Fachhochschule Nordwestschweiz
Hochschule für Technik

Software Architecture

Basic Terms and Concepts

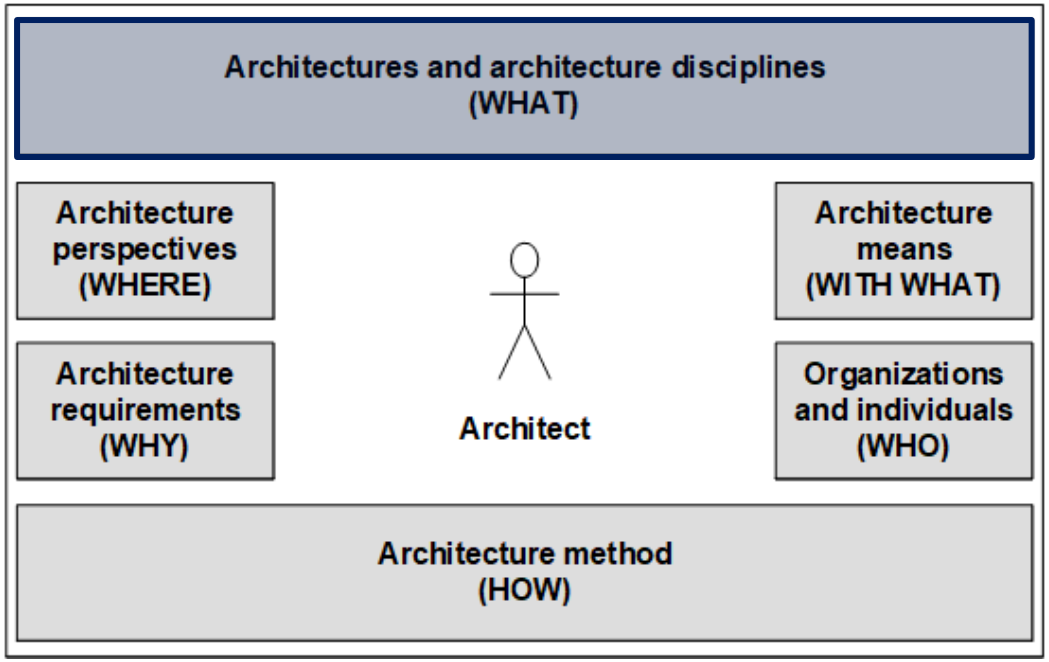
BSc

Ingo Arnold

Copyright Notice

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

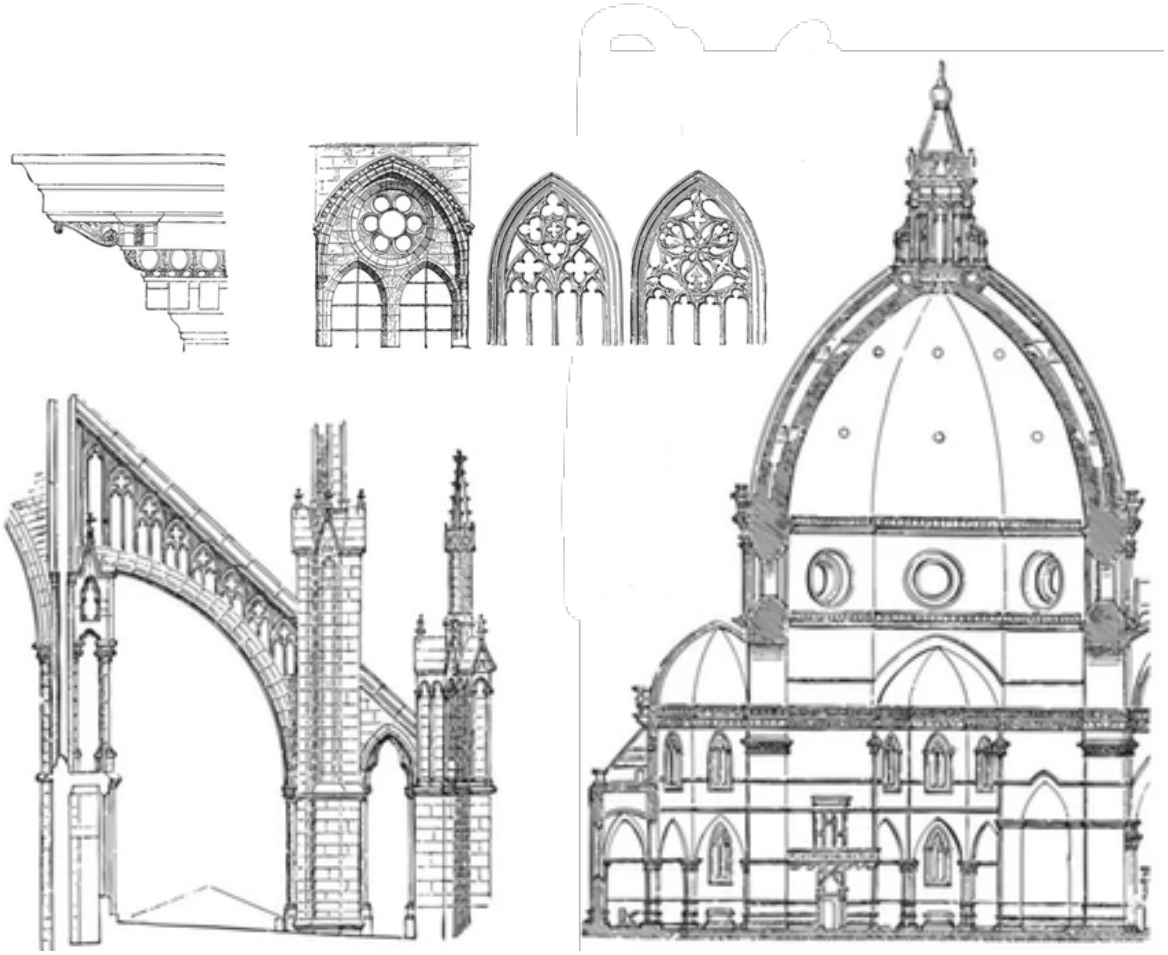
Lecture Agenda

- Classical Architecture
- Enterprise Business and Operating Model
- System
- Architecture Disciplines
- System Architecture
- Software Architecture

Classical Architecture
Overview

Classical Architecture is an Art and a Science

- Structures, like buildings, created by classical architects must address both **functional** (e.g., utilisation scenarios) and **non-functional** (e.g., conversion options (aka: changeability)) needs.
- An essential **requirement category** of classical architecture is **aesthetics** – that is, how buildings look, what style they adapt and how their appearance affect people.
- In *Software Architecture: A Comprehensive Framework and Guide for Practitioners* ([Vogel, Arnold et al 2011]), we argue that **classical architecture** is both – an **art and a science**.



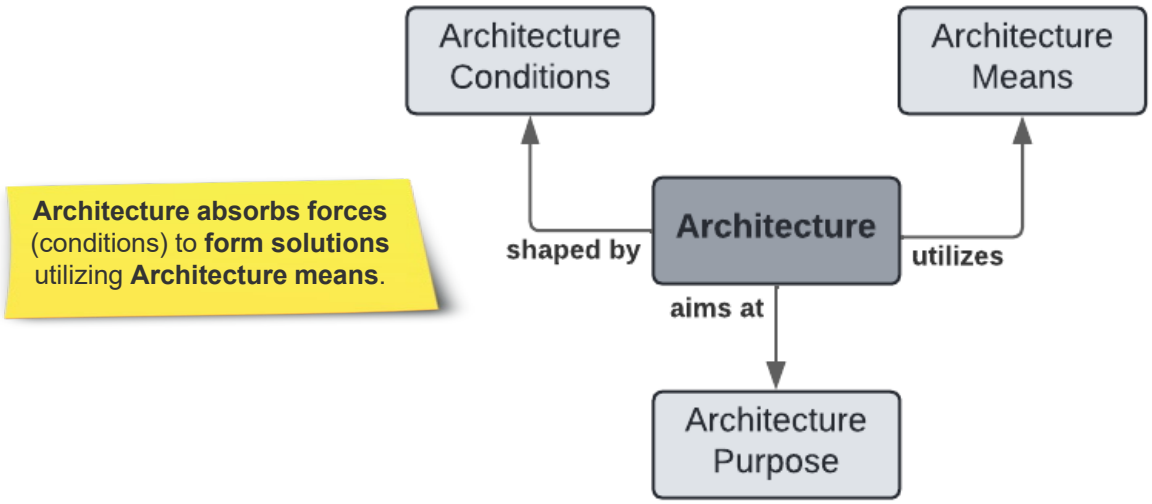
[1] [Vogel, Arnold et al 2011]

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).



Classical Architecture

Process and process result

Another aspect we want to recognise is that the term **Architecture** encompasses both: the **Activity of the Architect** as well as the **result** that **this Activity produces**.

Architecture refers to the **architectural thinking** and **activity** of the **Architect** ...

Architecture as a Discipline

... ... and the **result** of this **architectural thinking** and **acting process**.

Architecture as an Artefact

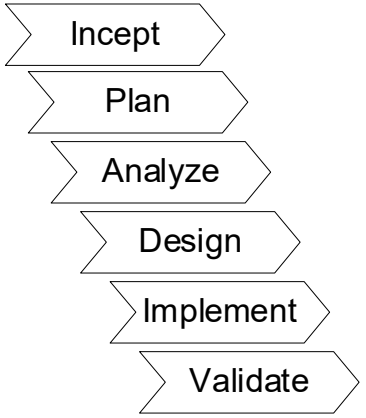
Classical Architecture

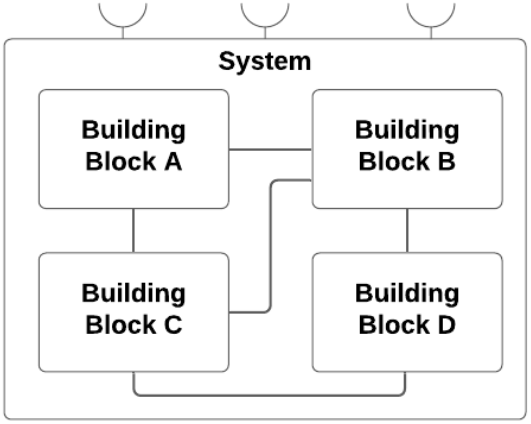
Process and process result

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





Lecture Agenda

- Classical Architecture
- Enterprise Business and Operating Model
- System
- Architecture Disciplines
- System Architecture
- Software Architecture

Enterprise Business and Operating Model
Overview

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**.

A **business model** describes the **rationale** of **how** an **organization creates, delivers**, and **captures value**, in **economic**, **social**, **cultural**, or other **contexts** [Osterwalder 2010].

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.

Enterprise Business and Operating Model
Business Model

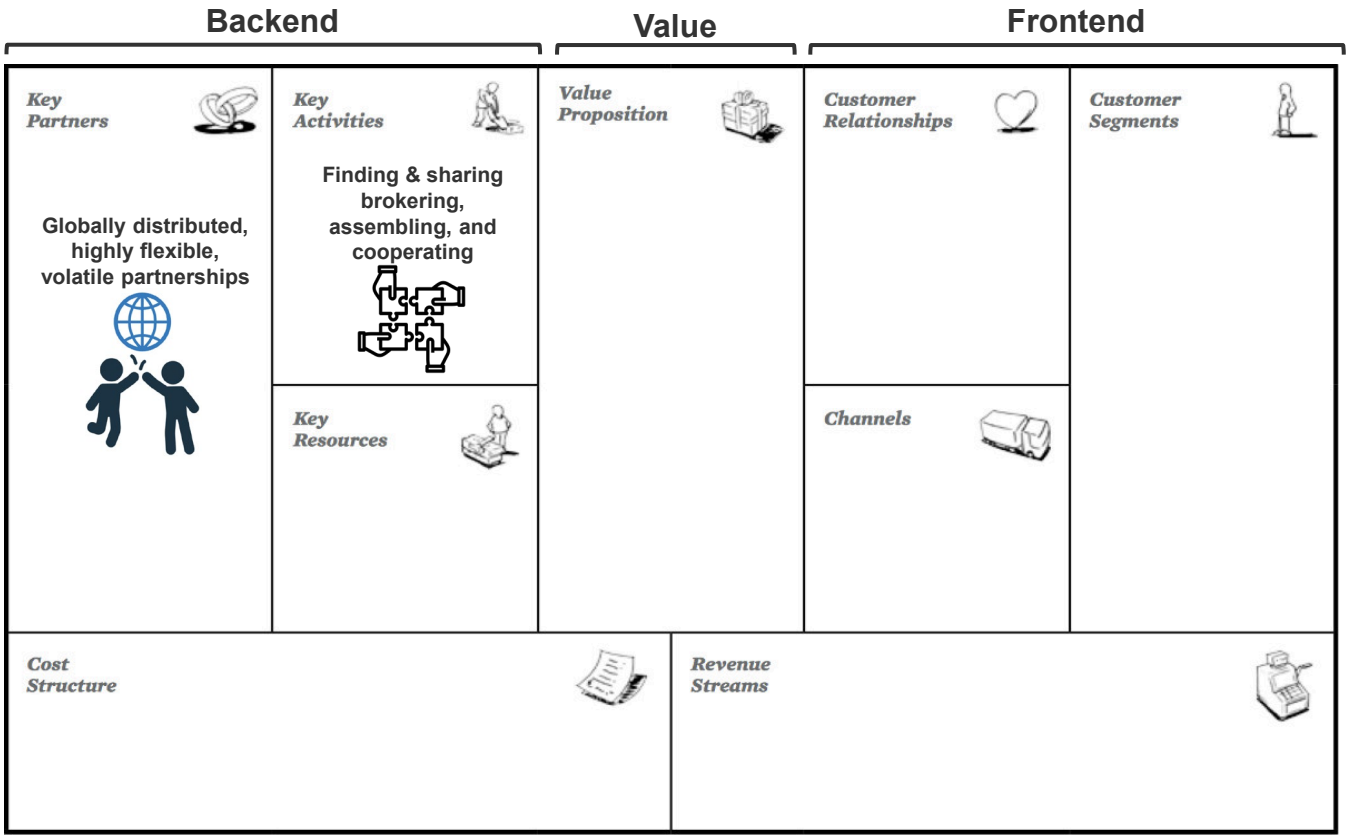
A **business model** describes the **rationale** of **how** an **organization creates, delivers**, and **captures value**, in **economic**, **social**, **cultural**, or other **contexts** [Osterwalder 2010].

Alexander Osterwalder et al. propose the **Business Model Canvas** as a **general template** for **discussing**, **elaborating**, and **documenting Business Models**.

Backend is where **value** is **created**, while the **Frontend demands** and **consumes** the created **value**

Partners are the **external organizations** that **provide** critical **inputs** to the **business** (e.g., raw materials, or IT services).

Activities are the **steps** the **business performs** to **deliver** its **value proposition**.



Enterprise Business and Operating Model
Business Model

A **business model** describes the **rationale** of **how** an **organization creates, delivers**, and **captures value**, in **economic**, **social**, **cultural**, or other **contexts** [Osterwalder 2010].

