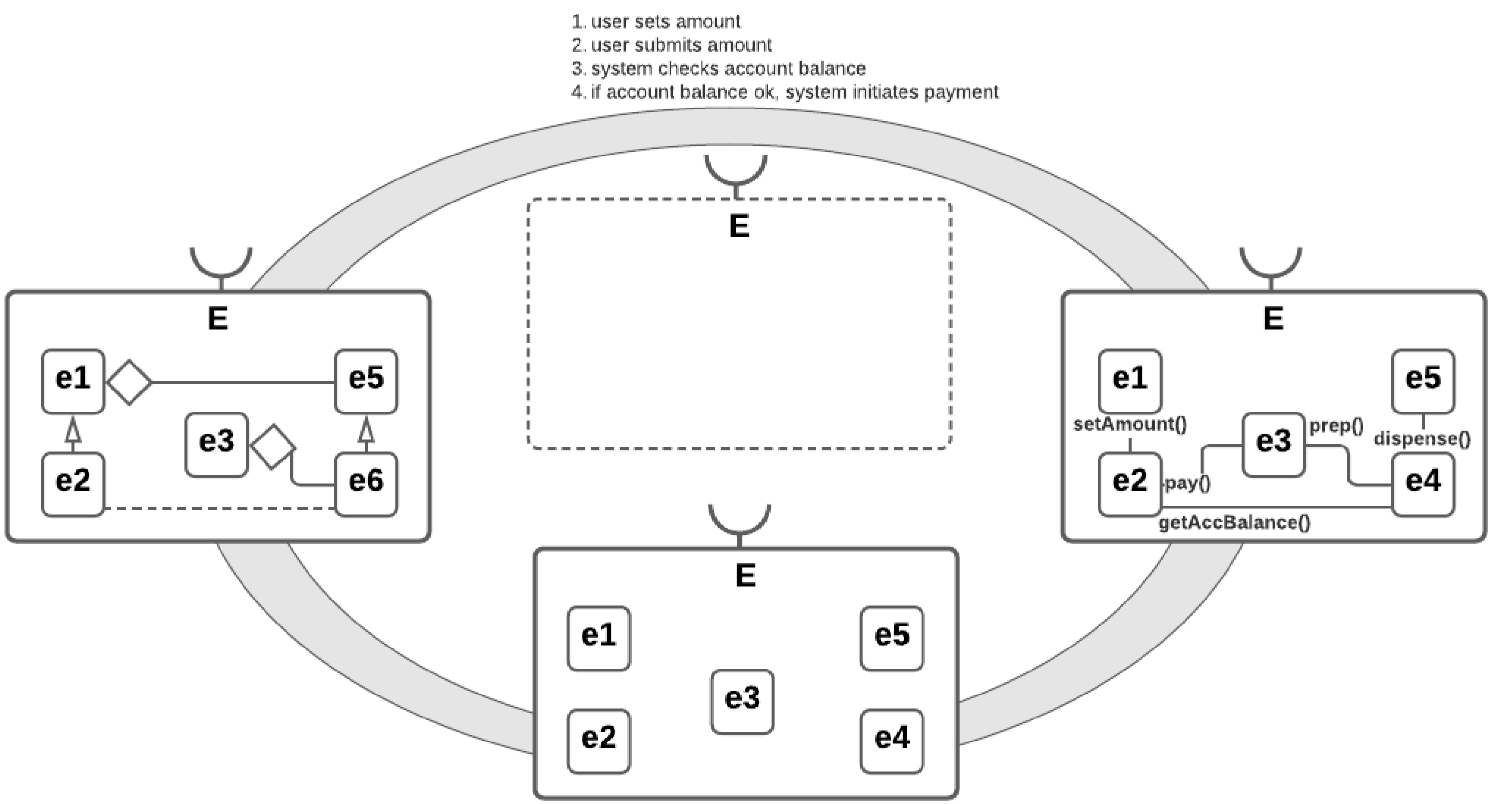


Architecture generalizes identified component candidates to understand their characteristics beyond an initially given scenario.

- Architecture identifies component candidates to realize the given scenarios.
- Architecture analyzes and designs the cooperation among component candidates.

Classical Architecture
Purpose

… and this **general approach to architecting a system is continued over and over** again – until the whole and its parts are fully constituted.



Universität
Basel

Lecture Agenda

- Classical Architecture
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture
- Software Architecture

Enterprise Operating Model
Architecture function

Architecture does not take place on a greenfield (i.e. without context) and never without reason. Furthermore, architecture must be implemented organizationally, for which architecture functions are responsible.

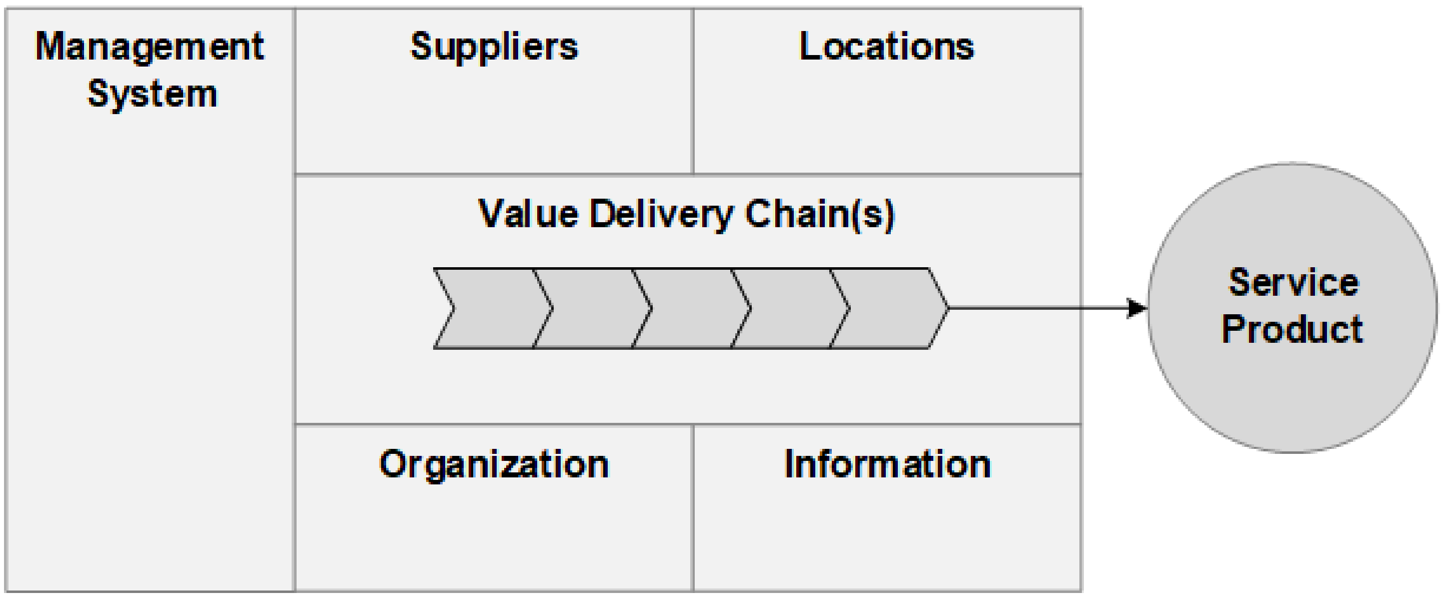
An **architecture function** establishes architecture capabilities organizationally — it is a sociotechnical system that realizes architecture disciplines by integrating their contributions into an **enterprise operating model**.

An **operating model** specifies and captures how the inner workings of an organization are designed to deliver value to its customers. It describes how an organization collaborates — but limits itself to the essentials of organizational cooperation.

Universität
Basel

Enterprise Operating Model
Operating model canvas

The **operating model canvas** is a template created for designing and discussing operating models ([Campbell 2017]). Embedded in an operating model are value delivery chains.



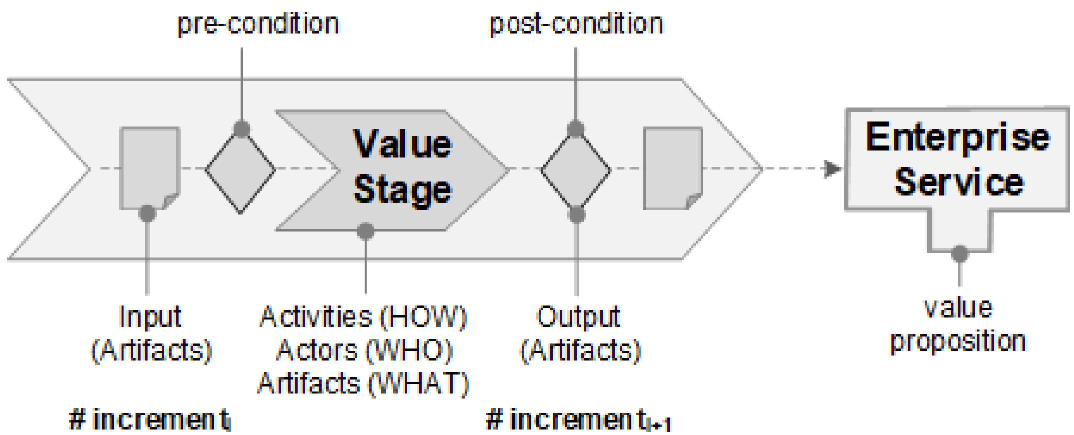
Lecture Agenda

- Classical Architecture
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture

Value Delivery Chain
Overview

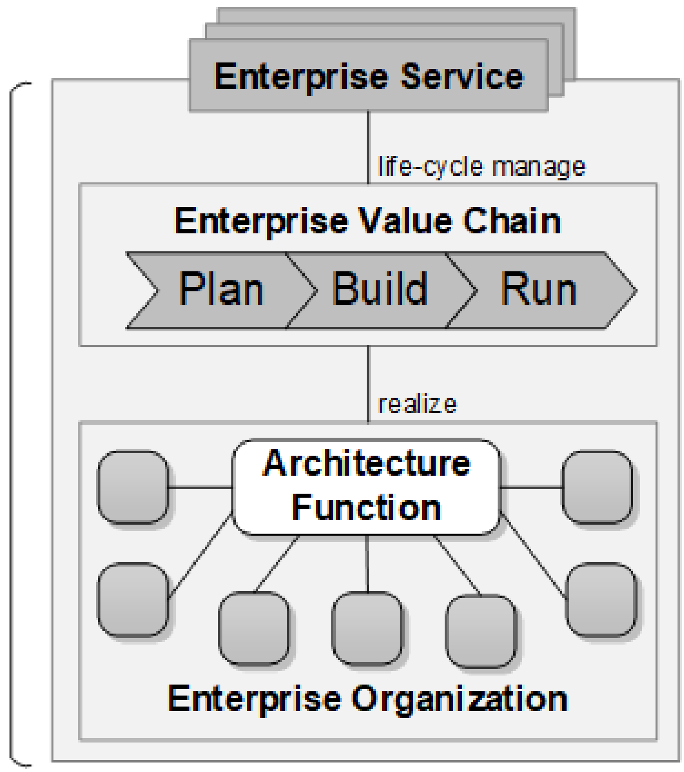
A **value delivery chain**, according to Michael Porter [Porter 2004], is a set of activities that a company in a particular industry performs to deliver a valuable product (i.e., goods or services) to the market.