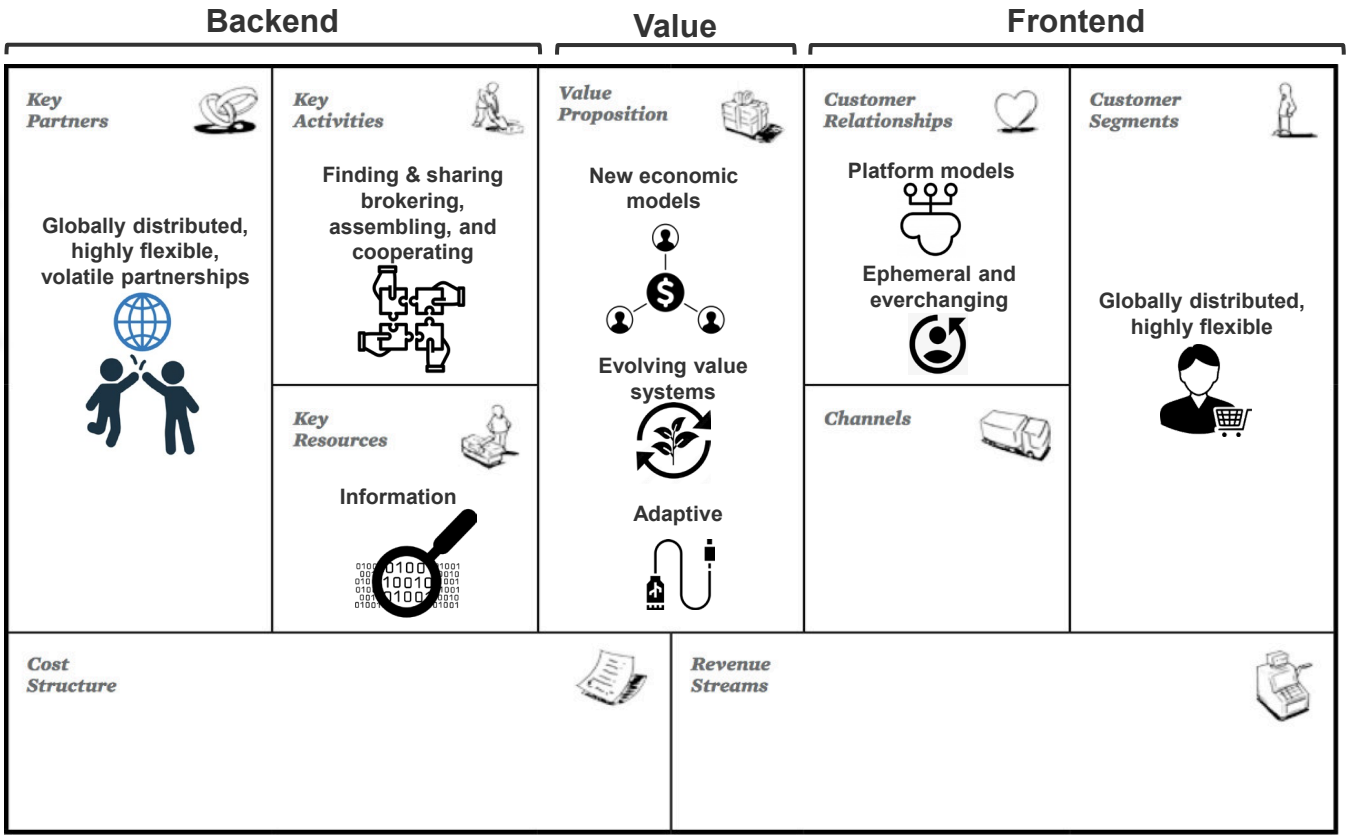
Activities are **performed** by or based on the **Enterprise's Resources.**

Resources are the **assets** of the enterprise, such as **employees**, **technology**, or **buildings**.

Value Proposition refers to what the Enterprise **offers** its **customers**, usually in the form of **services** or **products**.

Customer Segments distinguish consumer groups.

Customer Relationships describe **how customers** are **acquired** and **retained**.



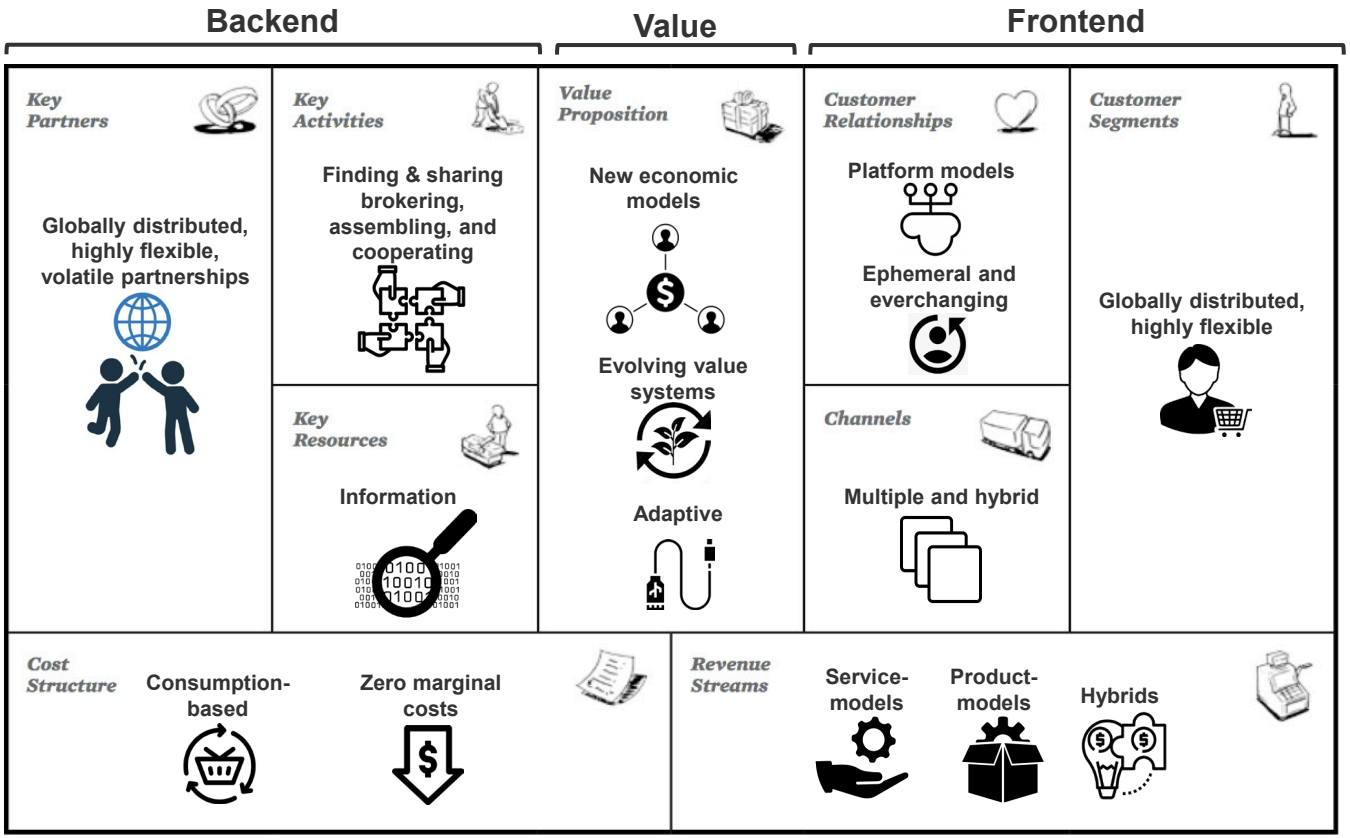
Enterprise Business and Operating Model
Business Model

A **business model** describes the **rationale** of **how** an **organization creates, delivers**, and **captures value**, in **economic**, **social**, **cultural**, or other **contexts** [Osterwalder 2010].

Channels are the means **by which** the **enterprise communicates** with its **customers**.

Cost Structure captures the **financial view** of the **resources used**.

Revenue Streams represent how the **enterprise makes money**.



Enterprise Business and Operating Model
Operating Model

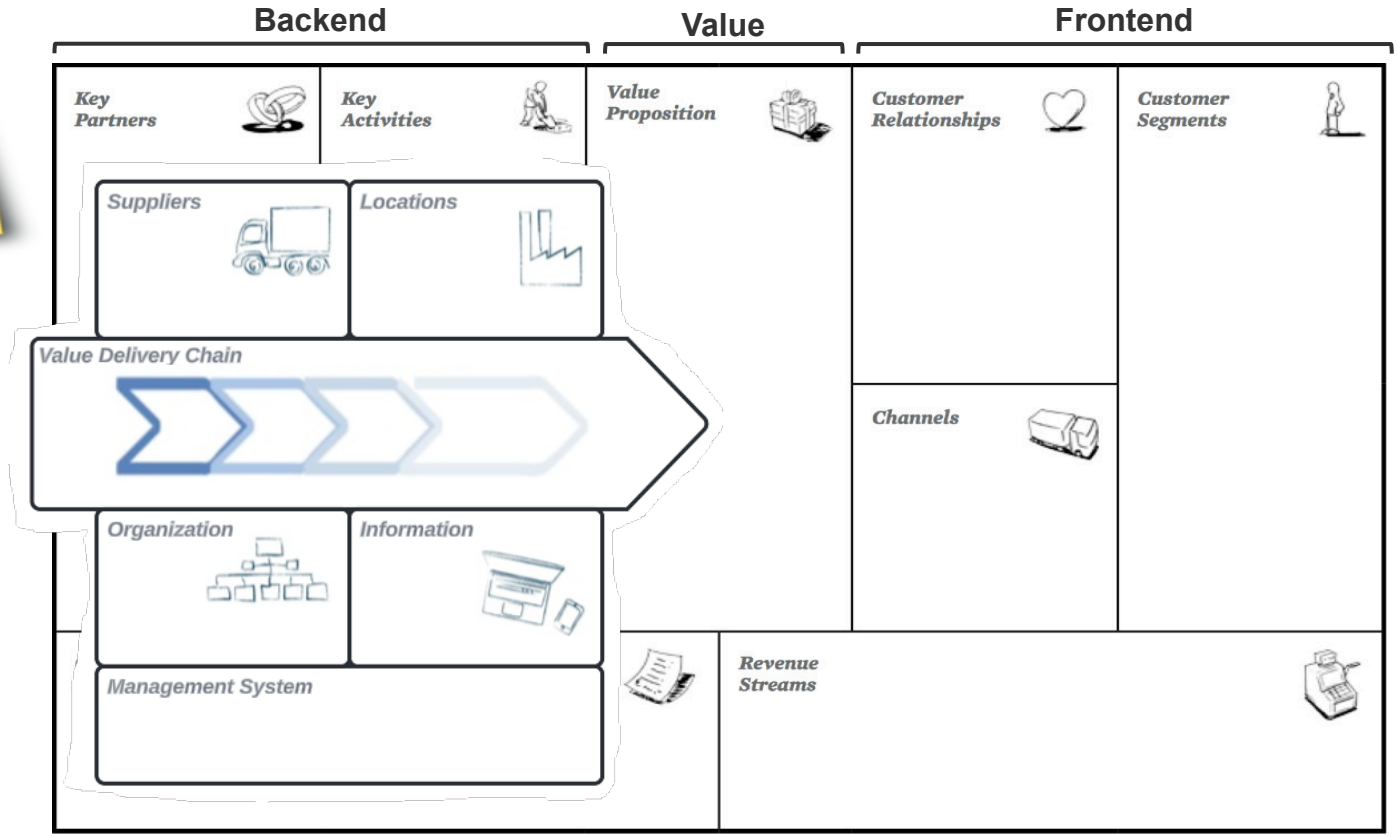
An **Operating Model specifies** and **captures** how the **inner workings** of an Enterprise are designed to deliver value to its customers.

Similar to the **Business Model Canvas**, there is also an **Operating Model Canvas** ([Campbell et al 2017]), which **hooks into** the **Business Model Canvas**.

Suppliers are synonymous with **Partners** in the **Business Model Canvas**.

Locations, **Organization**, and **Information** are synonymous with **Resources**.

Locations determine **where work** is **performed** and **what Resources** are **needed** at each location.



Enterprise Business and Operating Model

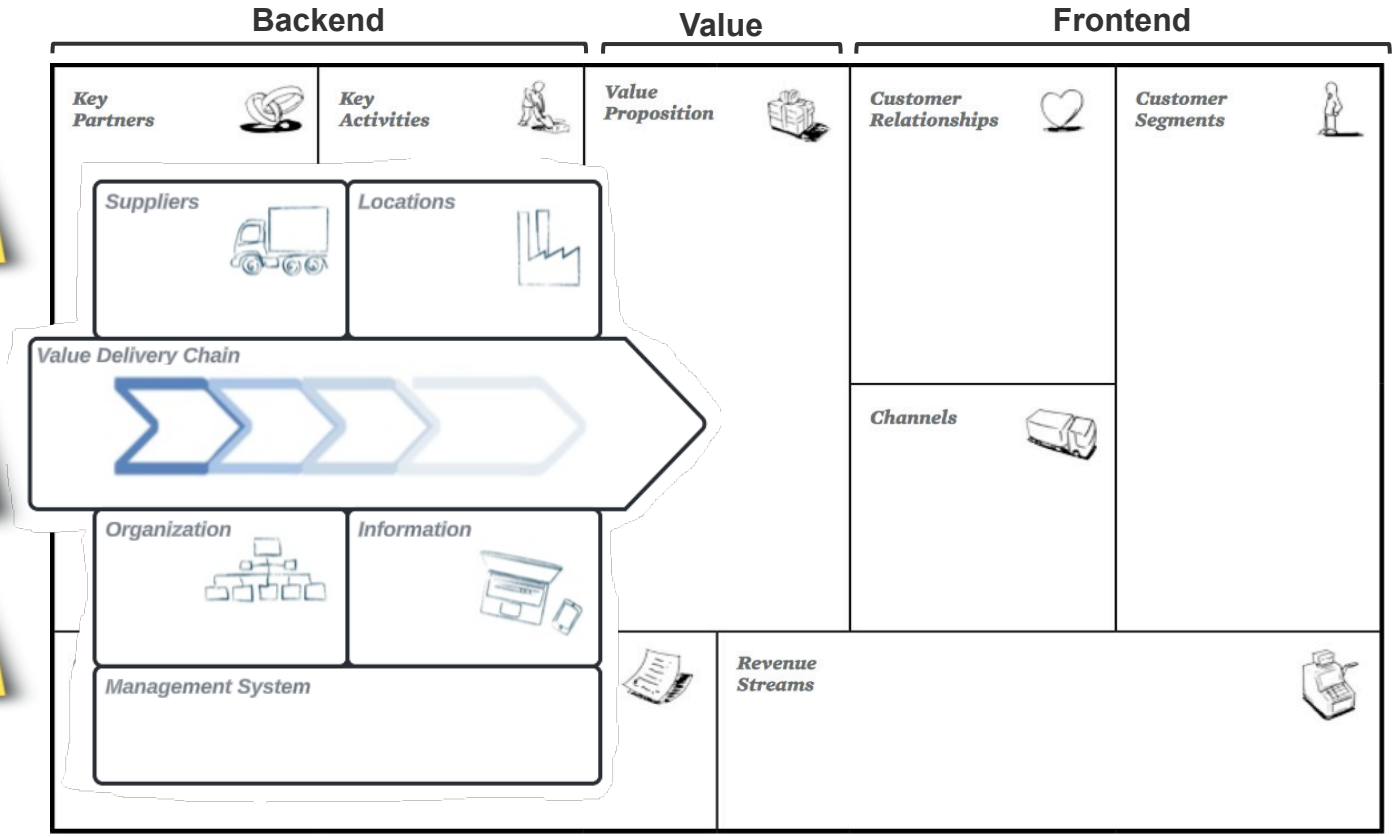
Operating Model

An **Operating Model specifies** and **captures** how the **inner workings** of an Enterprise are designed to deliver value to its customers.

Organization describes the **organizational structure** and **who performs** which **activities** in the **value chain**.

Information represents the **data** and the **technical means** to **create**, **modify**, **find**, **retrieve**, or **archive** the **data underlying** the **activities** of the **Operating Model**.

Management System refers to all the **means used** by **management** to **plan**, **build**, and **operate** the **business**.



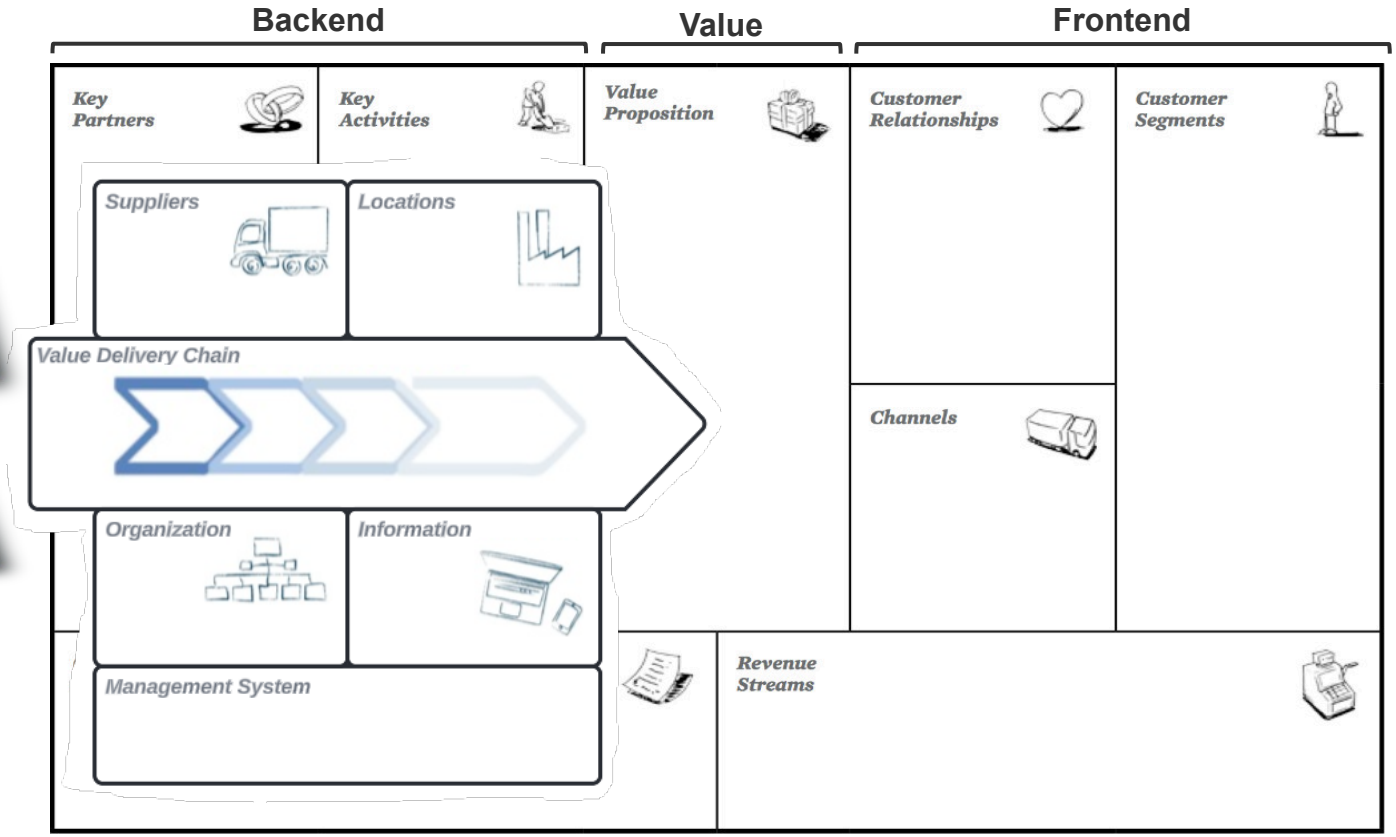
Enterprise Business and Operating Model

Operating Model

An **Operating Model specifies** and **captures** how the **inner workings** of an Enterprise are designed to deliver value to its customers.

The **Value Delivery Chain** describes the **steps** required **to deliver the value proposition** and **corresponds** to the **activities** in the **Business Model Canvas**.

Activities are **captured in the form of processes** that **explain** precisely **how work is performed** and **by whom**.

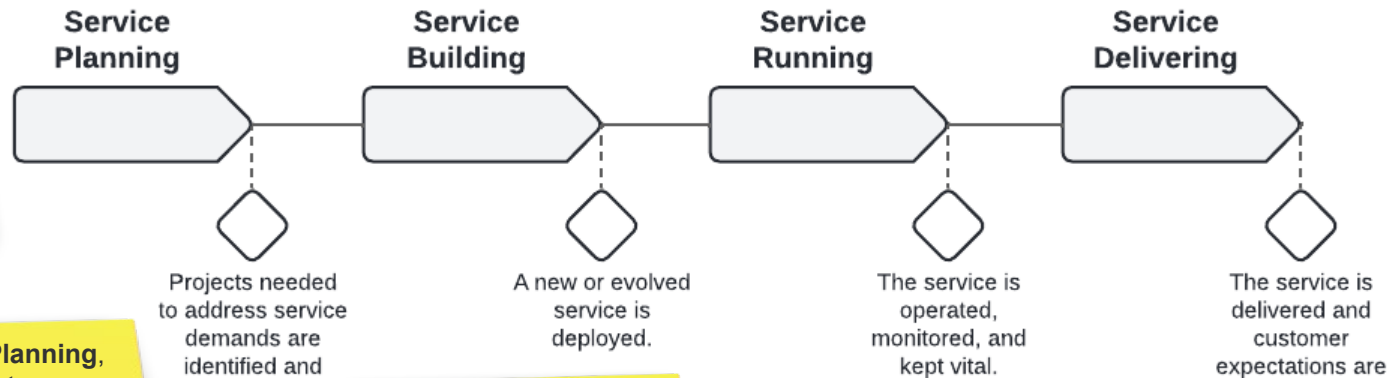


Enterprise Business and Operating Model

Enterprise Operating Model

The **Operating Model** presented here is **generic** and **represents** a **scheme** that shows a **value chain's main value streams**, **value stages** and the **contributions** appropriate **Enterprise Functions make** to the **value chain**.

From a **high-level perspective** the **generic Enterprise Value Chain comprises** the value streams **service planning, service building**, **service running**, and **service delivering**.



At the **end** of **Service Planning**, we know what **projects** are **needed** to **develop** the **service portfolio** that **meets** the **enterprise's service requirements**.

At the **end** of **Service Building**, new **services** are **built**, **tested**, and **deployed**.

During **Service Running services** are **monitored** and **kept vital** while **servicing** their **customers**.

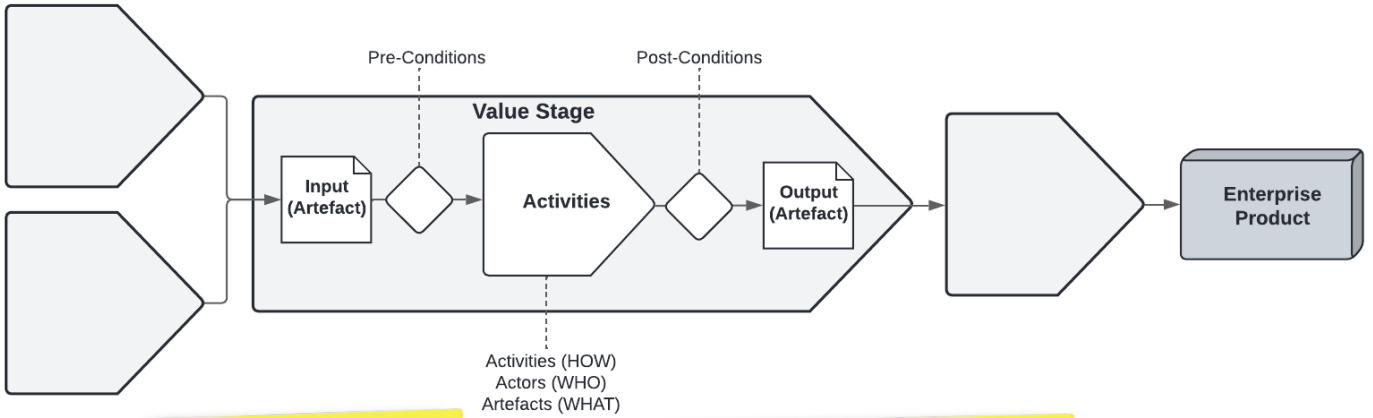
At the end of **Service Delivering**, **customer expectations** are **met**,

Enterprise Business and Operating Model
Value Delivery Chain

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 Delivery Chain** views an **Enterprise** as a **system** that is **decomposed into subsystems**, with **each subsystem** having its **inputs**, **transformation processes**, and **outputs**.

A **Value Delivery Chain** views an **Enterprise** as a **set of processes** at different levels of process aggregation

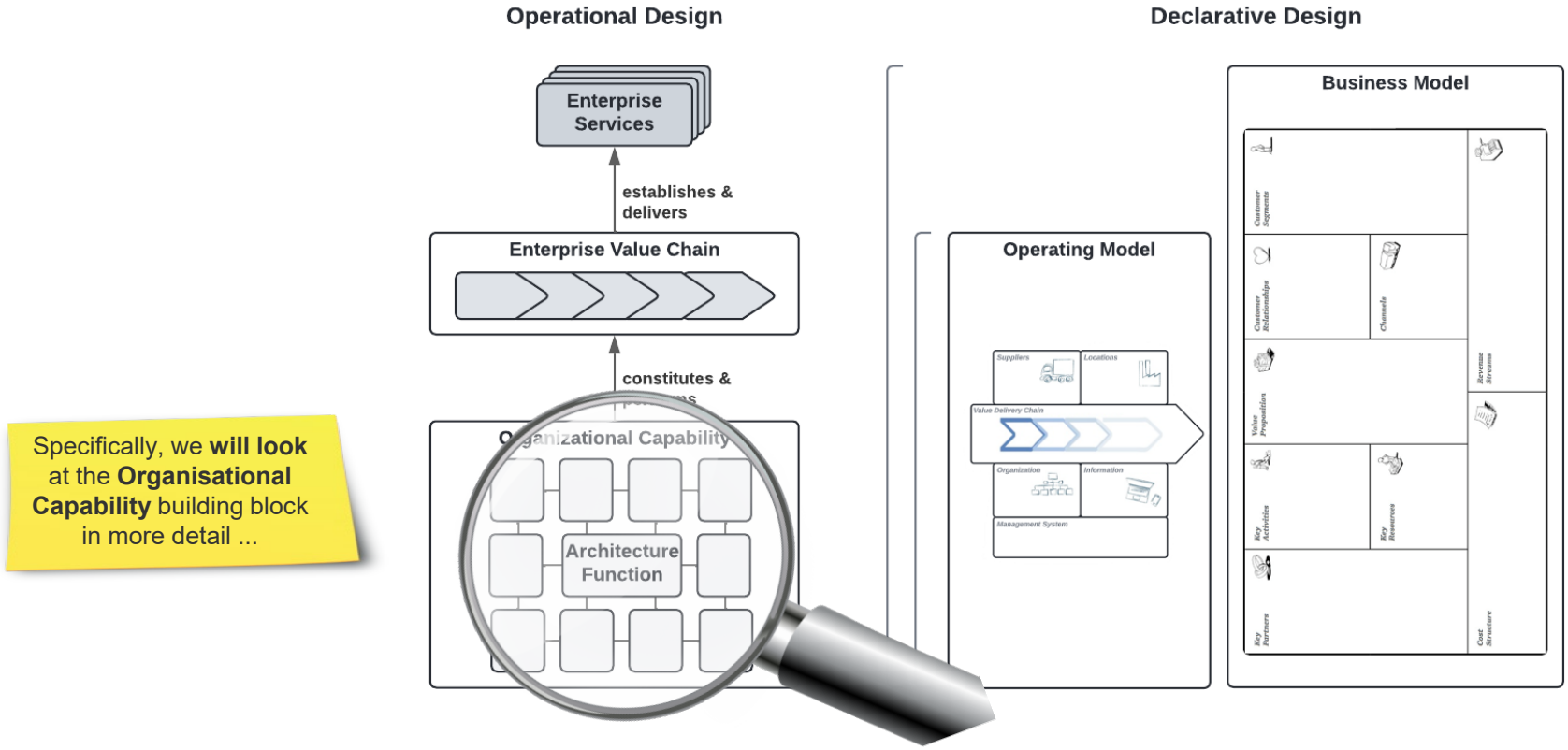


At the **next level of detail**, **activities**, **actors**, **artifacts**, and further means are **grouped** into **value stages**.

Value Stages are **arranged** such **that** the desired **overall outcome** is **achieved incrementally** by **passing its increments sequentially through all stages**.

Enterprise Business and Operating Model
Architecture Function

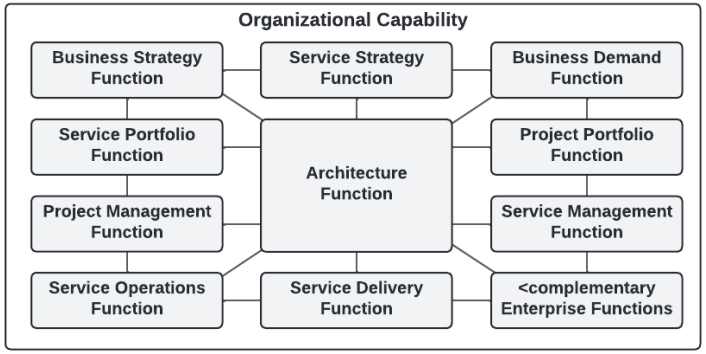
Now, we look at 'How' an **Architecture Function** (AF) **contributes** to **operationally realizing** an **Enterprise Operating Model** to **enable** its respective **Business Model**.



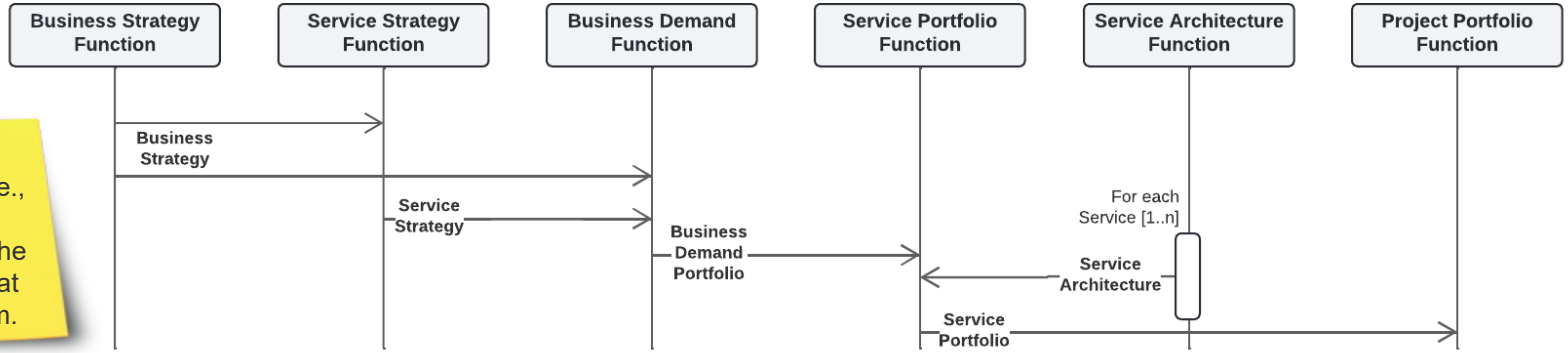
Enterprise Business and Operating Model
Architecture Function

Now, we look at 'How' an **Architecture Function** (AF) **contributes** to **operationally realizing** an **Enterprise Operating Model** to **enable** its respective **Business Model**.

An **operating model complements** the **static view** of **organisations** and the **general relationships** among them …

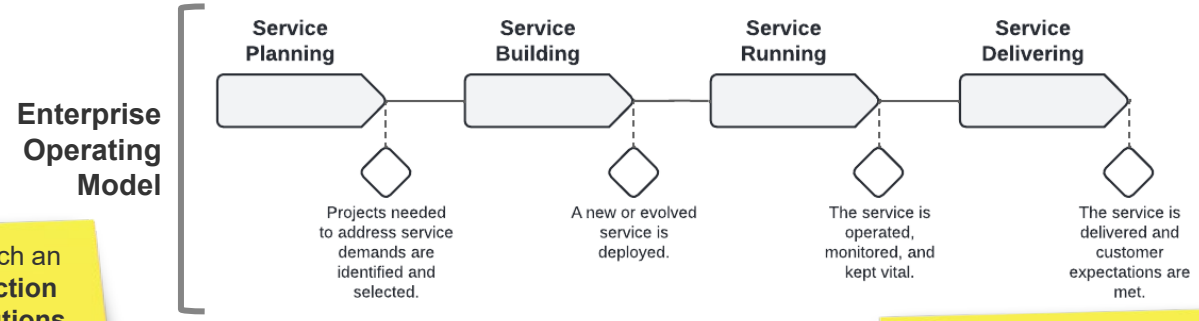


… by **describing** the **cooperative relationships** (i.e., **activities** and **processes**) **between organisations** and the **information** and **artefacts** that are **exchanged** between them.