A value chain views an **organization** as a set of processes at different levels of process aggregation. In addition, an organization is viewed as a system that is decomposed into subsystems, with each subsystem having its inputs, transformation processes, and outputs.



Value Delivery Chain
Architecture function and enterprise organizations

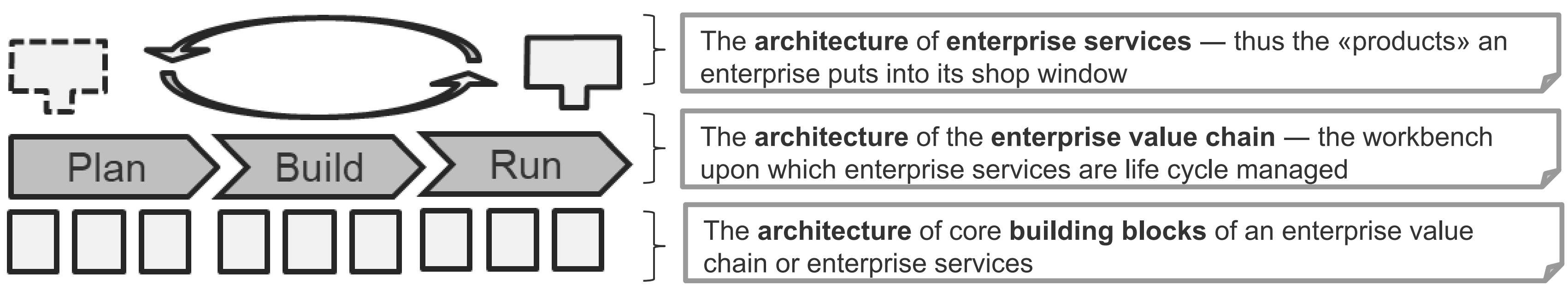
An **architecture function** is a business organization and thus a subsystem of the value chain. It cooperates with other business organizations and makes its contributions along the entire value chain ([Arnold 2021]).



Value Delivery Chain
Architecture for the enterprise services life-cycle

On the one hand, an architecture function contributes to the **establishment of the value chain** and its **building blocks** (for example, business applications or technical platforms).

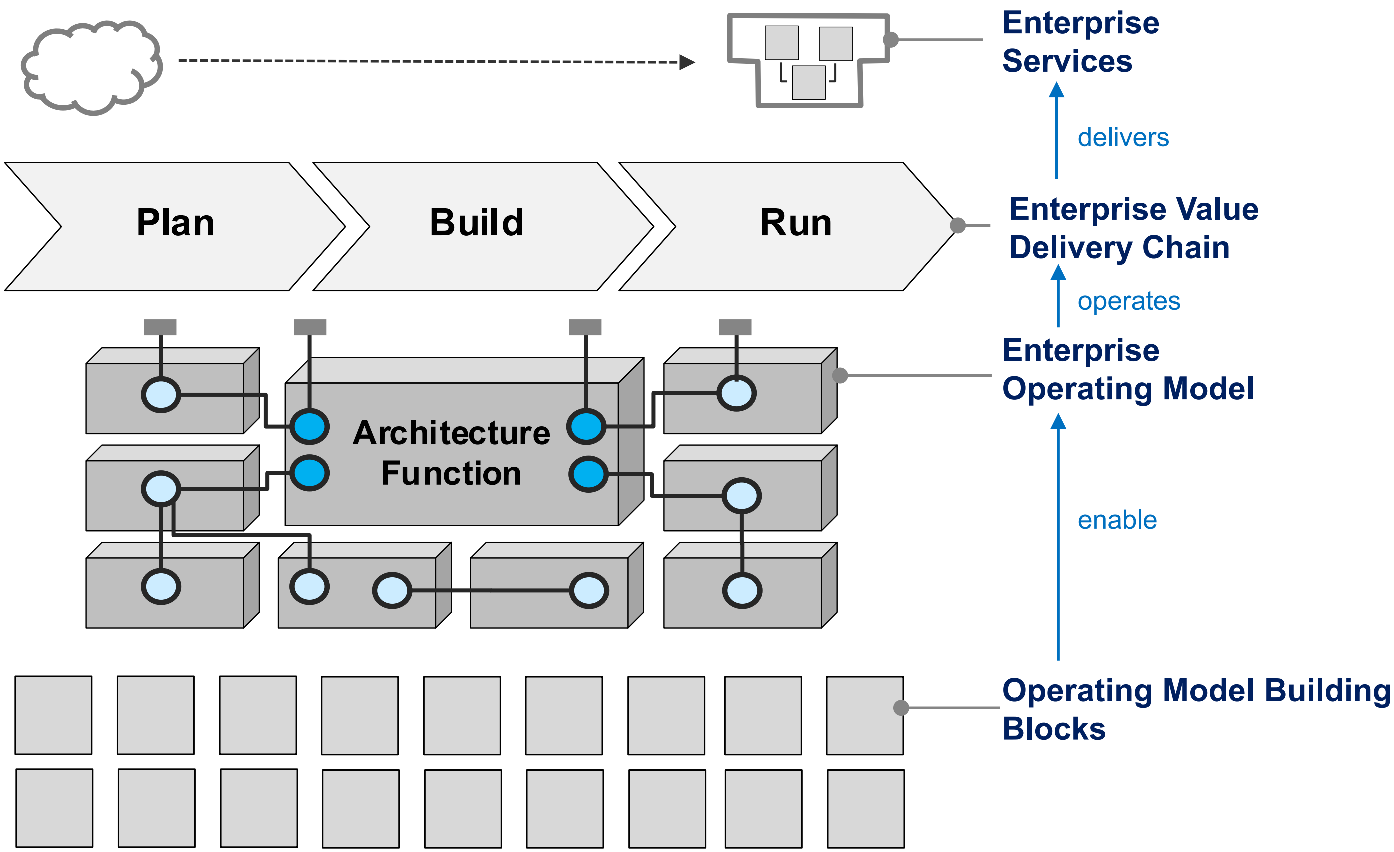
On the other hand, an architecture function contributes to the planning, creation, delivery and operation of **enterprise services**, which in turn are the value chain output.



Universität Basel

Value Delivery Chain
Architecture for the enterprise services life-cycle

Perspective on the whole mechanism and the cooperation of its parts.



Lecture Agenda

- Classical Architecture
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture
- Software Architecture

Architecture Disciplines
Overview

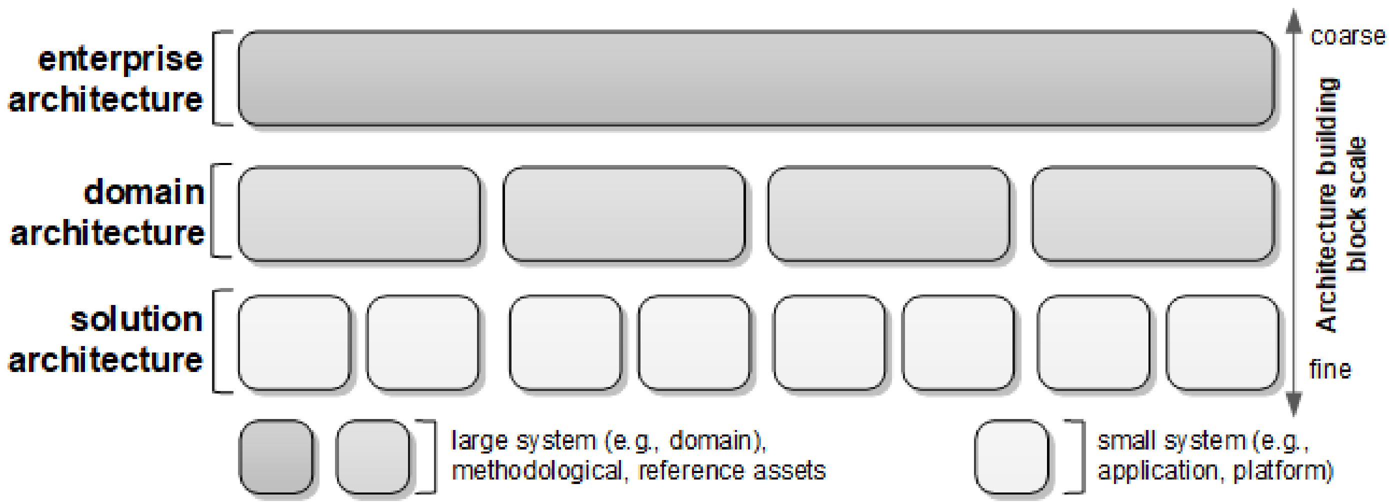
Although there are a myriad of **architecture disciplines** in practice, we will limit ourselves to distinguishing the three architecture disciplines of **enterprise**, **domain** and **software architecture** (aka: solution architecture).

On the one hand, we differentiate architecture disciplines along the **granularity** of the considered systems as well as the **planning horizon** that the respective discipline takes.

On the other hand, we distinguish architecture disciplines along their specific **contributions along an enterprise value chain**.

Architecture Disciplines
Architecture granularity

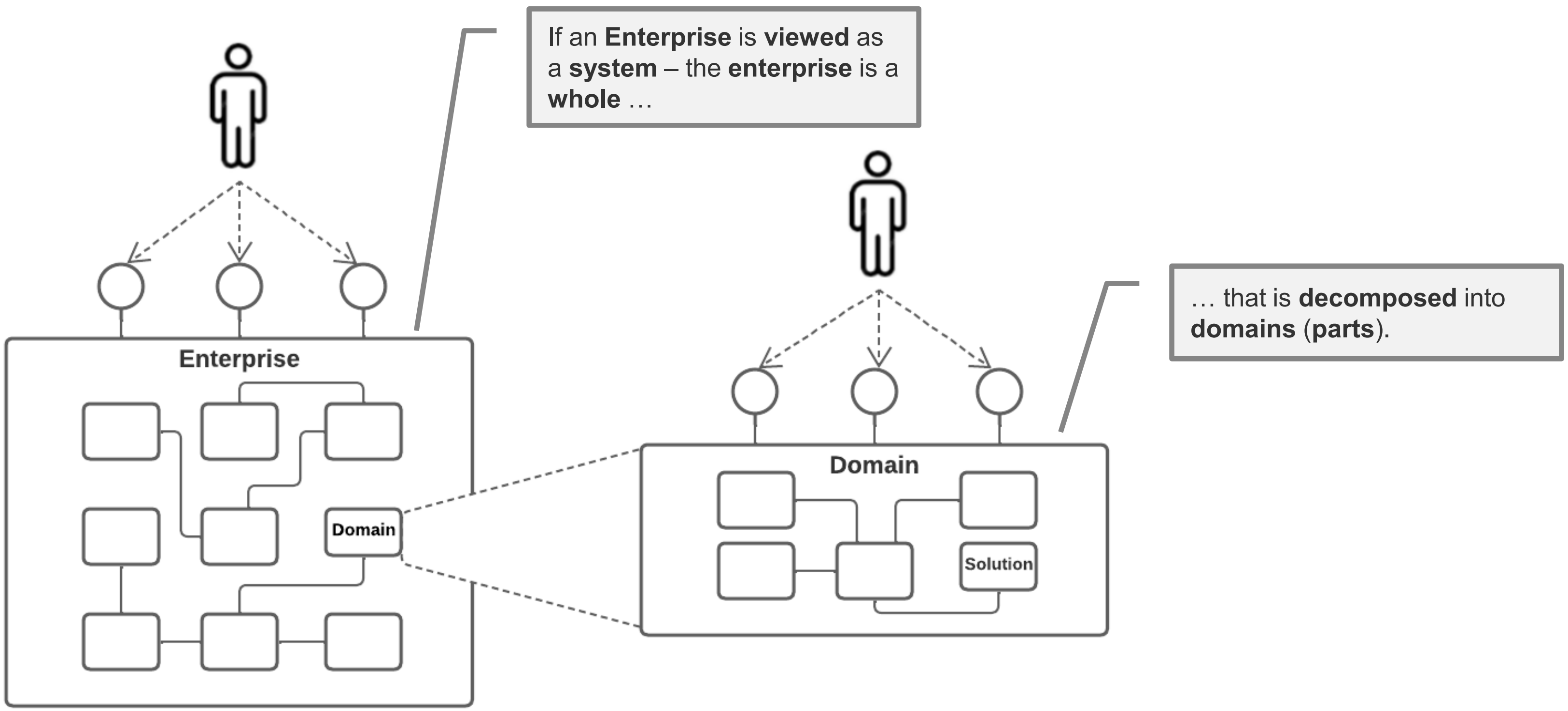
Differentiating **architecture disciplines** regarding the **granularity** of considered systems.



Universität Basel

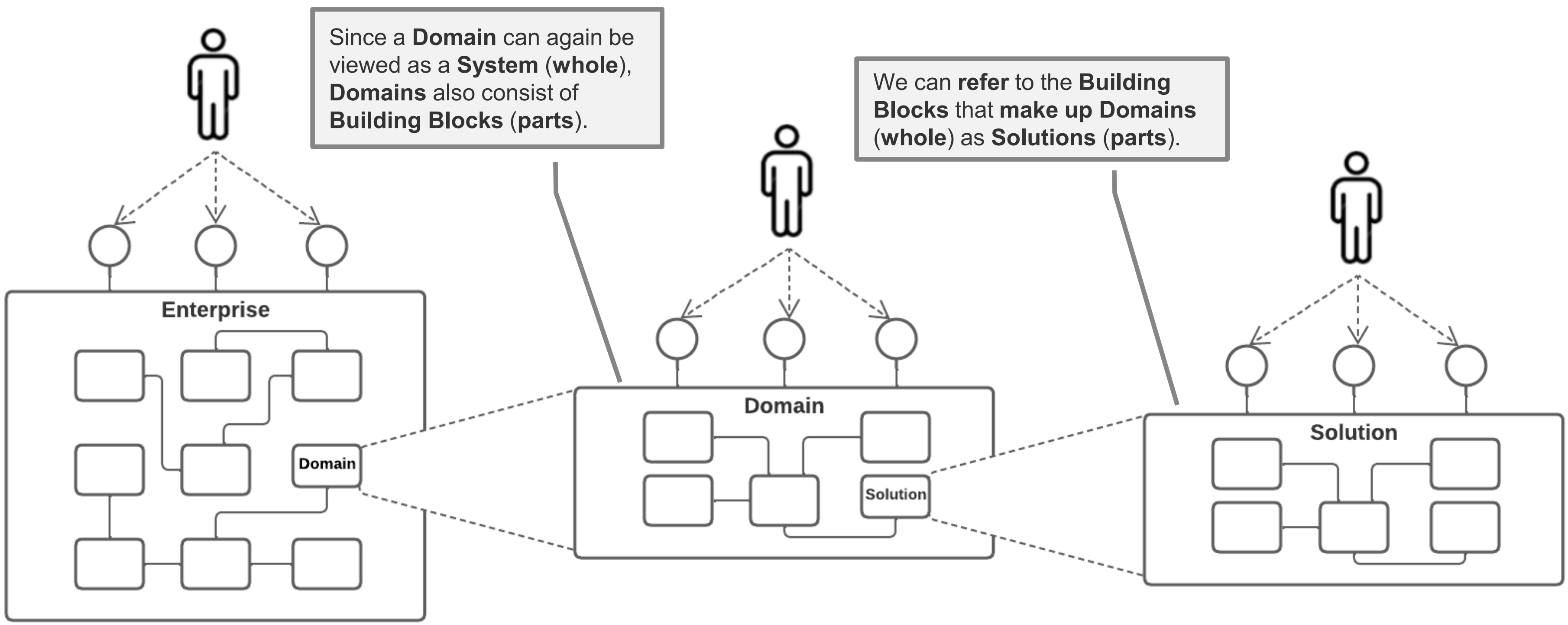
Architecture Disciplines
Architecture granularity

Differentiating **architecture disciplines** regarding the **granularity** of considered systems.



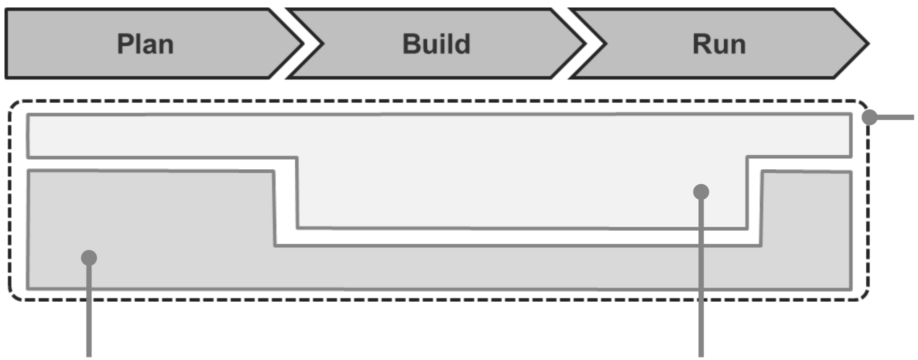
Architecture Disciplines
Architecture granularity

Differentiating **architecture disciplines** regarding the **granularity** of considered systems.



Architecture Disciplines
Enterprise, domain, and software architecture

Differentiating **architecture disciplines** regarding their **contributions along the enterprise value chain**



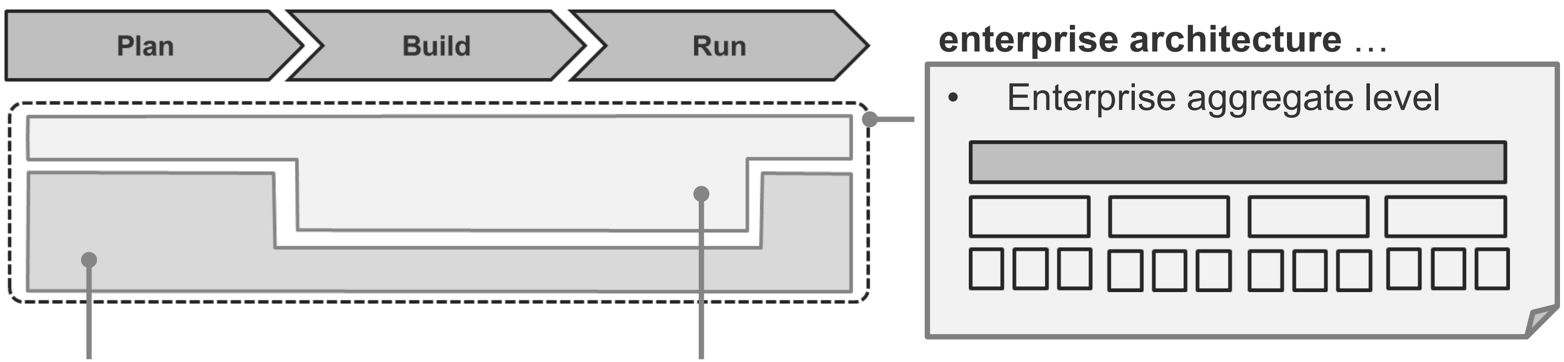
enterprise architecture …
- ensures alignment and integration between domain and software architecture
- Equips disciplines with methods, and policies

domain architecture …
- supports the process of making decisions as to which systems require improvement
- establishes transparency, overview, and orientation as preconditions for good decision making
- maintains as-is, and to-be architecture plans, roadmaps, and standards

software architecture …
- assumes a situation which is not ideal (i.e. *problem*) and therefore requires improvement
- usually means a new system or change to existing system is needed, where the established or refactored system improves the situation — i.e., is a *solution* addressing the given problem

Architecture Disciplines
Enterprise, domain, and software architecture

Differentiating **architecture disciplines** regarding their **contributions along the enterprise value chain**



domain architecture …

- Identify redundancies, systematic deficiencies, gaps, consolidation opportunities, etc.
- Analyse impact of planned changes
- Translate technological innovations into business opportunities
- Ensure strategic alignment along enterprise value chain

software architecture …

- Ensure requirements are addressed by elaborated solution (e.g. security, performance, availability, modifiability, etc.)
- Develop architecture alternatives and select most adequate alternative
- Elaborate software architecture and ensure conformity of implementation

Architecture Disciplines
Planning versus transformative architecture

While **domain architecture** is responsible for making the right directional decisions, **software architecture** is responsible for implementing them correctly.

software architecture
domain architecture

Do the right thing

Do the thing right

Lecture Agenda

- Classical Architecture
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture
- Software Architecture

System
Overview

After introducing the context in which an architecture function makes its enterprise, domain, as well as software architecture contributions, we consider system as another central concept, or term.

The term **system** refers to a very generic concept. A systemic perspective allows us to view, investigate, conceptualize as well as receive very different concepts or *things* in a unified way (i.e., in the sense of systems).

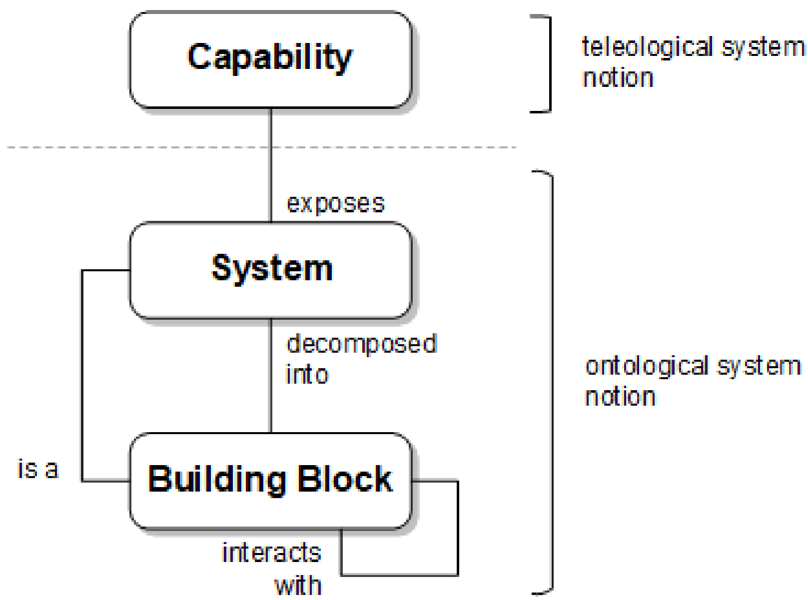
For example, a family and a football team are **social systems**, while a car or a washing machine are **technical systems**.