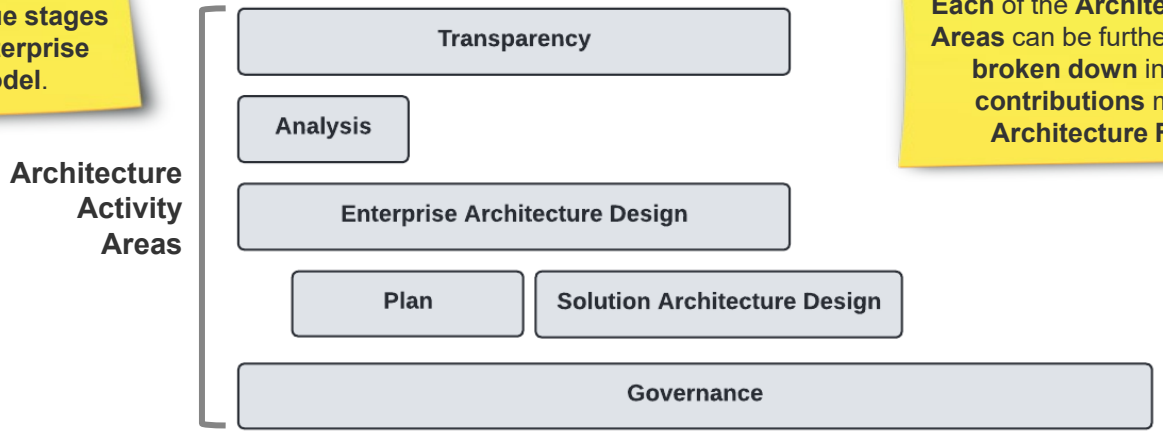


Enterprise Business and Operating Model
Architecture Function

Here, you see **value delivery chain contributions** an **Architecture Function** typically makes within an Enterprise.



Many areas in which an **Architecture Function makes** its **contributions span multiple value stages** of the generic **Enterprise Operating Model**.



Each of the **Architecture Activity Areas** can be further **detailed** and **broken down** into **specific contributions** made by an **Architecture Function**.

Architecture contributions can be **viewed** as **services** that an **Architecture Function offers** other **functions** in the **context** of an **enterprise operating model**.

Lecture Agenda

- Classical Architecture
- Enterprise Business and Operating Model
- System
- Architecture Disciplines
- System Architecture
- Software Architecture

System
Overview

After introducing the context in which an architecture function make their contributions, we consider system as another central concept.

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).

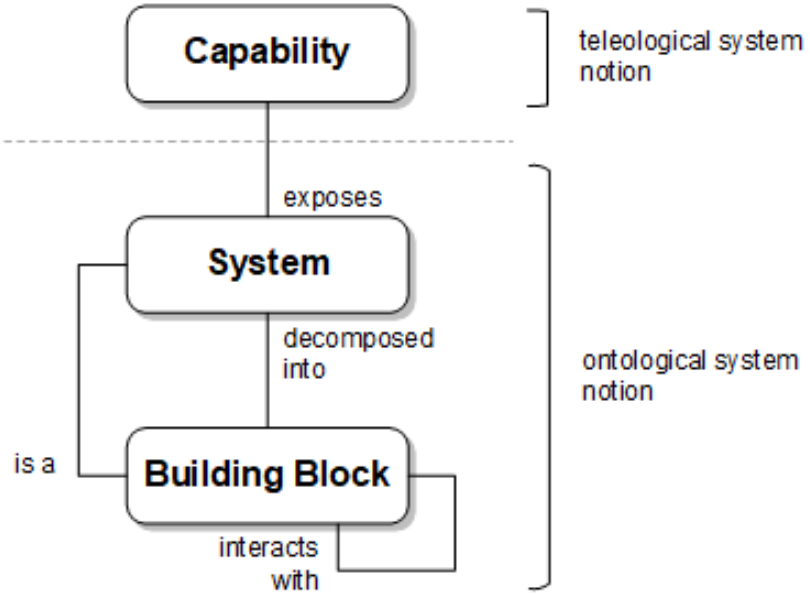
For example, a family and a football team are **social systems**, while a car or a washing machine are **technical systems**.

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.

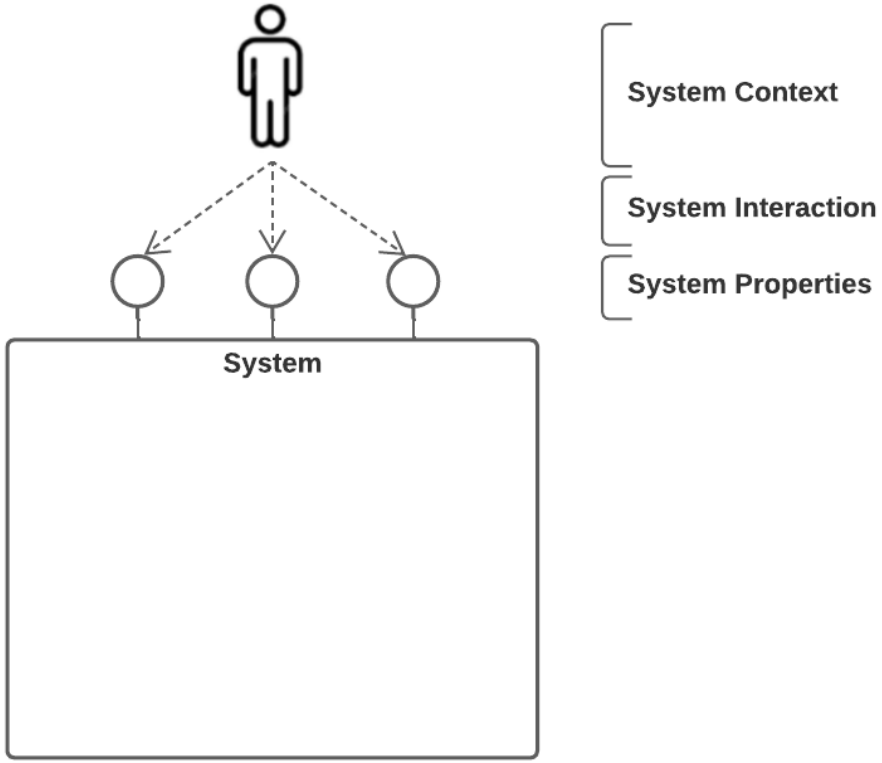
On the other hand the **ontological system notion**, which deals with the construction of a system — i.e. with its composition, environment, structure and production



System
Teleological versus ontological system notion

System Theory focuses on how the **different parts** of a **system interact** and how **systems function** over time and **in the context of larger systems**.

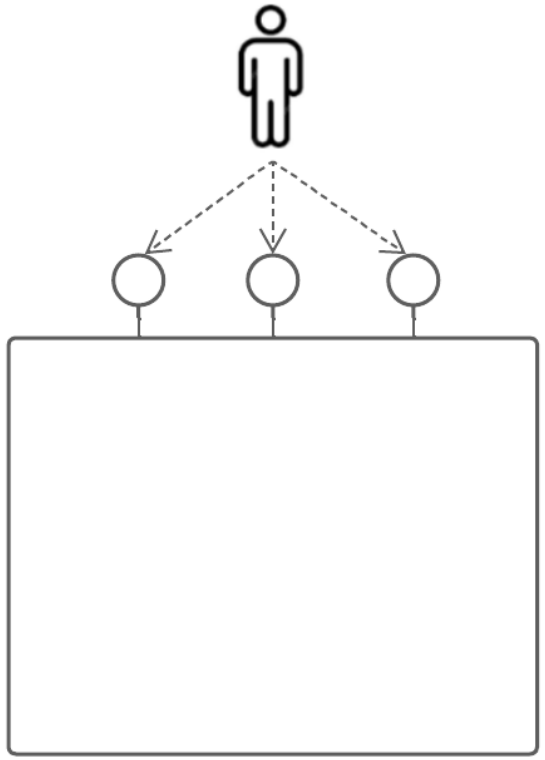
The **Theory of Systems**[1][2] provides us with a solid foundation on which we observe fundamental System aspects.



[1] [Systems Theory 2024]
[2] [Luhmann 2024]

System
Teleological versus ontological system notion

The **Teleological Perspective** refers to the **view of a system** in terms of its **goals**, **purposes** or **end states** – essentially, this **perspective** is about **looking** at a **system from the outside**.



A **System** can be viewed from the **outside**.
This view is called **Teleological** (also: **black-box**) **perspective**.

As a **User** of a **System** this is the perspective we are primarily (if not exclusively) interested in.

It tells us **how we can interact** with the **System** and **benefit** from its **value proposition**.

System
Teleological versus ontological system notion

The **Teleological Perspective** refers to the **view of a system** in terms of its **goals**, **purposes** or **end states** – essentially, this **perspective** is about **looking** at a **system from the outside**.

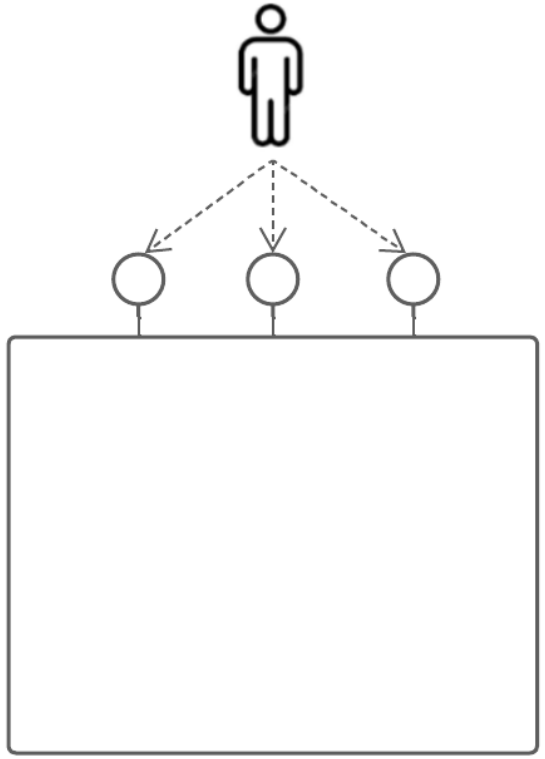
A **System** can be viewed from the **outside**.
This view is called **Teleological** (also: **black-box**) **perspective**.

If the **System** was a **Car**, then this is how a **Teleological Perspective** *onto* the Car-System might look like.

System
Teleological versus ontological system notion

The **Ontological Perspective** examines a system's **components**(1) and their inherent as well as co-operative **relationships**, **constituting** the externally visible **properties** – it is a **view into a system**.

We can also zoom into a **System**. This view is called **Ontological** (also: **white-box**) **perspective**.



We take this **View** when we need to **understand** (e.g. as an Architect) **how the System is constructed** internally in order to **establish** what it **offers** to the **outside world**.

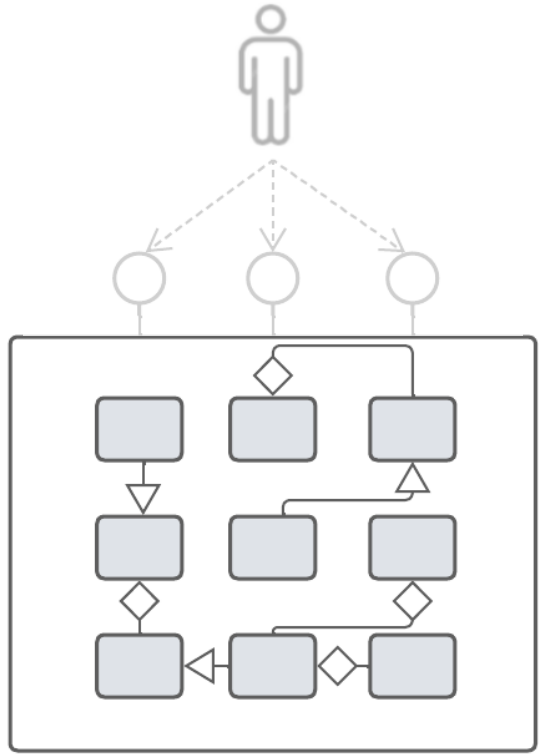
(1) Component = System Building Block = Building Block = Sub-System

System
Teleological versus ontological system notion

A **Static View** illustrates **the basic relationships** that exist **between building blocks** without limiting these to interaction relationships. It **provides** an **overview** of the **building blocks** at **design time**.

We take this **View** when we need to **understand** (e.g. as an Architect) **how the System is constructed** internally in order to **establish** what it **offers** to the **outside world**.

We can also zoom into a **System**. This view is called **Ontological** (also: **white-box**) **perspective**.



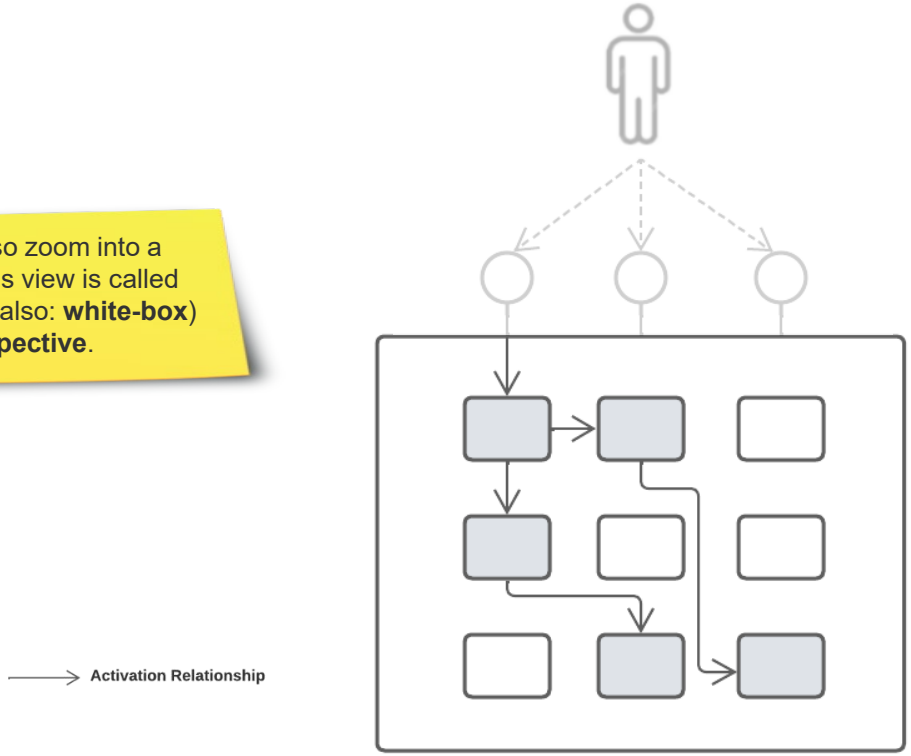
In a **Static View**, we are interested in the **Building Blocks** of the system and the **fundamental relationships** that exist among them.

System
Teleological versus ontological system notion

A **Dynamic View** illustrates **how components interact** and communicate with **each other** to **realise** the externally visible **properties of systems** – it shows the **interaction** of components at **execution time**.

We take this **View** when we need to **understand** (e.g. as an Architect) **how the System is constructed** internally in order to **establish** what it **offers** to the **outside world**.

We can also zoom into a **System**. This view is called **Ontological** (also: **white-box**) **perspective**.



In a **Dynamic View**, we want to understand how **System Building Blocks cooperate** in order to **realise System Properties**.

System
Teleological versus ontological system notion

The **Ontological Perspective** examines a system's **components**(1) and their inherent as well as co-operative **relationships**, **constituting** the externally visible **properties** – it is a **view into a system**.

We can also zoom into a **System**. This view is called **Ontological** (also: **white-box**) **perspective**.

AI-generated (OpenAI)

If the **System** was a **Car**, then this is how an **Ontological Perspective** *into* the Car-System might look like.