Enterprise services, as introduced above, we could call **digital systems**. An enterprise organization (e.g., an architecture function), on the other hand, we would call a **sociotechnical system**.

System
Teleological versus ontological system notion

Two **system notions** are distinguished. One is the **teleological system notion**, which deals with the external behavior of a system.

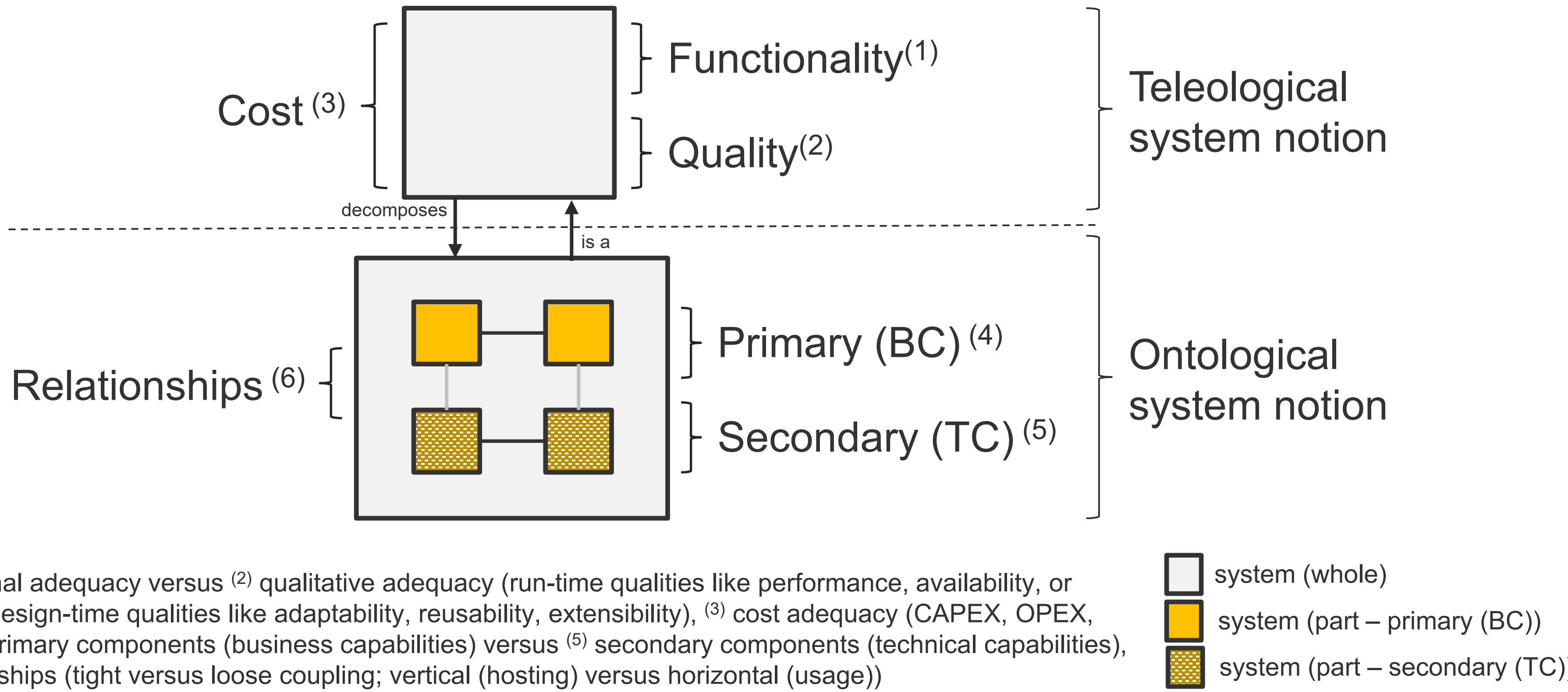
On the other hand the **ontological system notion**, which deals with the construction of a system — i.e. with its composition, environment, structure and production



Universität Basel

System
System capabilities

From a teleological point of view, systems offer **functional** as well as **quality-**related attributes. System functions and qualities together accomplish the purpose or represent the utility of a system — they realize its **capabilities**.



(1) functional adequacy versus (2) qualitative adequacy (run-time qualities like performance, availability, or security; design-time qualities like adaptability, reusability, extensibility), (3) cost adequacy (CAPEX, OPEX, time), (4) primary components (business capabilities) versus (5) secondary components (technical capabilities), (6) relationships (tight versus loose coupling; vertical (hosting) versus horizontal (usage))

Universität Basel

System
Primary versus secondary capabilities

Systems are **wholes** of corresponding **parts**. Among the system parts we distinguish **primary** (i.e., business (BC)) from **secondary** (i.e., technical (TC)).

System parts can be considered as systems (i.e., as wholes) themselves. Furthermore, system parts interact with each other.

While business-oriented building blocks interact with their peers, technical building blocks interact accordingly with technical building blocks (**cooperation relationship**).

Business building blocks are operated on the basis of technical building blocks (**placement relationship**).

(1) functional adequacy versus (2) qualitative adequacy (run-time qualities like performance, availability, or security; design-time qualities like adaptability, reusability, extensibility), (3) primary components (business capabilities) versus (5) secondary components (technical capabilities), (4) relationships (vertical (placement) versus horizontal (cooperation))

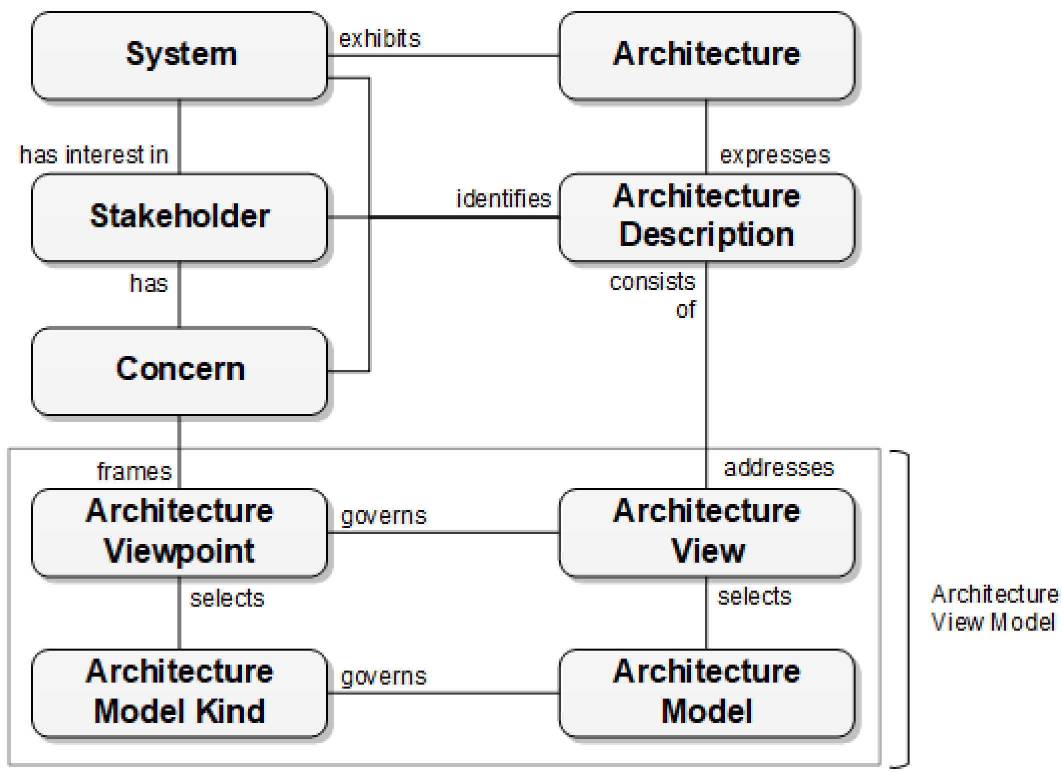
Universität Basel

Lecture Agenda

- Classical Architecture
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture
- Software Architecture

System Architecture
Architecture meta model

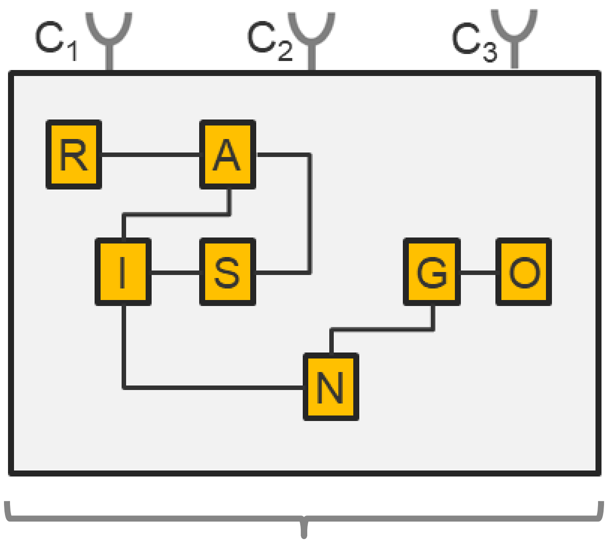
The IEEE Computer Society proposes a meta model that explicates the relationships between **system**, **architecture**, **architecture description**, **architecture views**, **and models** [IEEE 2000]



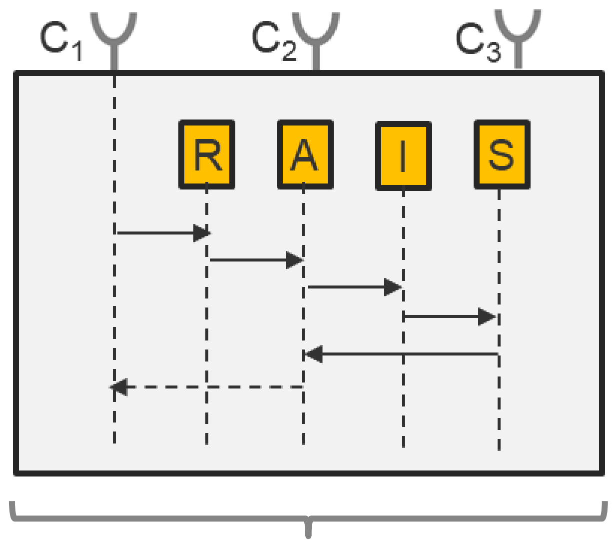
System Architecture

Dynamic versus static system architecture

System architecture defines how a (whole) system realizes its externally visible properties (e.g. functional and quality attributes) on the basis of its parts — i.e. how its parts relate to each other statically and dynamically.



Static System Architecture
(whole-part, generalization-specialization relationships)

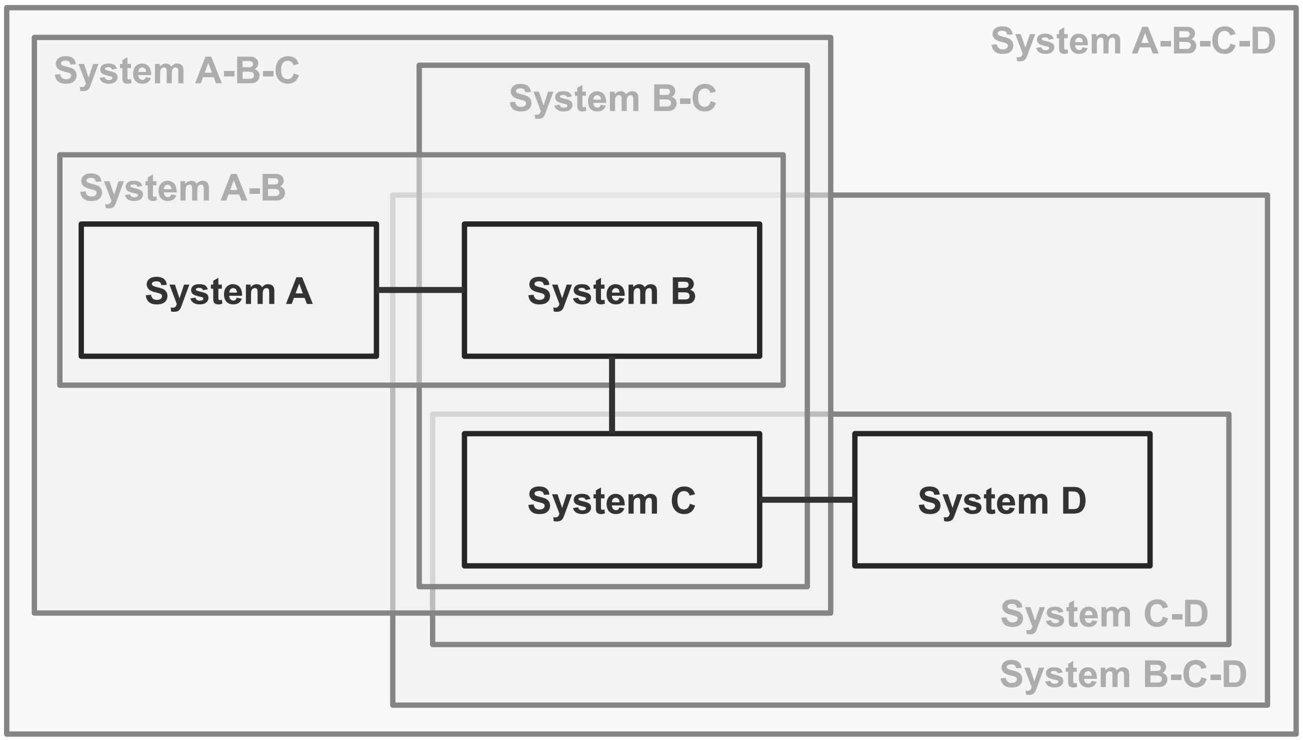


Dynamic System Architecture
(component collaboration relationships)

C_x capability x
X part x
— relationship

System Architecture
Holism versus particularism

Connecting systems creates new systems. Note: each innocent line in a *boxes and lines* diagram connects systems — i.e., binds them into a new whole and thus establishes a new system.



Universität
Basel

System Architecture
Architecture evolution

System Architecture does not stand still. **Systems evolve** due to their everchanging environment **and so does their architecture**.

Universität Basel

System Architecture
Architecture evolution

System Architecture does not stand still. **Systems evolve** due to their everchanging environment **and so does their architecture**.

System Architecture
Architecture as entirety of significant design decisions

"Architecture is the total of significant design decisions, where significant is measured by cost of change" ([Booch 2009])

Universität Basel

System Architecture
Architecture view models

System architecture organizes its insights using **architecture view models**. A view model combines isolated perspectives into a holistic architecture model.

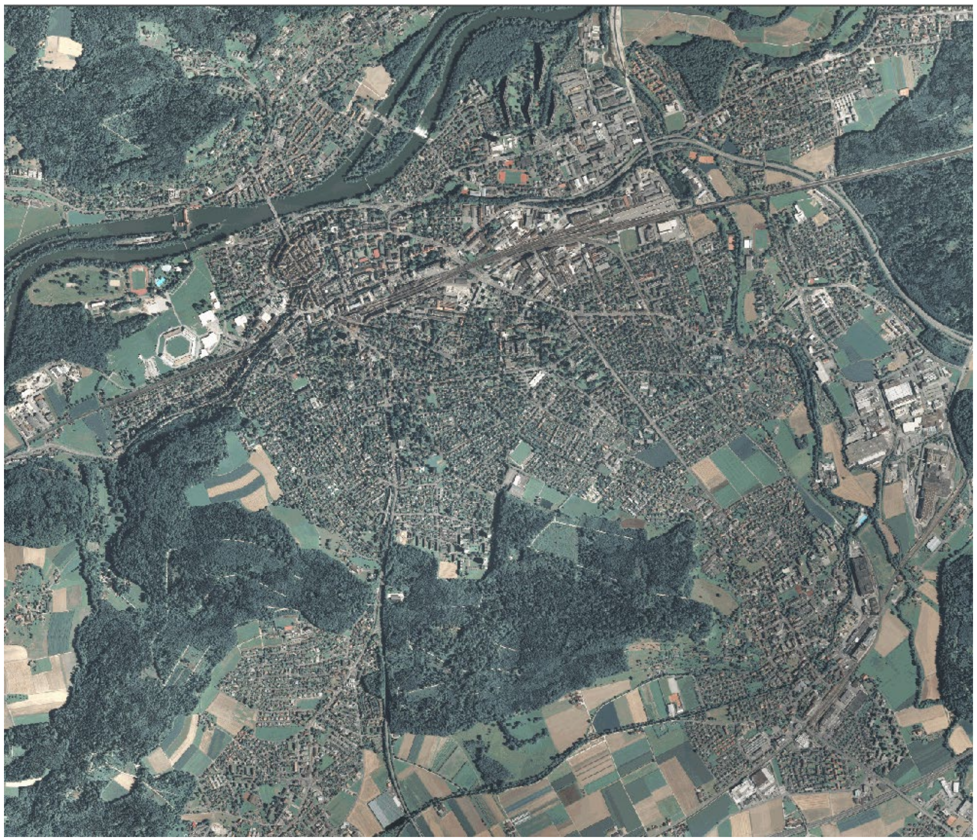


Photo (Reality)

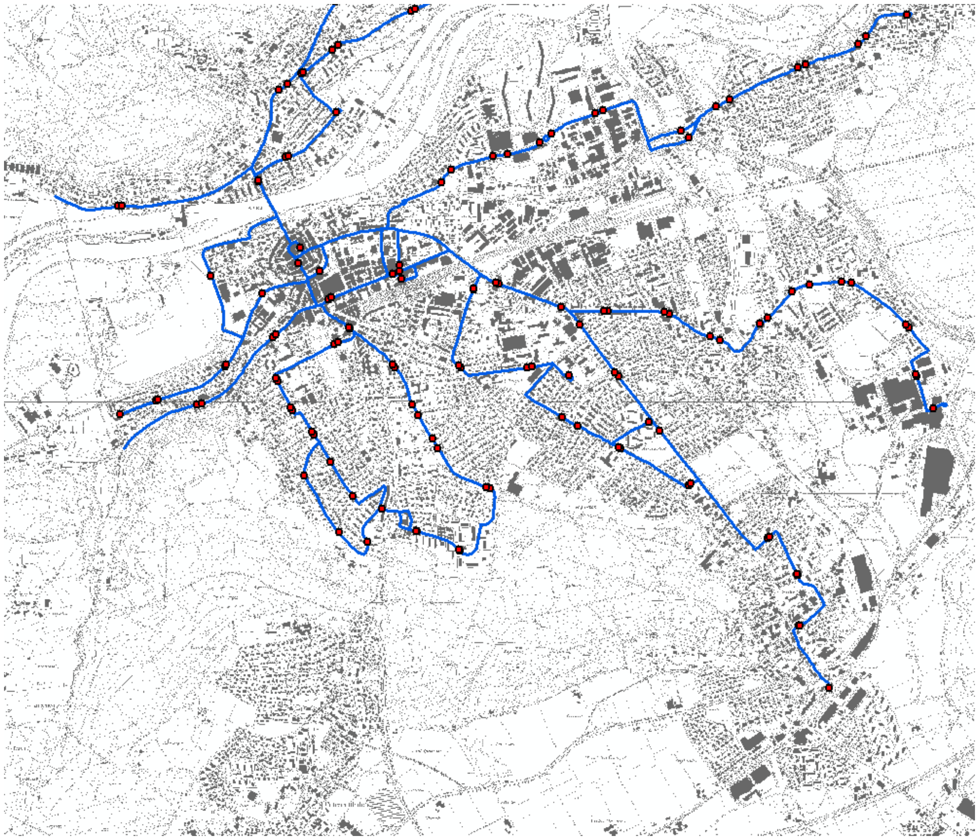
System Architecture
Architecture view models

System architecture organizes its insights using **architecture view models**. A view model combines isolated perspectives into a holistic architecture model.