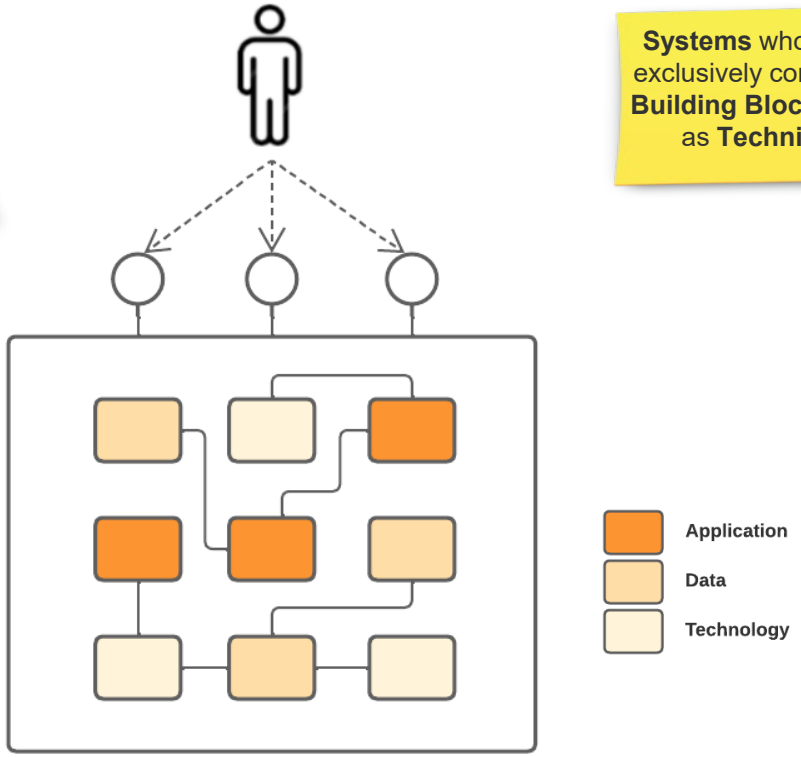
(1) Component = System Building Block = Building Block = Sub-System

System
Building Blocks – Technical Systems

A **Technical System** is a configuration of **interconnected technical building blocks** that provide functions used by other systems or humans.

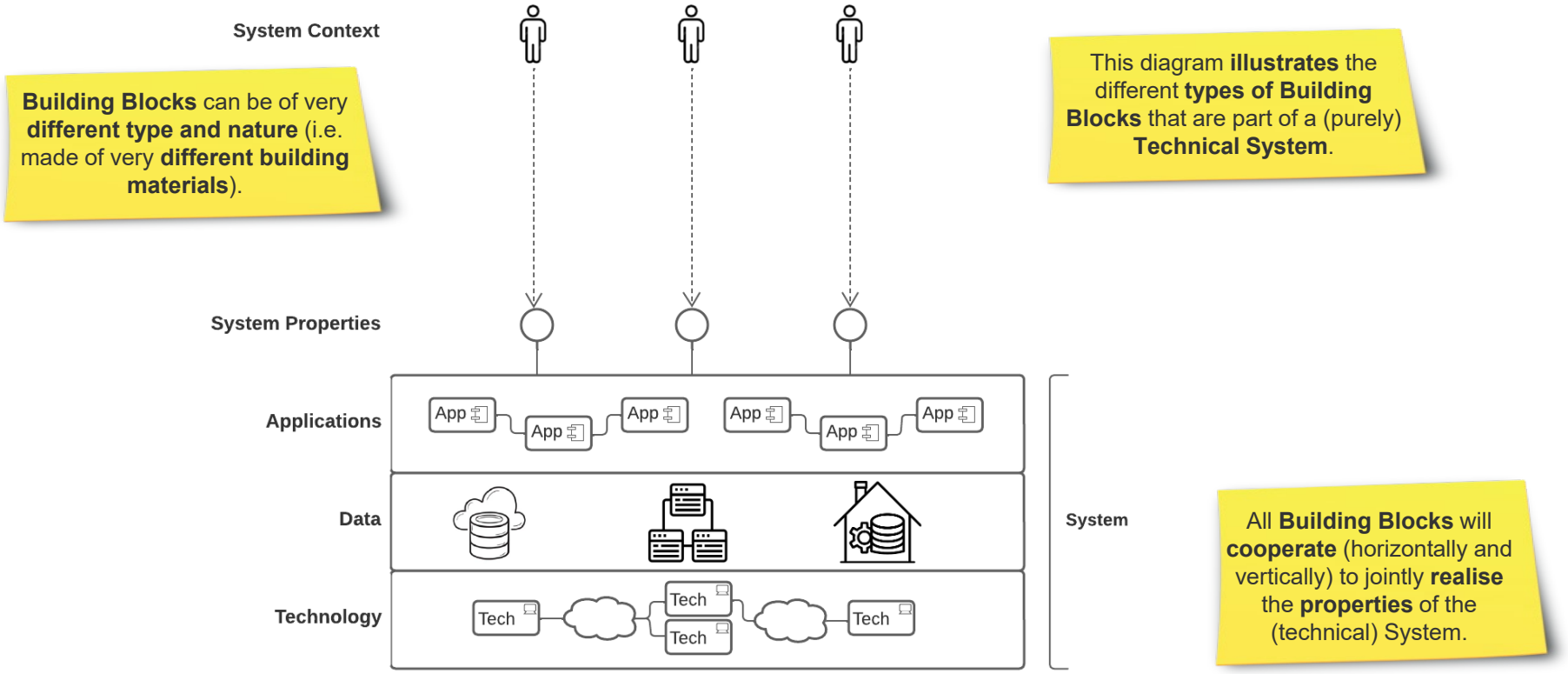
Building Blocks can be of very **different type and nature** (i.e. made of very **different building materials**).

Systems whose **components** exclusively consist of **Technical Building Blocks**, are referred to as **Technical Systems**.



System
Building Blocks – Technical Systems

A **Technical System** is a configuration of **interconnected technical building blocks** that provide functions used by other systems or humans.



Building Blocks can be of very **different type and nature** (i.e. made of very **different building materials**).

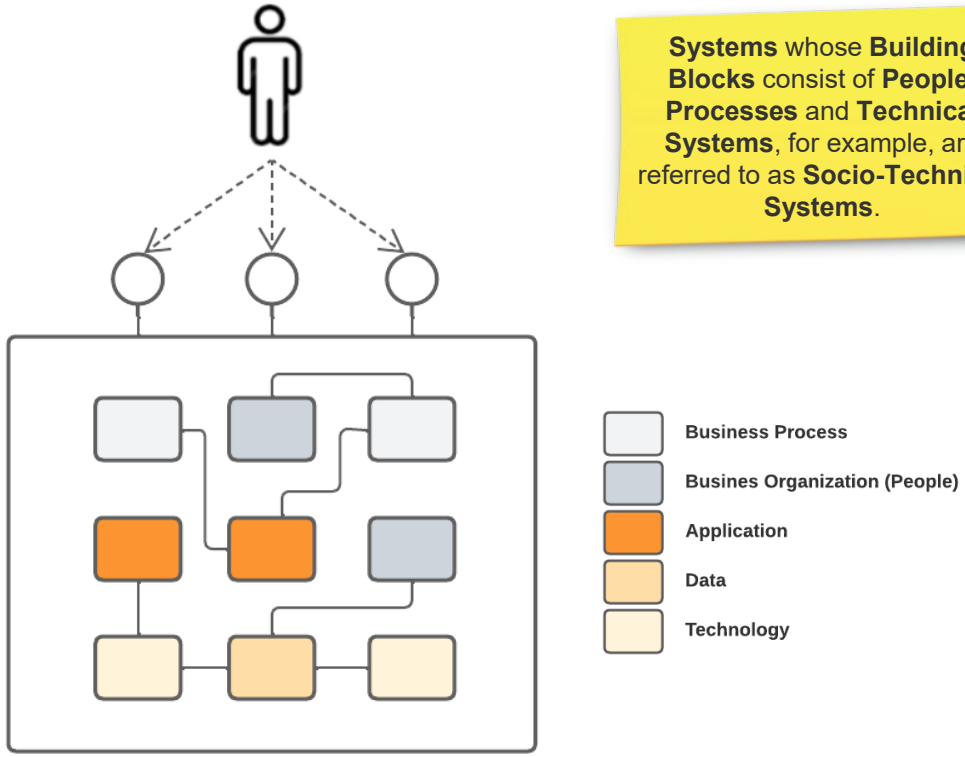
This diagram **illustrates** the different **types of Building Blocks** that are part of a (purely) **Technical System**.

All **Building Blocks** will **cooperate** (horizontally and vertically) to jointly **realise** the **properties** of the (technical) System.

System
Building Blocks – Socio-Technical Systems

A **Socio-Technical System emphasizes** the **interaction between technology**, **people** and the **organisational** or **social context** in which they **operate**.

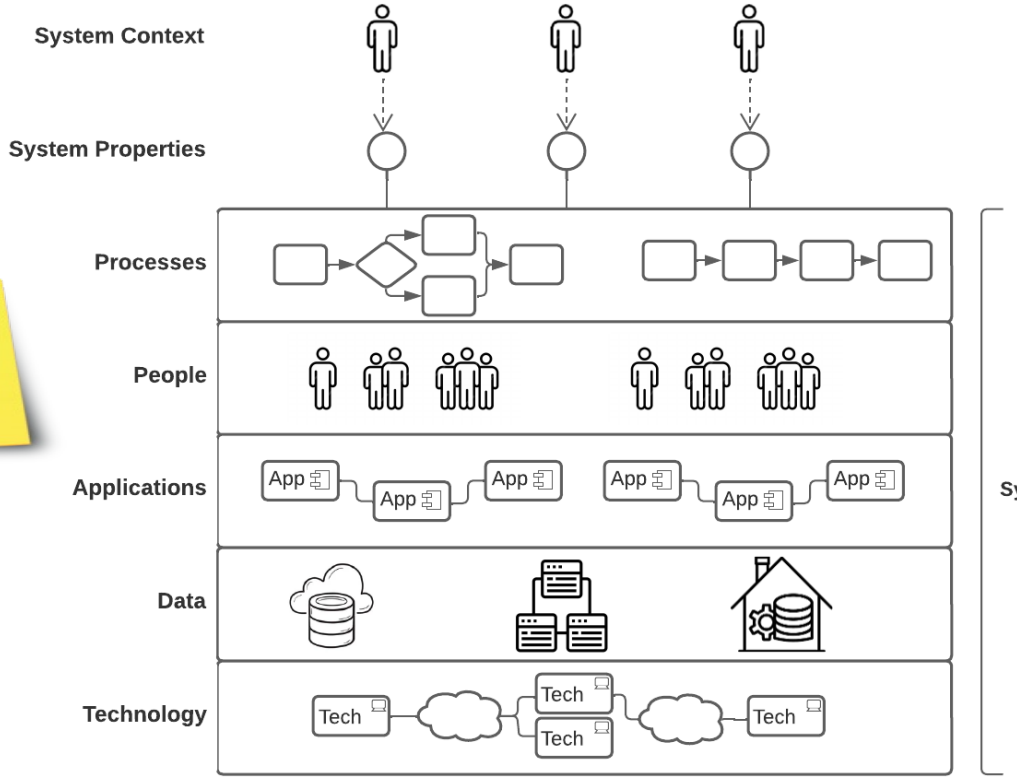
Building Blocks can be of very **different type and nature** (i.e. made of very **different building materials**).



Systems whose **Building Blocks** consist of **People**, **Processes** and **Technical Systems**, for example, are referred to as **Socio-Technical Systems**.

System
Building Blocks – Socio-Technical Systems

A **Socio-Technical System emphasizes** the **interaction between technology**, **people** and the **organisational** or **social context** in which they **operate**.



This diagram **illustrates** the different **types of Building Blocks** that are part of a **Socio-Technical System**.

Building Blocks can be of very **different type and nature** (i.e. made of very **different building materials**).

All **Building Blocks** will **cooperate** (horizontally and vertically) to jointly **realise** the **properties** of the (socio-technical) System.

Lecture Agenda

- Classical Architecture
- Enterprise Business and Operating Model
- System
- Architecture Disciplines
- 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 **solution architecture** (note that solution architecture comes closest to software architecture in the sense that software architecture activities are usually embedded in solution architecture).

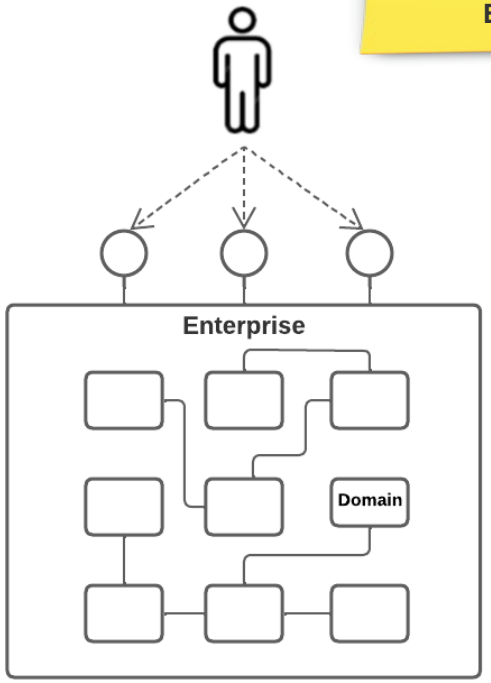
On the one hand, we differentiate architecture disciplines along the **granularity** of the considered systems as well as the **planning horizon** that the respective discipline takes.

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

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

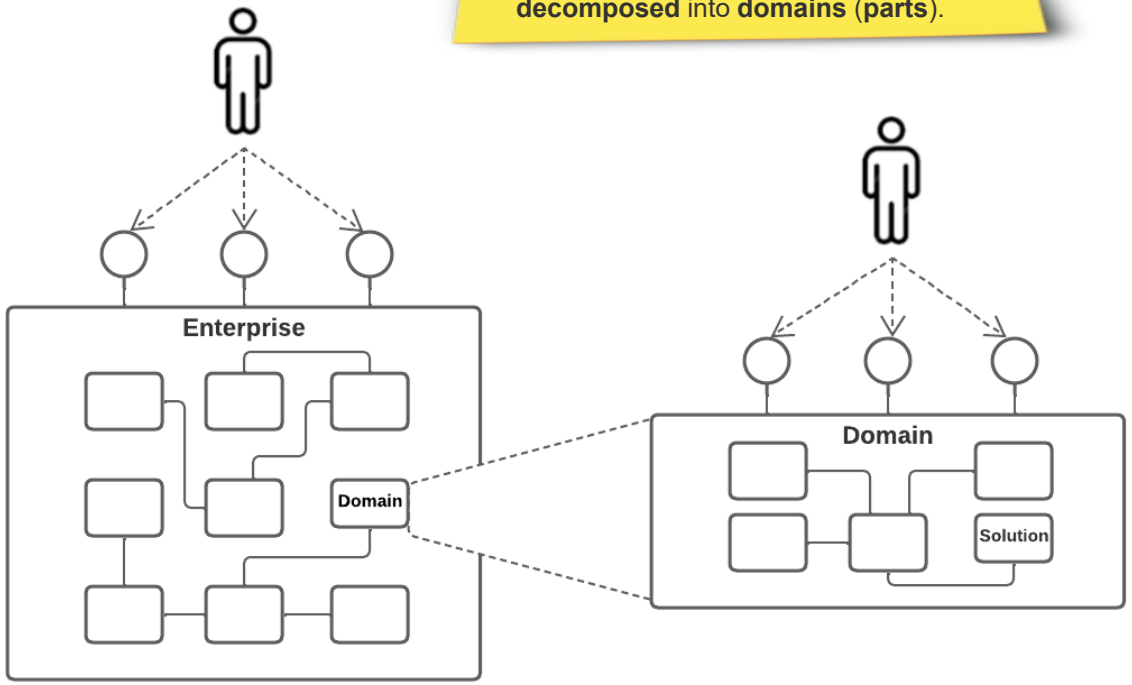
When we looked at essential concepts, we said that **Systems** (**whole**) consist of **Building Blocks** (**parts**).



Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

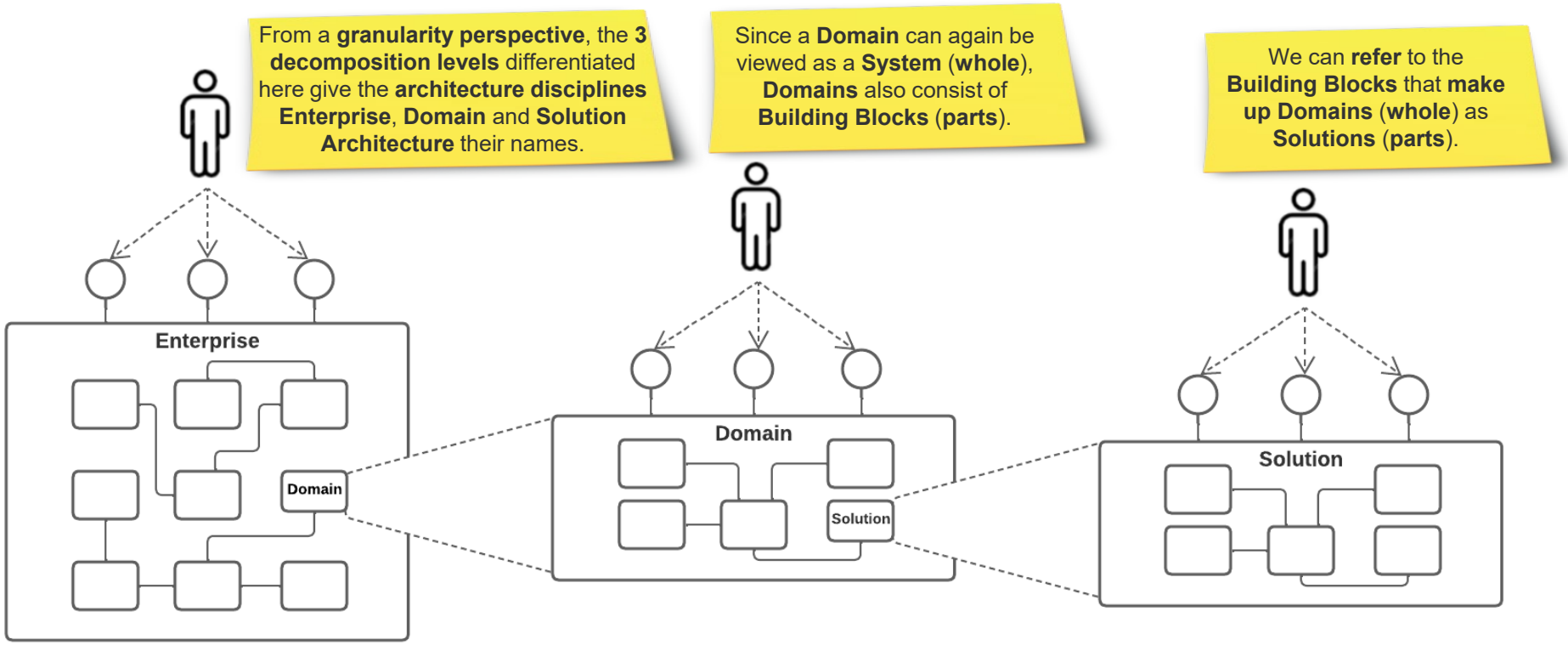
Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

Based on this consideration, we can say that – if an enterprise is viewed as a system – the **enterprise** is a **whole** that is **decomposed** into **domains** (**parts**).



Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.



Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

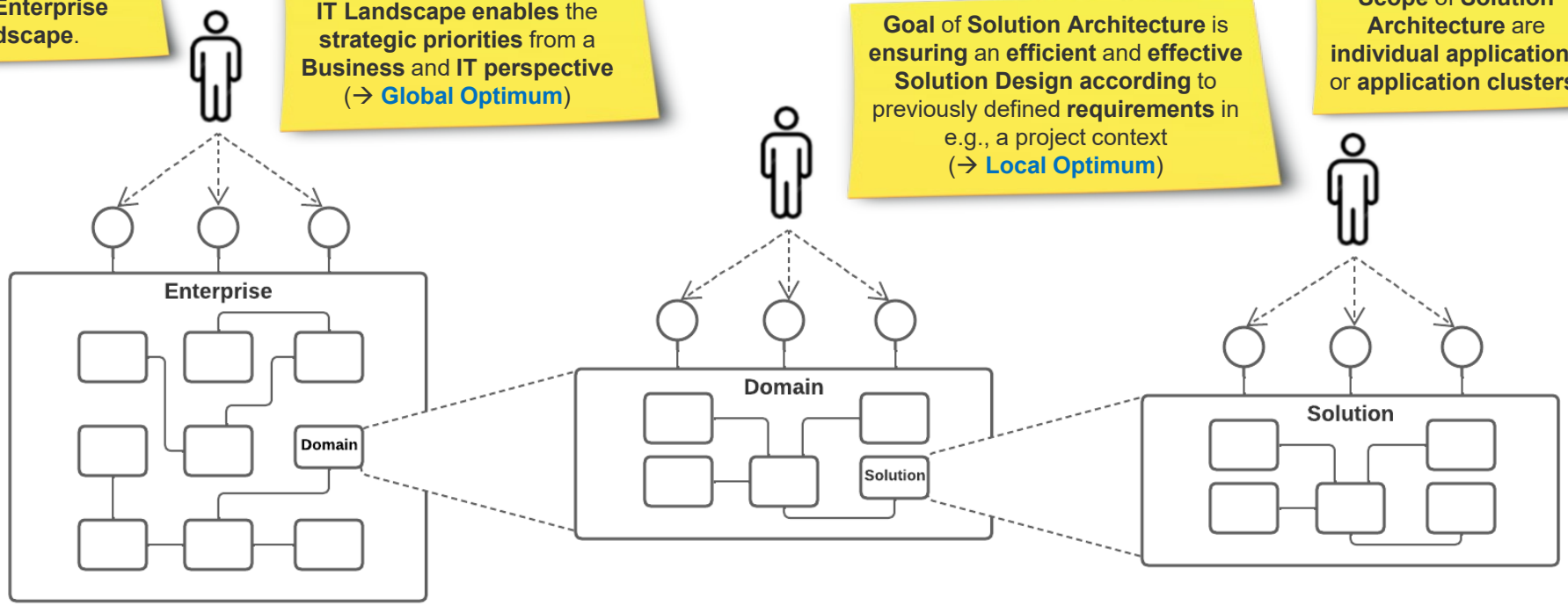
Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

Scope of **Enterprise Architecture** is an **entire Enterprise Landscape**.

Goal of **Enterprise Architecture** is to **ensure** an **IT Landscape enables** the **strategic priorities** from a **Business** and **IT perspective** (→ **Global Optimum**)

Goal of **Solution Architecture** is **ensuring** an **efficient** and **effective Solution Design according** to previously defined **requirements** in e.g., a project context (→ **Local Optimum**)

Scope of **Solution Architecture** are **individual applications** or **application clusters**.



Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

The **City Planner** or **City Architect** of the wonderful city of **Basel** has a very **strategic** and **far-reaching planning** horizon.

The **classic Architectural Disciplines** serve as a nice **analogy** here.

At the same time, his **view spans** the **city** in its **entirety** and **full breadth**.

It is the **Enterprise Architect** who corresponds to the **City Planner** or **City Architect**.

More **fine-grained planning perspectives** he **delegates** to his City Planner colleague: the **Neighbourhood Architect**.

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

The only **major difference** is that the **space** they have to plan **is smaller**.

The **Neighbourhood Architect** has a planning and working **perspective** very **similar** to that of the **City Architect**.

As soon as a **zoning plan** is **in place**, the **Neighbourhood Architect delegates** to the **Architect** who designs individual **buildings**.

It is the **Domain Architect** who corresponds to the **Neighbourhood Architect**.

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

The classical **Architect** now **designs** individual **buildings** in the zoning plan intended for them.

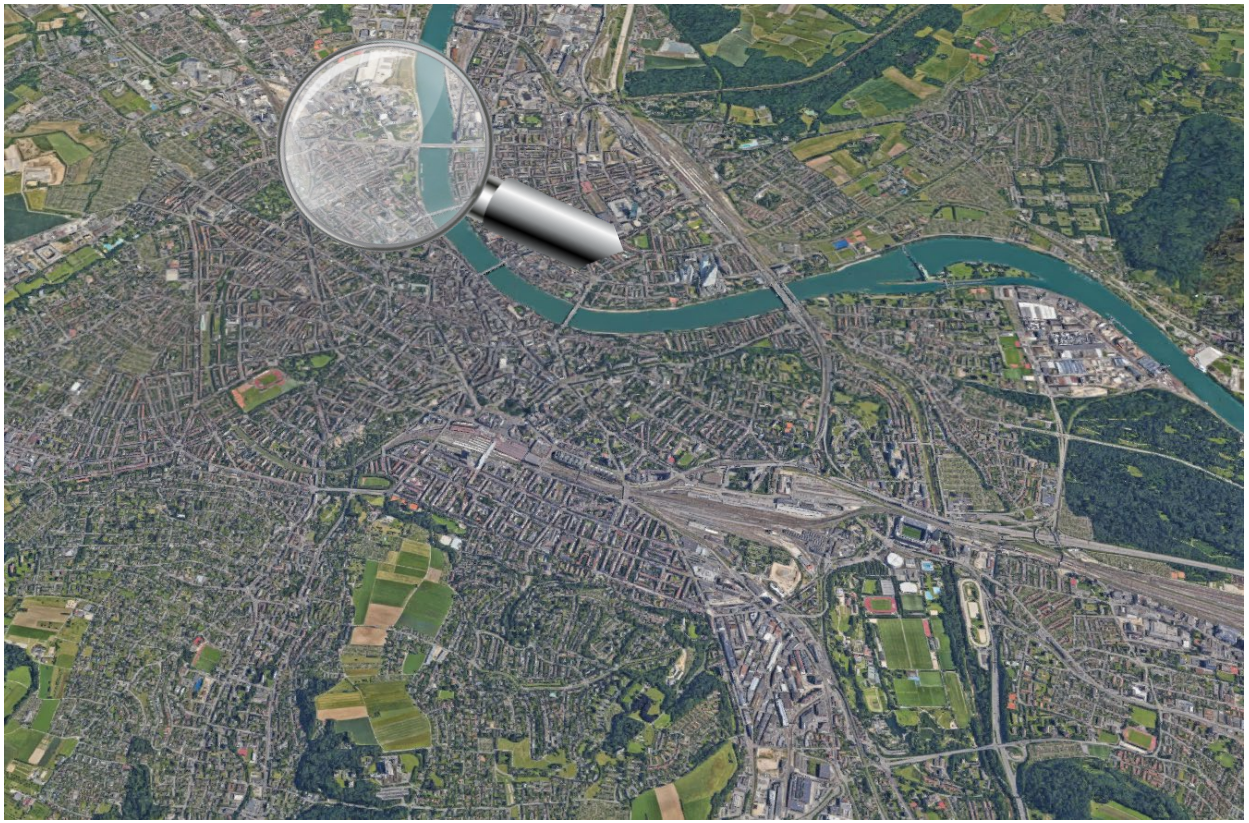
The **Architect** designs the **architecture** of the **building** such that it **addresses** the client's **requirements**, pareto-optimally.

His **view** is **not** so much a **Portfolio Planning** or **Evolutionist** perspective but **more** of a **Solutionist Perspective**.

It is the **Solution Architect** who corresponds to the **classical Architect**.

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.



Another **example** in the **beautiful Rhine city** of **Basel** ...

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

... **shows** a **Company Site** (Novartis Campus) as a **Domain** …

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

... ... and the **Cloud Building** (HR Building and Canteen on the **Novartis Campus**) as the work of a **classical Architect**(1).

(1) [Gehry 2024]

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

A final **example** in **sunny Basel**
...

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

... **shows** a **part** of the **public transport system** (Swiss Railway Station) as a **Domain** ...

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

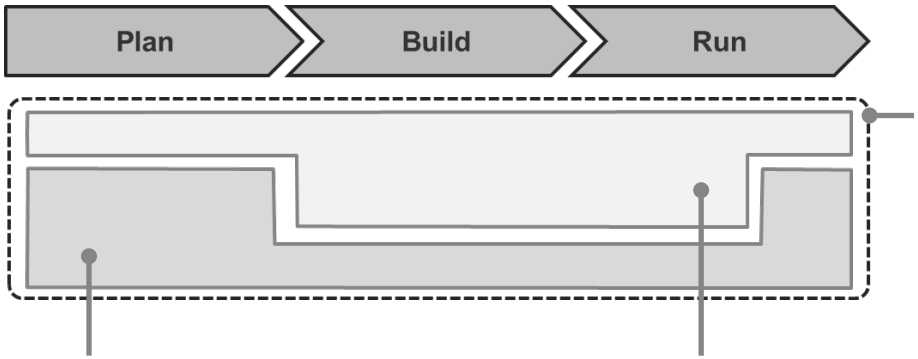
Three key Architecture Disciplines correspond to **different levels** of **granularity**, which in turn are based on the System concept Whole versus Part.

... and the **SBB Main Building** as the work of a **classical Architect**(1).

(1) [La Roche 2024]

Architecture Disciplines
Enterprise vs. Domain vs. Solution Architecture

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



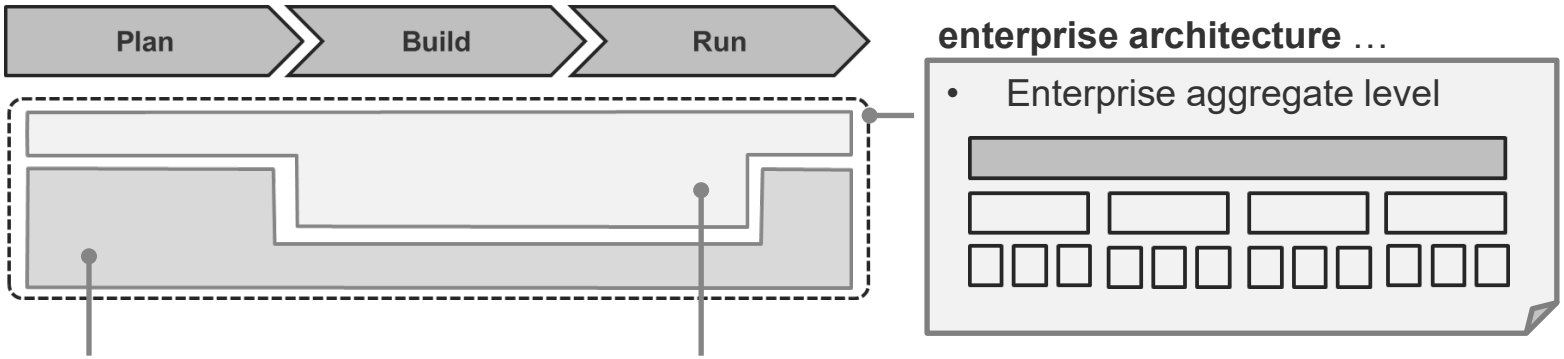
enterprise architecture …
- ensures alignment and integration between domain and solution architecture
- Equips disciplines with methods, and policies

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

solution 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 vs. Domain vs. Solution Architecture