Map (Model)

System Architecture
Architecture view models

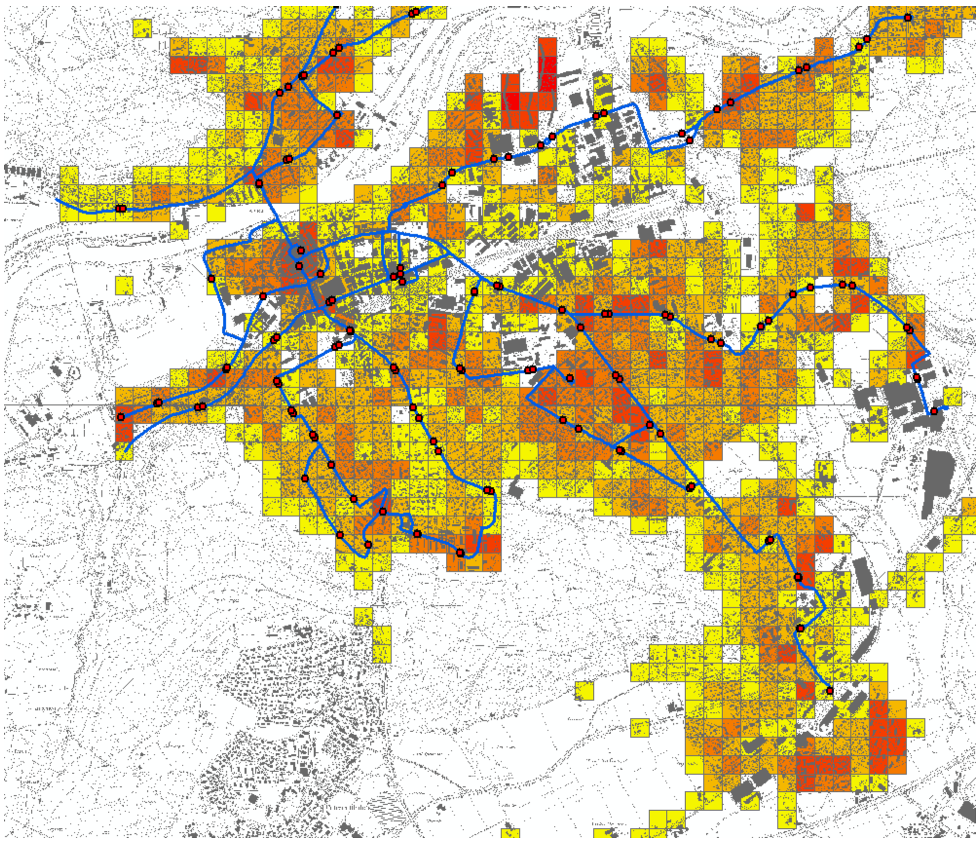
System architecture organizes its insights using **architecture view models**. A view model combines isolated perspectives into a holistic architecture model.



Public Transport

System Architecture
Architecture view models

System architecture organizes its insights using **architecture view models**. A view model combines isolated perspectives into a holistic architecture model.

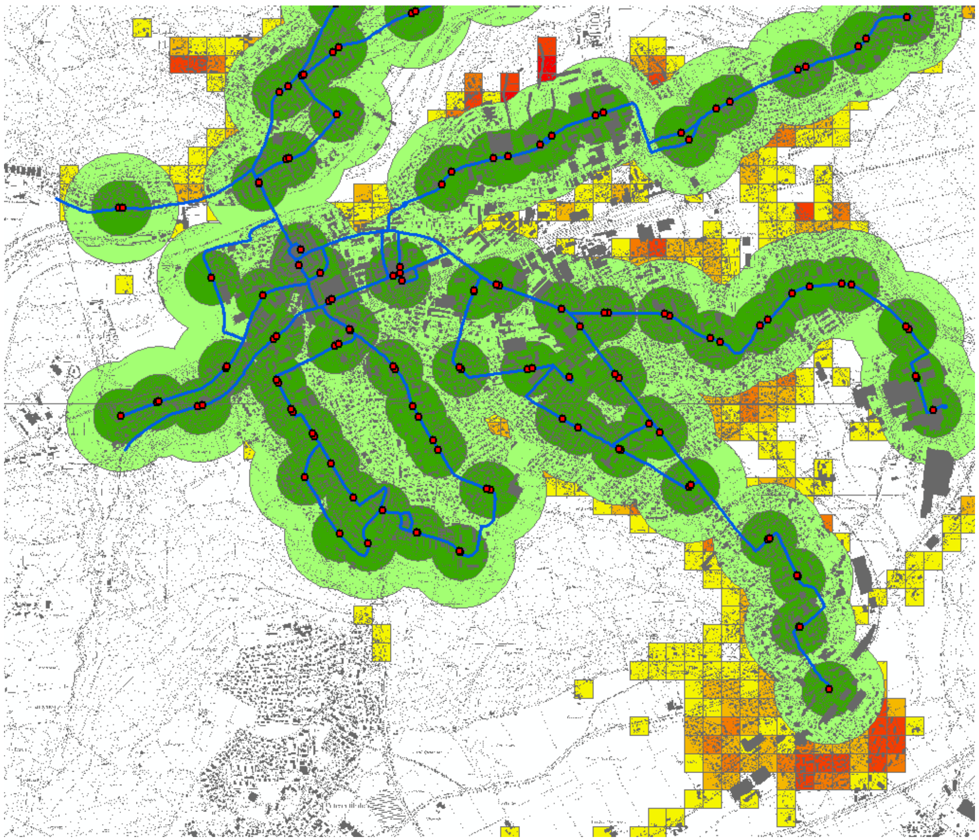


Population Density

Universität Basel

System Architecture
Architecture view models

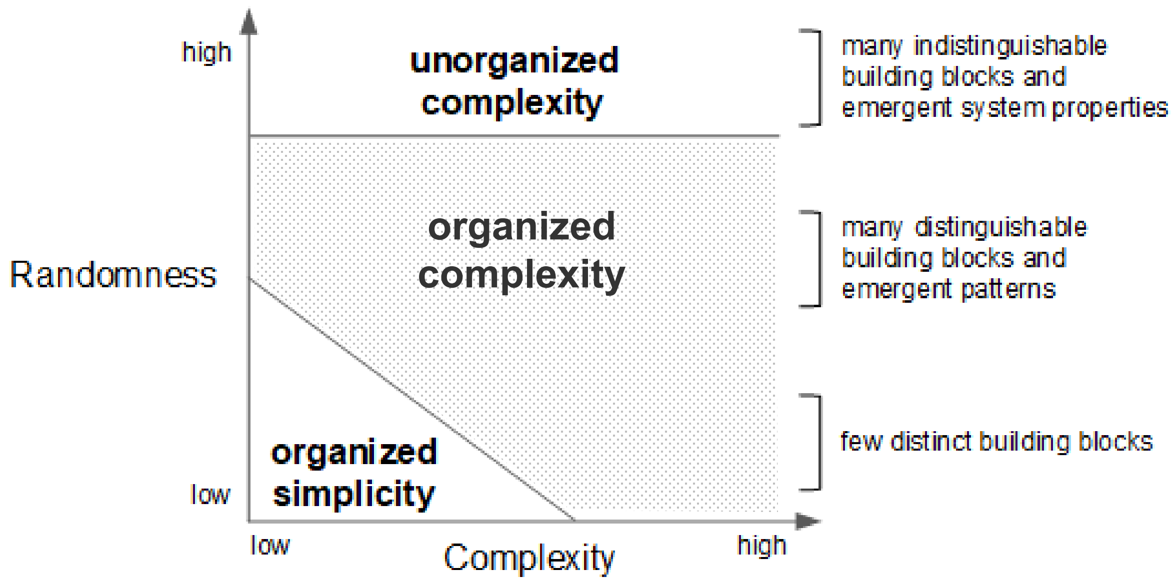
System architecture organizes its insights using **architecture view models**. A view model combines isolated perspectives into a holistic architecture model.



Public Transport Buffer Zones

System Architecture
Architecture relevance

The **relevance of architecture** is a function of problem complexity and randomness — architecture is relevant in dealing with **organized complexity**.



System Architecture
Architecture contribution in the enterprise

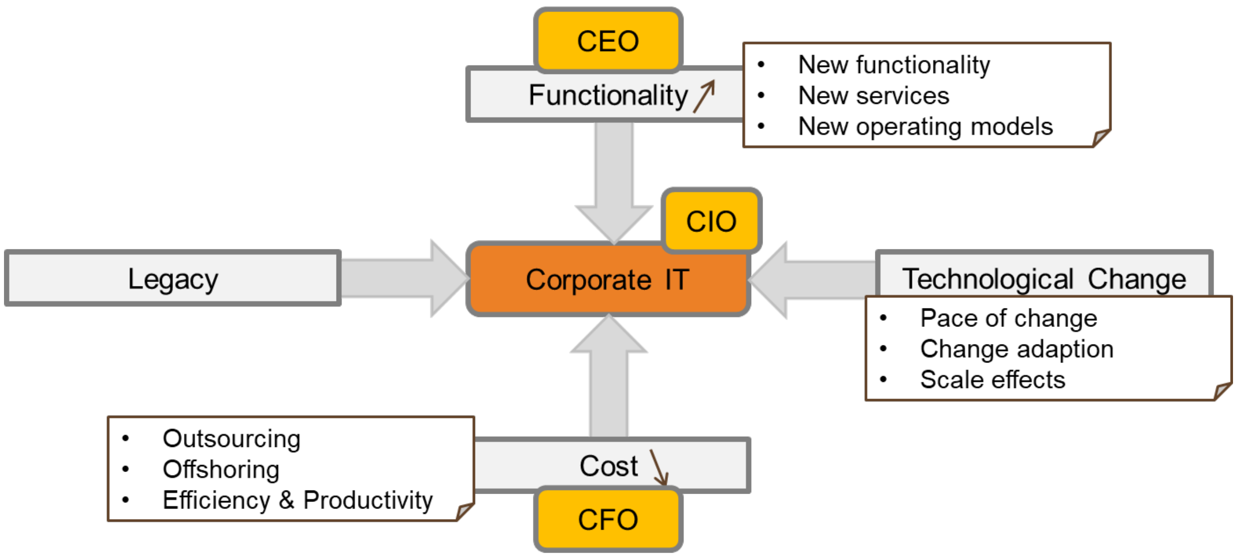
Systems (thus system architectures) are confronted with inherently complex problem spaces — they have to adequately **evolve in these environments**.

Business	Customers	>1.1 billion customers served around the world
	Challenges	Globalization, consumerizm, new technologies and applications, cost pressure, demographical dynamics, ..
	Touch Points	customers, whole-salers, interest groups, «bad guy», states and societies, ..
	Innovation	New ways to conduct business, new markets, business models, form factors, ...
	Business Projects	Projects which develop the business, joint ventures, mergers & acquisitions, ...
	Organisational Structure	Business & Product Units, Customer Channel Units, Matrix & federal organisations, projects
	Regions & Locations	Regional structure and distribution channels, regional legislation, geo-cultural specifics
IT	Associates & Externals	Internal versus external associates, skills & expertise, life-long learning, ..
	Programs & Projects	Projects and project organisations, inter-project dependencies, many moving bits and pieces, ..
	Applications	IT-based solutions realizing business capabilities, COTS,; SaaS, domain-specific apps, ...
	IT Platforms	IT-based solutions realizing application enabling platform, PaaS, ..
	Network & Messaging	IT-based solutions realizing and enabling IT Platforms, IaaS, ..
	Data Centers	Physical systems and facilities realizing and enabling fundament for all IT further up, DR, ..
	Sourcing & Partners	Multi-national, global outsourcing, off-shoring and services partners / sourcing models world-wide

System Architecture

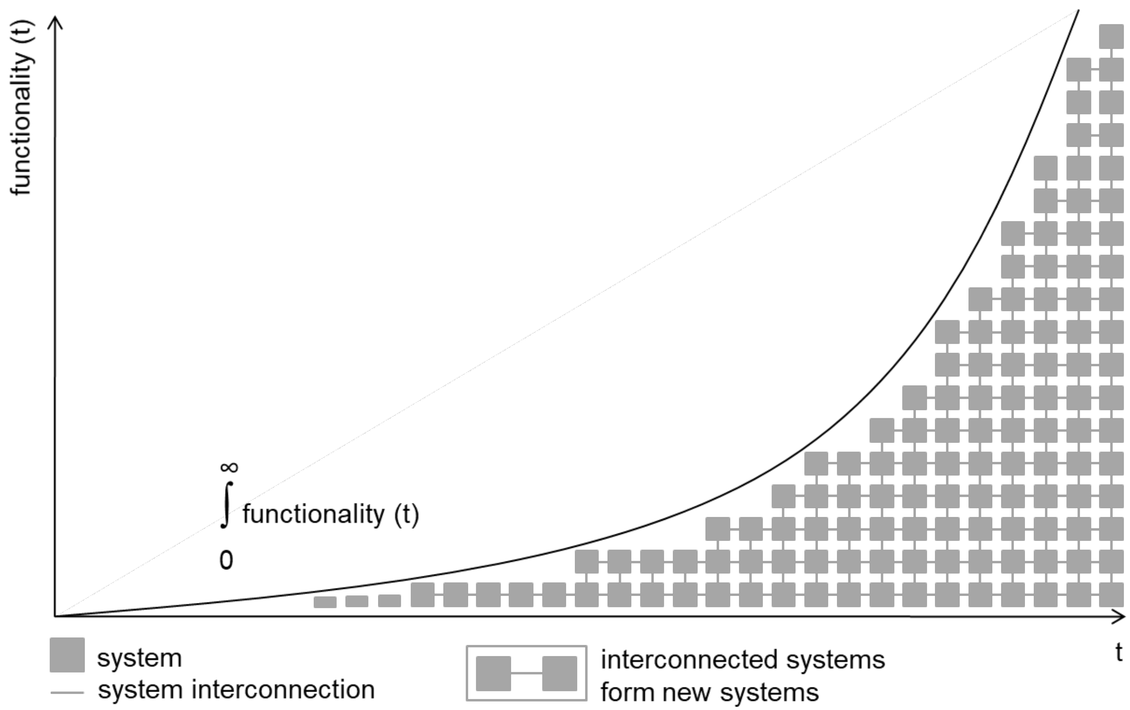
Architecture contribution in the enterprise

Constantly evolving business models and technical innovations require **adaptable systems** while at the same time cost pressure increases and legacy investments must be kept vital.



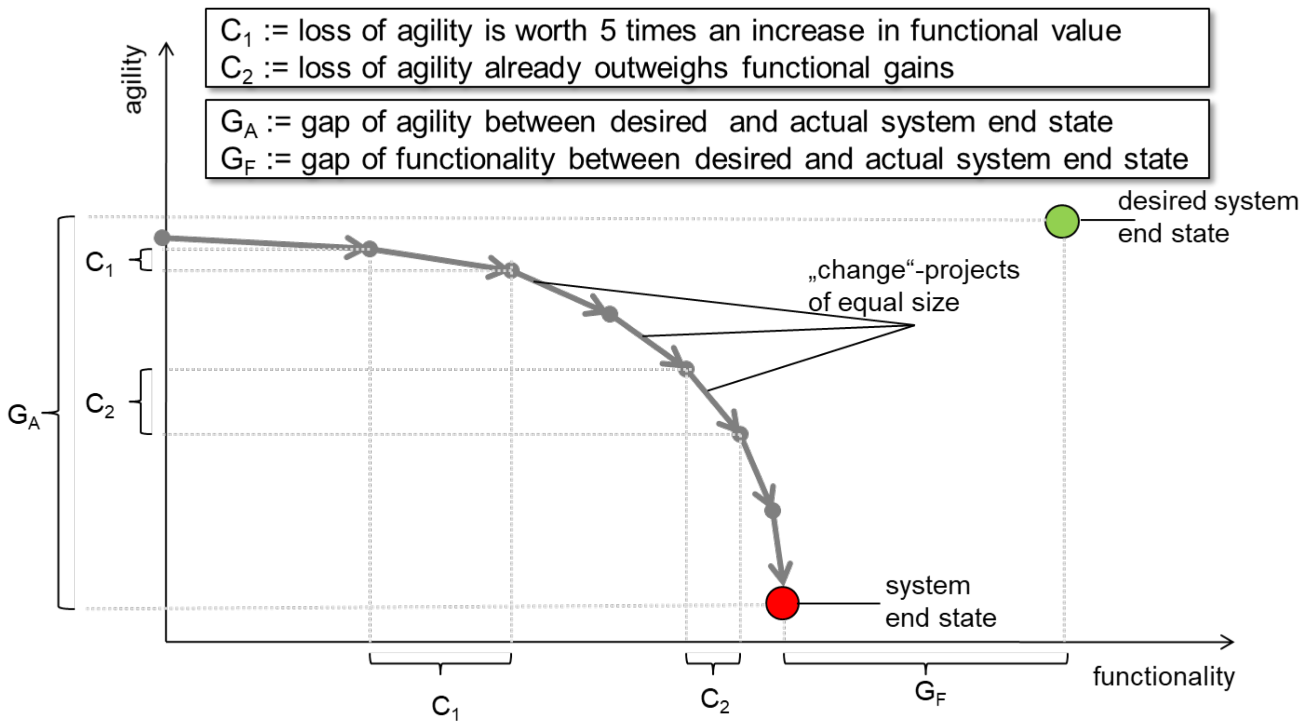
System Architecture
Architecture contribution in the enterprise

Legacy complexity grows exponentially, if systems are predominantly added and interconnected and at the same time never decommissioned.



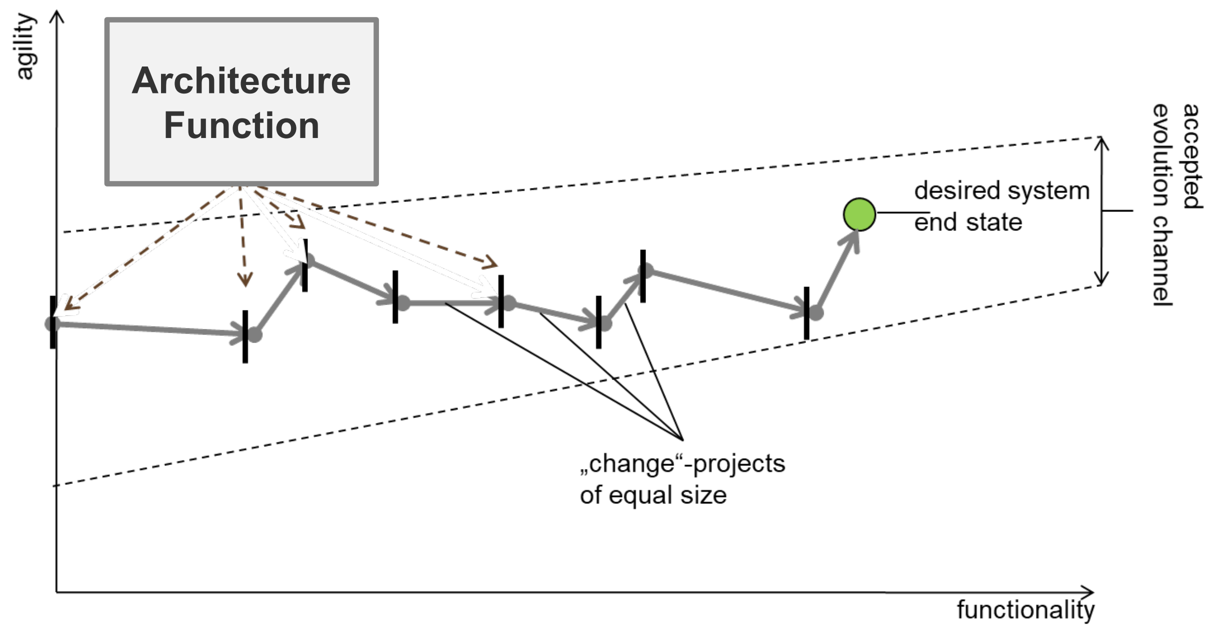
System Architecture
Architecture contribution in the enterprise

For large systems a short-term, one-sided focus on functionality (at the cost of agility) leads to a **complexity problem and crisis** in the medium to long term ([Murer et al 2010]).



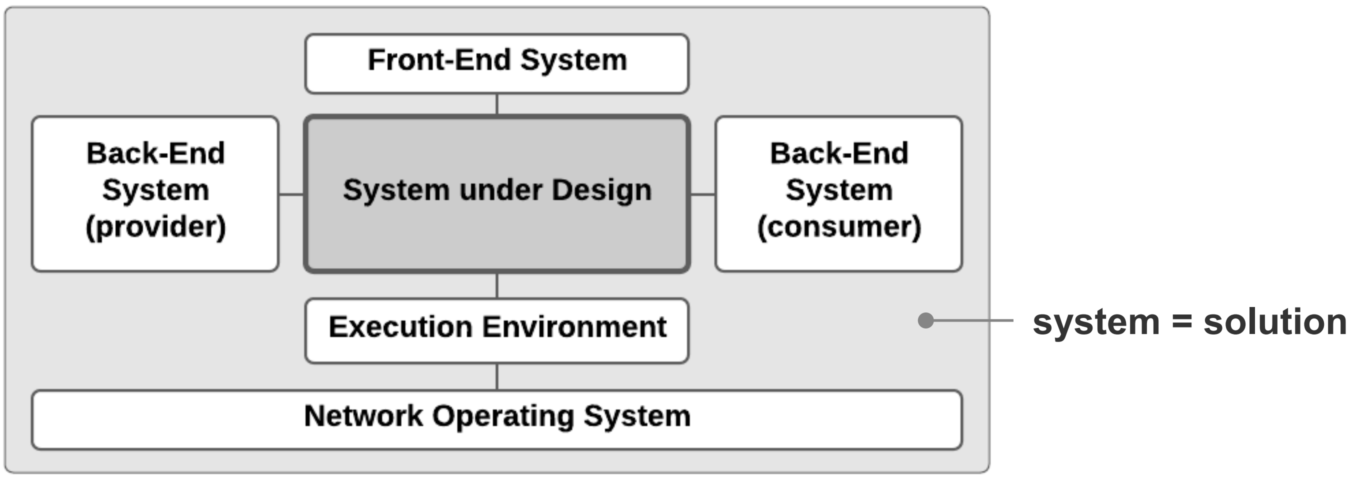
System Architecture
Architecture contribution in the enterprise

Architecture functions make important contributions to **controlling** (i.e., domain architecture (*do the right thing*)) and **implementing** (i.e., software architecture (*do the thing right*)) the **transformation of large systems**.



System Architecture
Architecture differentiation

If you develop a solution and consider the whole solution as a system, then only a part of this solution consists of its own, genuinely new contribution (i.e., system under design).



Beyond this part, practically every serious solution consists of further components (e.g., execution environment, network operating system, back-end systems (consumer and provider) and front-end systems).