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



enterprise architecture …

- Enterprise aggregate level

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

solution architecture …

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

Architecture Disciplines
Planning versus transformative architecture

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

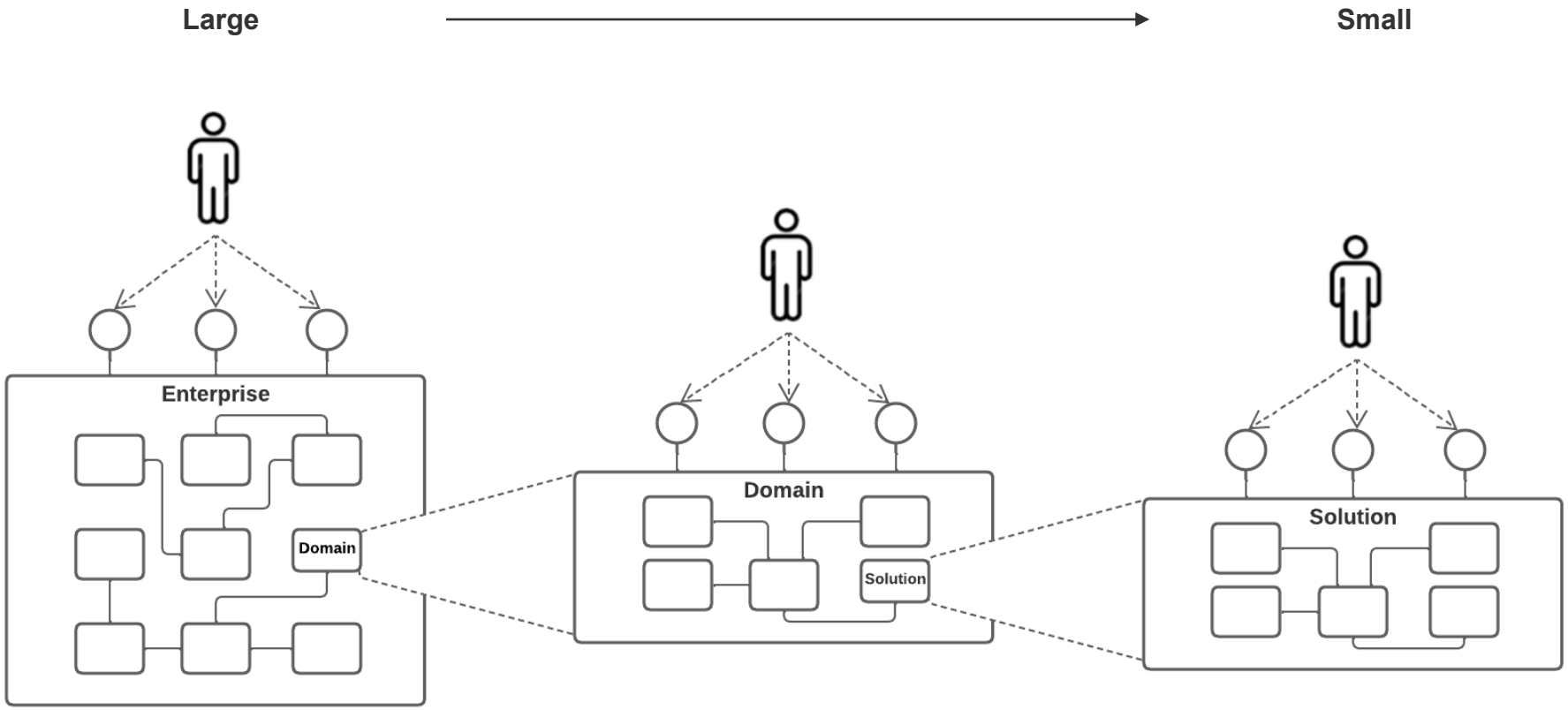
solution architecture
domain architecture

Do the right thing

Do the thing right

Architecture Disciplines
Fun Fact

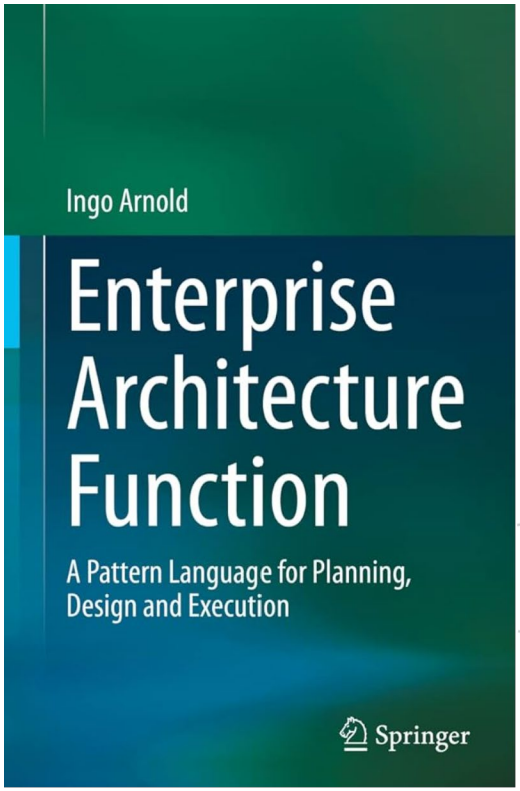
While **Architectural Development** often goes from the **large to the small**, the **development of Architects** takes the **opposite direction** – see my own biography.

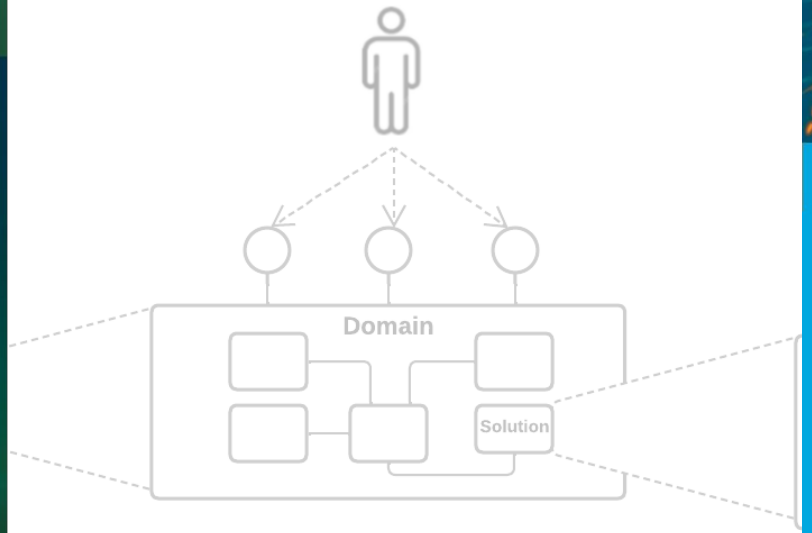


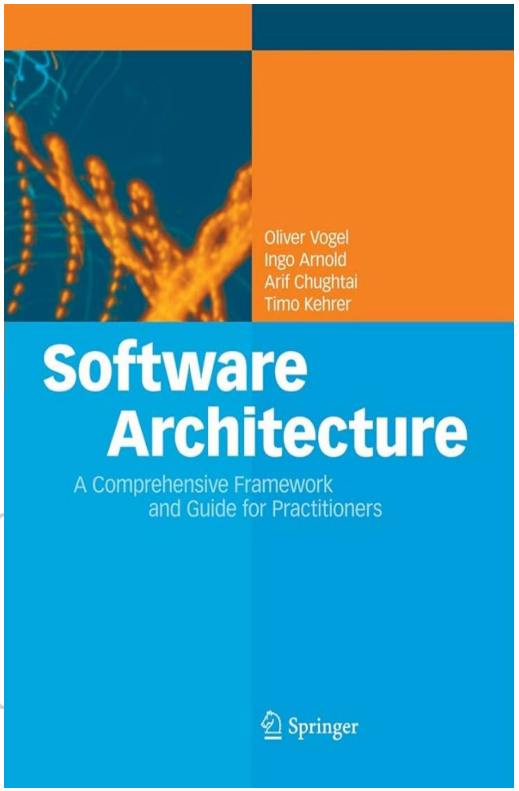
Architecture Disciplines
Fun Fact

While **Architectural Development** often goes from the **large to the small**, the **development of Architects** takes the **opposite direction** – see my own biography.

Enterprise Architecture (2022) ← **Solution Architecture** (2011)







Lecture Agenda

- Classical Architecture
- Enterprise Business and Operating Model
- System
- Architecture Disciplines
- System Architecture
- Software Architecture

System Architecture
Architecture of Systems

Before we take a closer look at what **Architecture** is, we recognise that **Architecture** is an **inherent attribute** of a broad variety of **different entity types** – which we will collectively refer to **Systems**.

The **Architecture** of **Technical Systems** (e.g., motor vehicles, computers)

The **Architecture** of **Urban Systems** (e.g., cathedrals, houses, cities, parks).

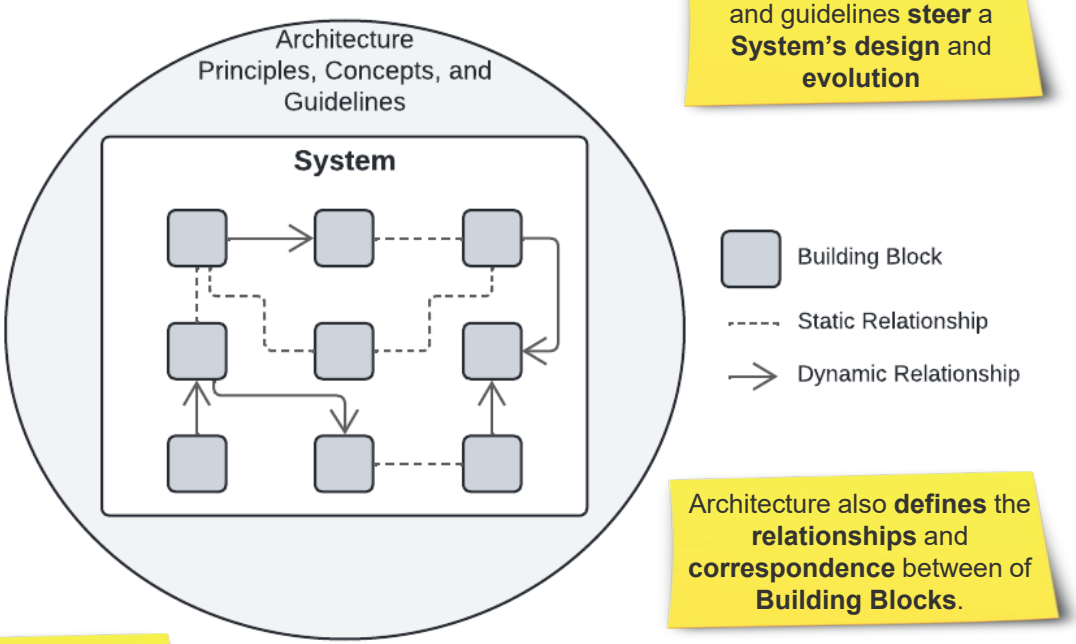
The **Architecture** of **Social Systems** (e.g., soccer teams, families, political parties).

System Architecture
Architecture of Systems

While **many varying definitions** of **Architecture** exist, the vast **majority shares characteristics** that Thierry Perroud alludes to in his definition.

Architecture defined by Thierry Perroud ([Perroud 2013])

- The **Architecture of a System** …
  - defines the **structure** and orderly arrangement of the **building blocks** (i.e., the parts) of the system (i.e., the whole)
  - the **building blocks' inherent** (static) as well as **collaborative** (dynamic) **relationships** with each other
  - as well as the **principles and guidelines** that **determine their design** and **evolution** over time



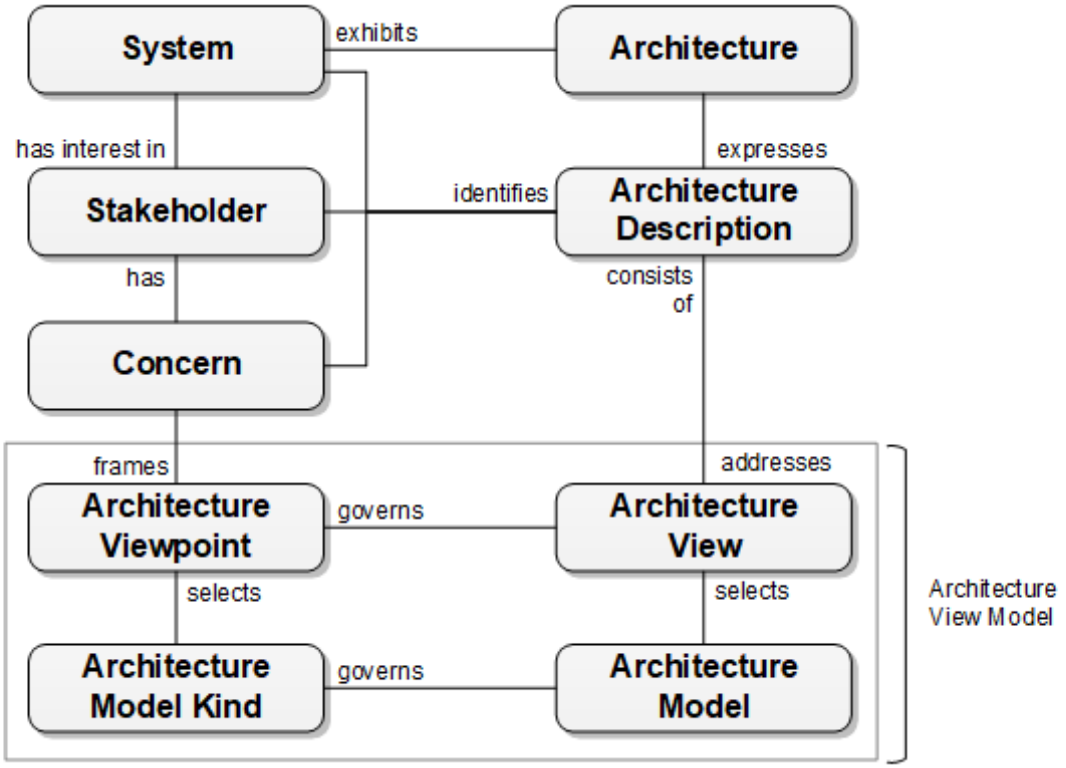
Architecture **principles** and guidelines **steer** a **System's design** and **evolution**

Architecture also **defines** the **relationships** and **correspondence** between of **Building Blocks**.

Architecture **defines** the **structure** and **arrangement** of a System's **Building Blocks**.

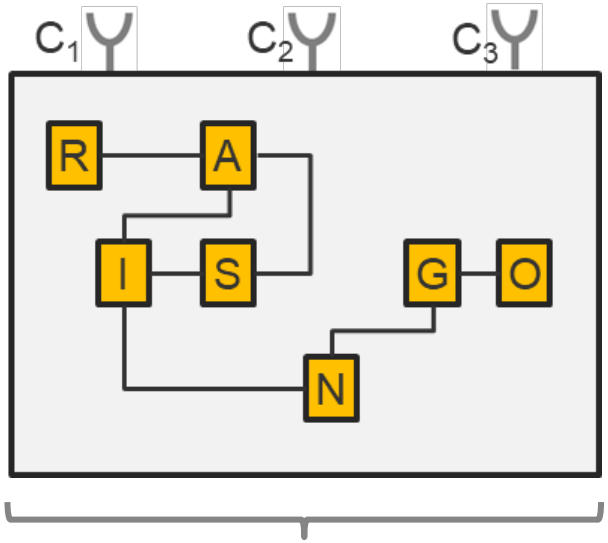
System Architecture
Architecture meta model

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

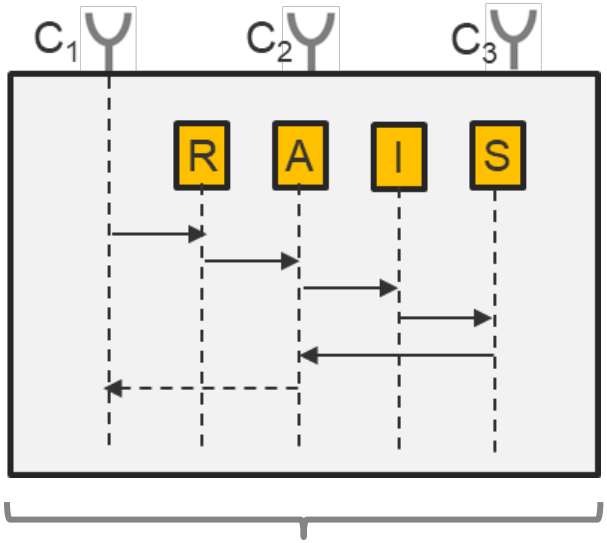


System Architecture
Dynamic versus static system architecture

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)

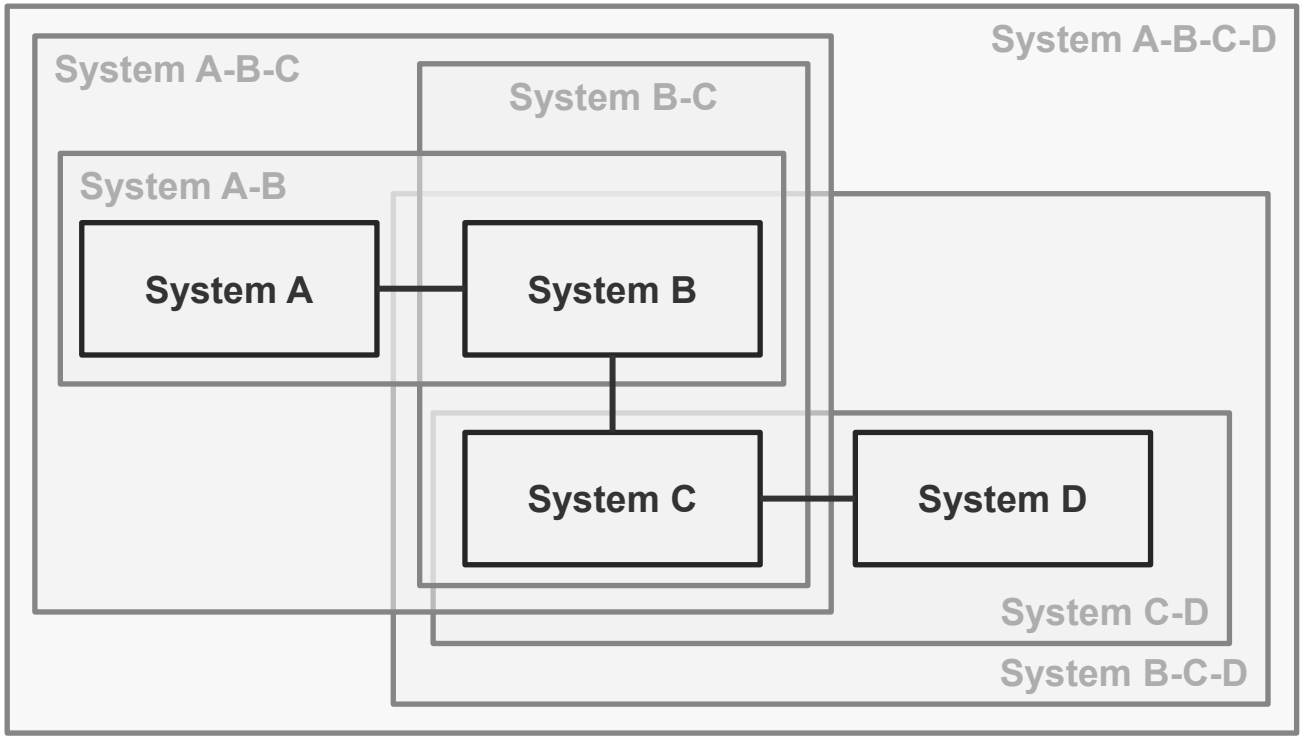


Dynamic System Architecture
(component collaboration relationships)

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.