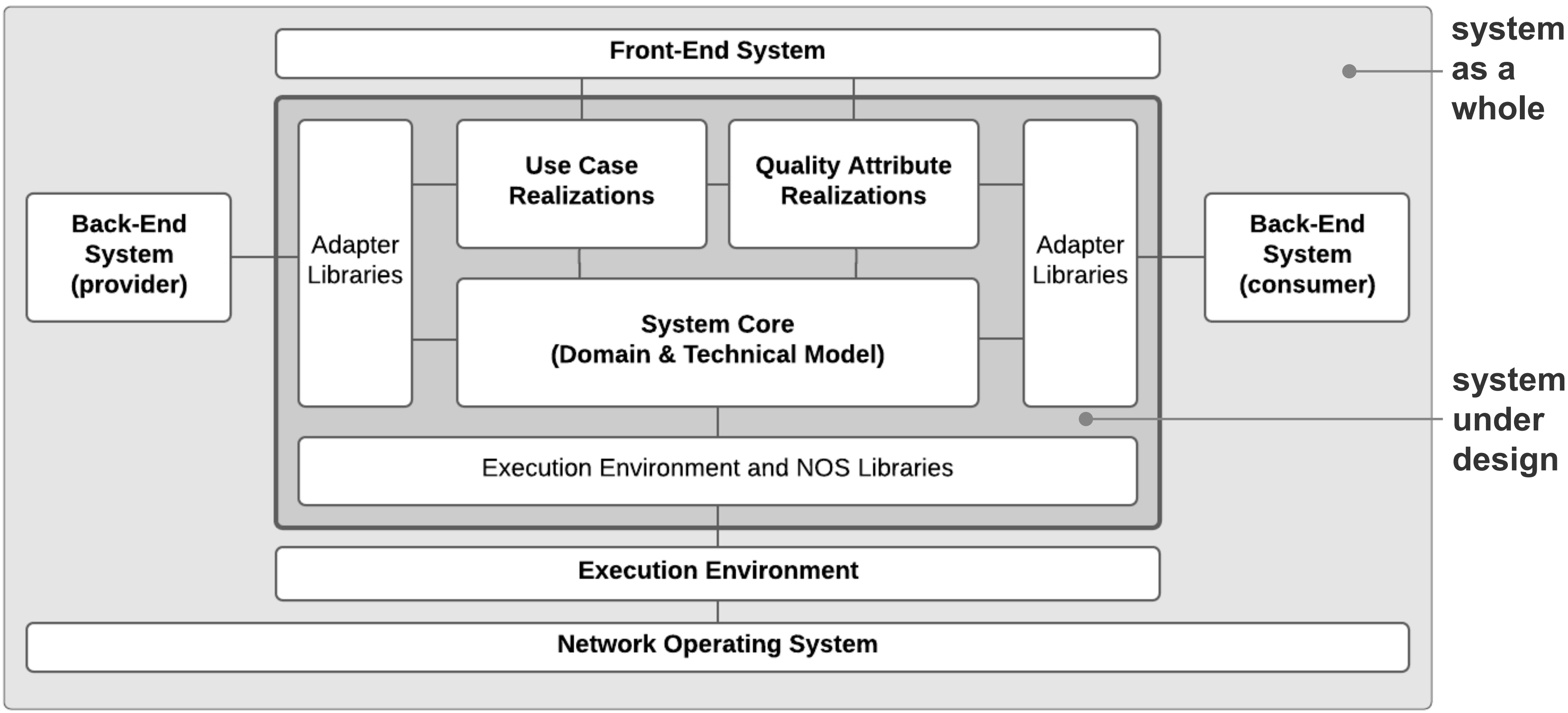
System Architecture
Architecture differentiation

The further components make a broad range of contributions to complement systems under design:

- **Front-End Systems** include client-side technology platforms like client devices, operating systems, or web browsers.
- **Back-End Systems** (consumers and providers) represent other systems which provide or consume data and services to/from the system under design. For example, via web services, ETL- or messaging technologies, integration broker platforms, or database APIs.
- **Execution Environments** provide run-time containers to the system under design. Examples are JEE web-application servers, CORBA platforms, or the .Net runtime environment.
- **Network Operating Systems (NOS)** provide fundamental services to all operated systems. For example, distributed connectivity, file, printing, naming, directory, or time services.

System Architecture
Architecture differentiation

Any **system under design** decomposes into further components itself – this is outlined, below.



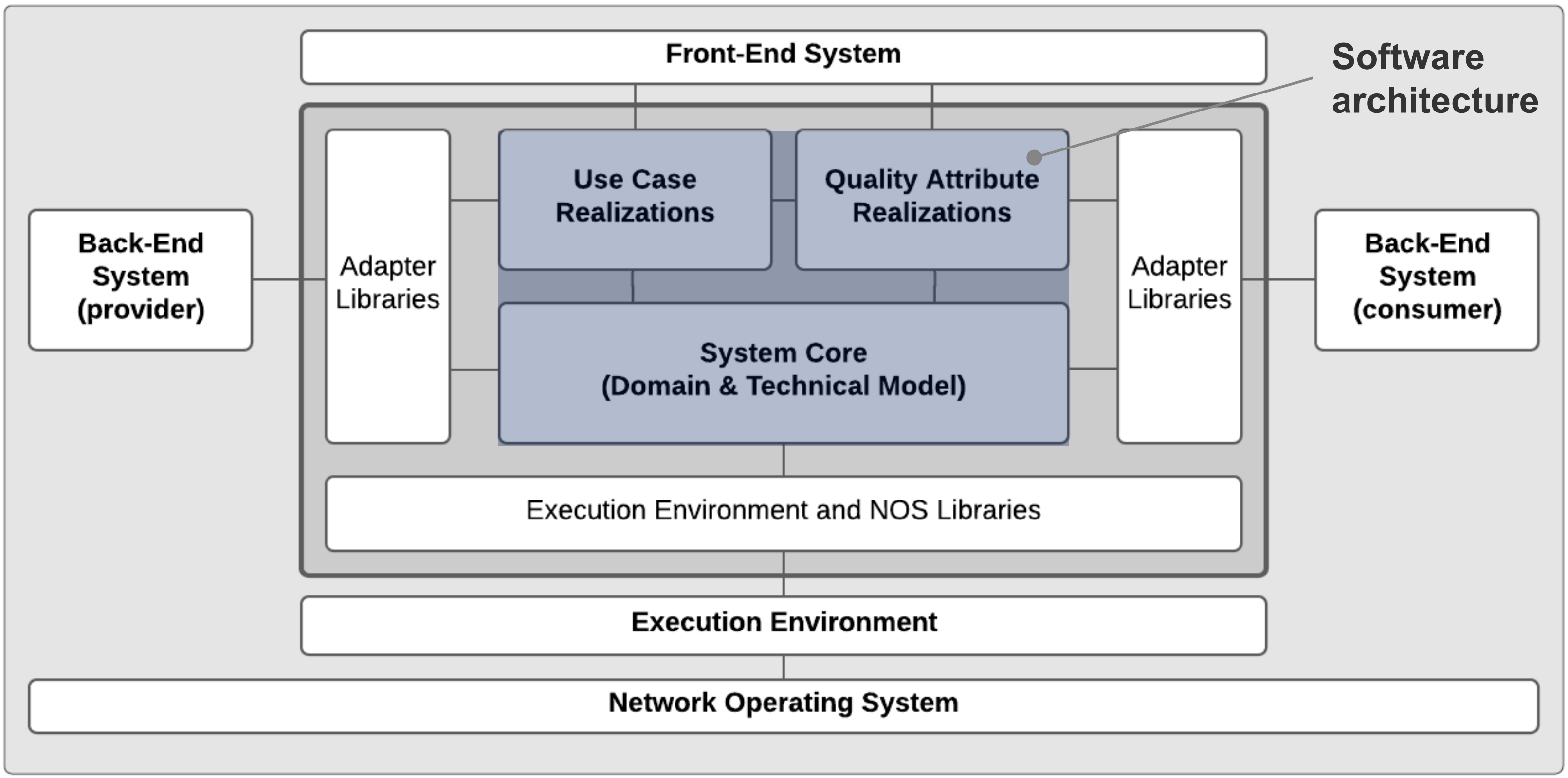
System Architecture
Architecture differentiation

The components of the system under design in more detail are ...

- **Use Case Realizations**. Components and component collaboration to realize required system functionality.
- **Quality Attribute Realizations**. Components and component collaboration to realize required system quality (e.g., security, performance, availability, modifiability).
- **System Core (Domain & Technical Model)**. Main business components (i.e., application-specific and poorly reusable) and components that help realize technical aspects (i.e., generic and well reusable).
- **Adapter Libraries**. Libraries that enable horizontal connectivity and information exchange between system under design and other systems (e.g., JDBC, JMS, RMI).
- **Execution Environment and NOS Libraries** (e.g., Servlet/JSP, EJB, JNDI).

System Architecture
Architecture differentiation

Any system under design usually decomposes into a set of generic components as the ones outlined, below.



Universität Basel

Lecture Agenda

- Classical Architecture
- Enterprise Operating Model
- Value Delivery Chain
- Architecture Disciplines
- System
- System Architecture
- Software Architecture

Software Architecture
Exercise

* 1-BSc_SWA_Definition

Universität Basel

Software Architecture
Exercise

*2-BSc_SWA_Modelling

Universität Basel

Software Architecture
Exercise

*3-BSc_SWA_BoxAndArrow

Bibliography
Lecture

[Arnold 2022]
Arnold, Ingo, *Enterprise Architecture Function — a pattern language for planning, designing, and executing*, Springer Science and Business Media, Berlin Heidelberg, 2022

[Booch 2009]
Booch, Grady, Object-Oriented Analysis and Design with Applications, Pearson Education, Amsterdam, 2009

[Campbell 2017]
Campbell, Andrew; Gutierrez, Mikel; Lancelott, Mark, *Operating Model Canvas*, Van Haren Publishing, 2017

[Convey 1968]
Conway, Melvin, How Do Committees invent? Datamation, https://www.melconway.com/Home/pdf/committees.pdf, 1968

Bibliography
Lecture

[Fowler 2003]
Fowler, Martin; *Who Needs an Architect*, IEEE Software,
http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

[Gartner Glossary 2020]
Gartner, *Gartner Glossary*, https://www.gartner.com/en/information-technology/glossary, 2020**[IEEE 2000]**
IEEE Computer Society, *IEEE Recommended Practice for Architecture Description of Software-Intensive Systems*, IEEE std. IEEE — pp. 1472-2000, New York, 2000

[Murer et al 2010]
Murer, Stephan; Bonati, Bruno; Furrer, Frank, Managed Evolution – A Strategy for Very Large Information Systems, Springer Science & Business Media, Berlin Heidelberg, 2010

[Porter 2004]
Porter, Michael, *Competitive Advantage — Creating and Sustaining Superior Performance*, Simon and Schuster Free Press, New York, 2004

Universität Basel

Bibliography
Lecture

[Vogel, Arnold et al 2011]
Vogel, Oliver; Arnold, Ingo; Chugtai, Arif; Kehrer, Timo, *Software Architecture: A Comprehensive Framework and Guide for Practitioners*, Springer Science and Business Media, Berlin Heidelberg, 2011

Questions