System Architecture
Architecture evolution

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

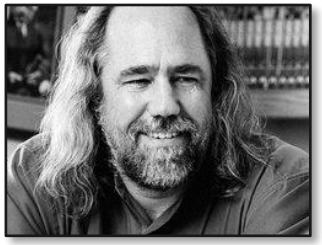
System Architecture
Architecture evolution

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

System Architecture

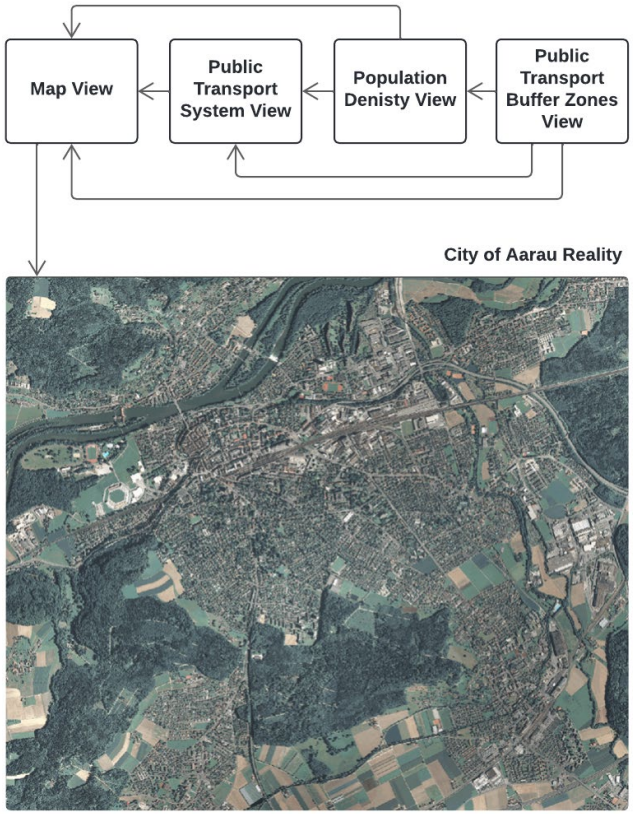
Architecture as entirety of significant design decisions

"**Architecture** is the **total** of **significant design decisions**, where **significant** is **measured** by **cost of change**. All architecture decisions are design decisions but not all design decisions are architecture decisions." [Booch 2009]



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.



Over the next slides to come, we will look at a **small example** of a **View Model** (in the **GIS domain**) consisting of **4 Views + 1 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.

Photo (Reality)

First, let's take a look at the **Reality of Aarau** – an **aerial photo** of the city of Aarau.

What we are looking at here is **not yet a view** – it is **reality**

Well, we can **criticise** the **above point right away**, since it is a **photograph** – i.e., **a model of 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)

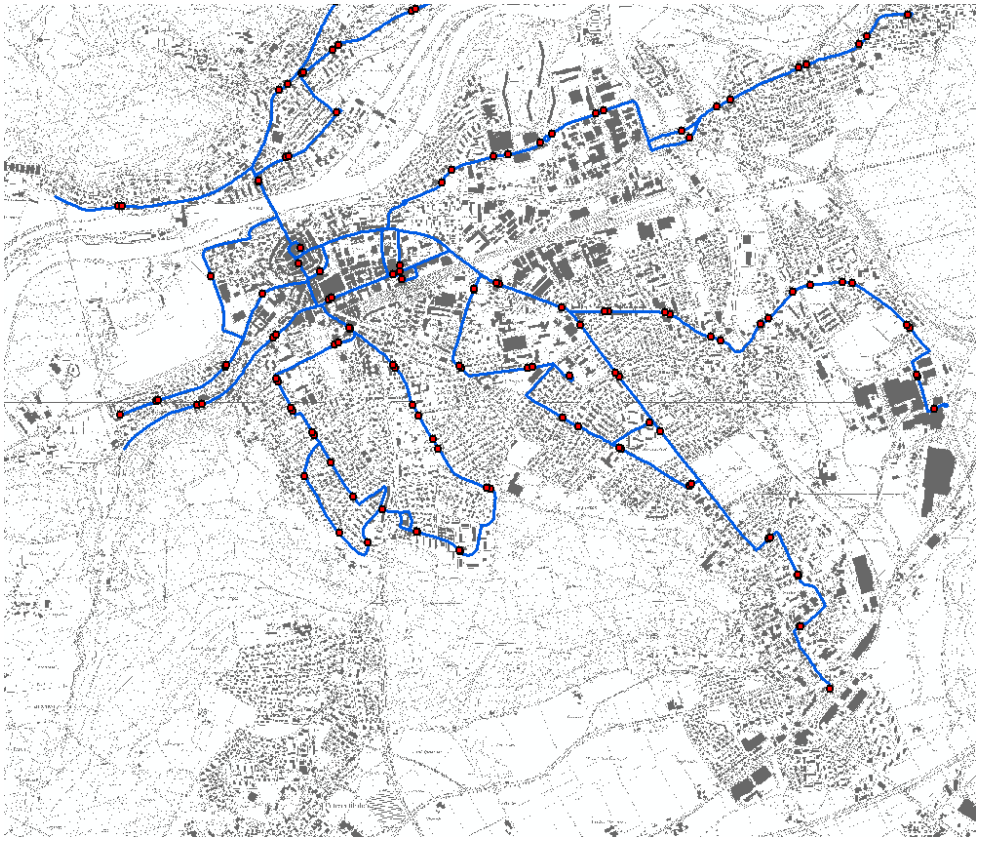
We use our good old familiar **Maps** as an **analogy** and look at a **geographical section of Switzerland** (i.e. Aarau) **as the System**.

The **Map reduces** the wide **variety of aspects** that **make up** the **real city of Aarau**.

What **remains recognisable** (i.e., remains) on this map are **urban structures** such as **houses** and **blocks**, **streets**, **railways** and other landscape features such as the **River Aare**.

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

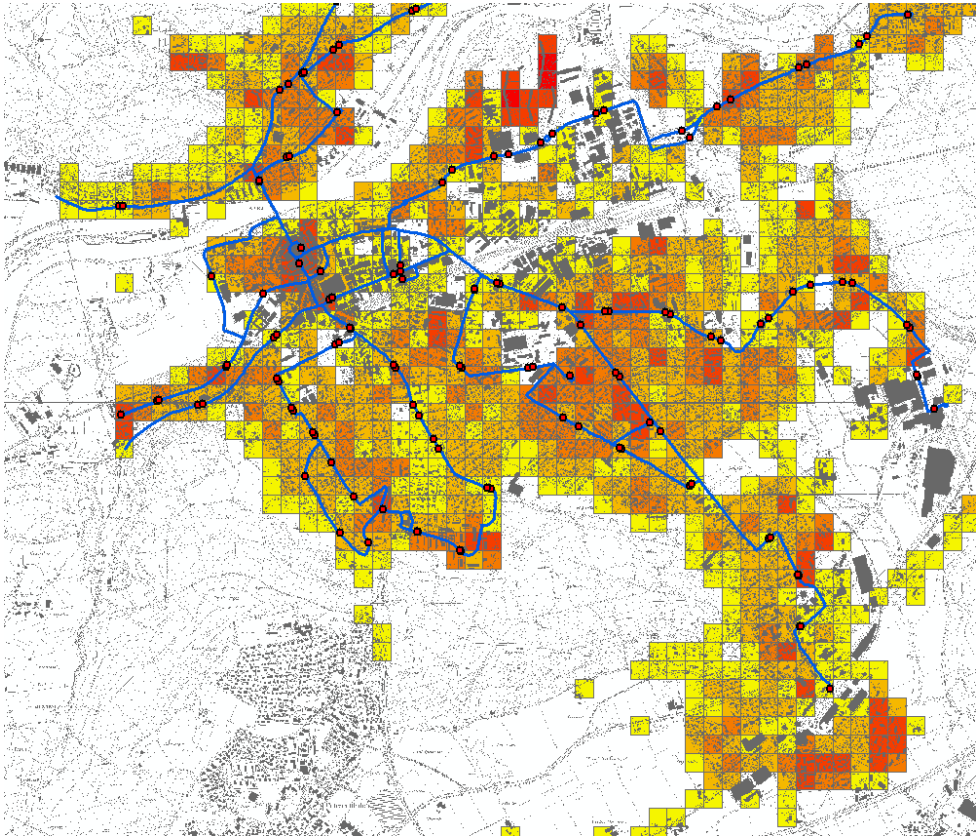
The **diagrams** we have on the right have been **generated by a GIS** (i.e., Geo Information System).

With the help of this system, we have the **local public transport routes** (i.e., tram and rail transport) displayed as **blue lines** and the respective **stops** as **red dots**.

The **nice thing** about this analogy is that the **Public Transport System view** can be **superimposed directly onto the map of the city of Aarau** – **synchronised via geo-coordinates**.

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

Using the GIS further, we now place a view (**Population Density view**) over the existing views.

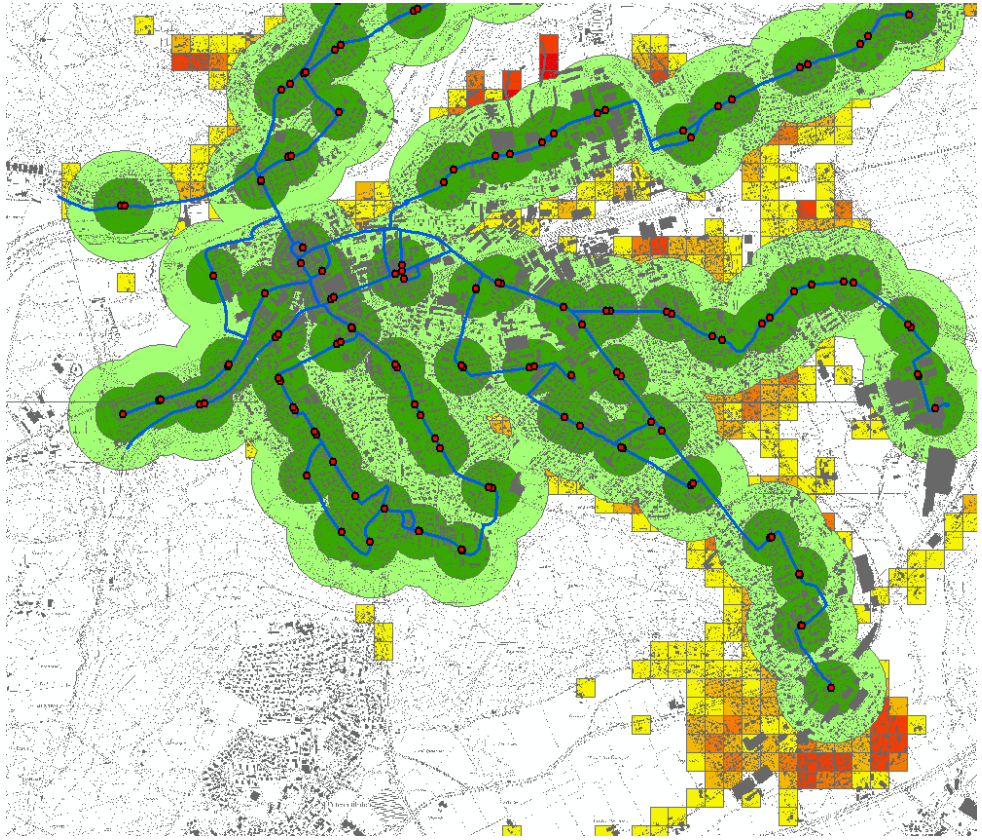
The Population Density view **shows the density of the population at a certain grain size** for the entire city of Aarau.

Areas in Aarau marked by **dark red quadrants show a very high population density**, while light yellow quadrants show a very low one, accordingly.

White areas are practically **not populated at all**.

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

Finally, the GIS gives us **Public Transport Buffer Zones** as a last view.

Like the previous ones, we **place this view on top of the stack of the other views** – like a **transparent foil**.

The new view introduces further information: in which **area** around a tram station is it **very easy to reach the stop** (dark green circle) and in which **area** is the **stop still ok to reach** (light green circle)?

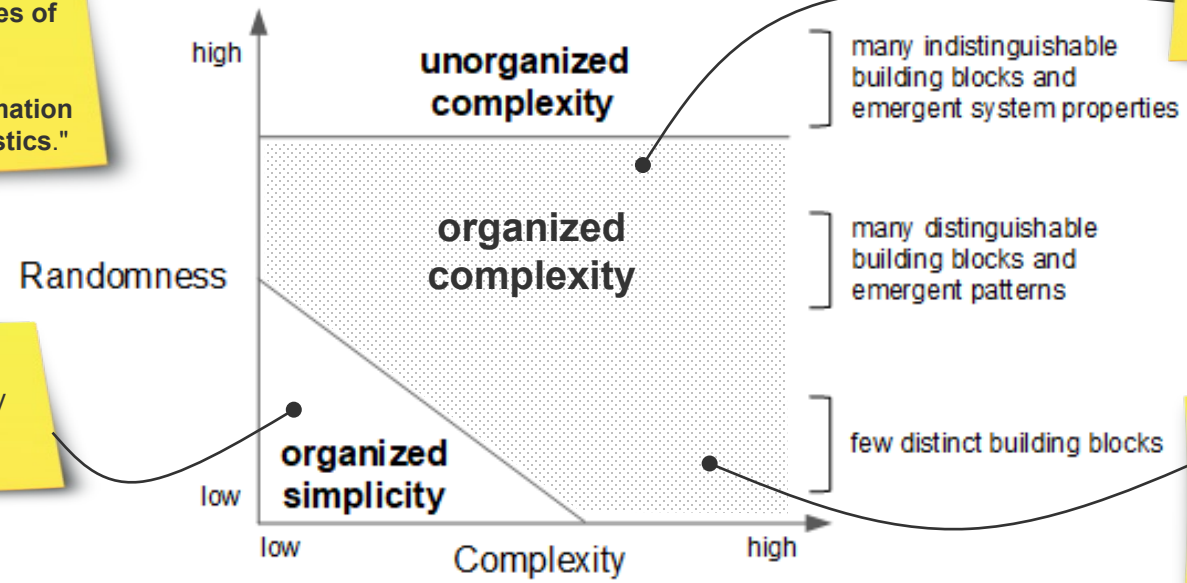
With this analogy we can see that the **power of View Models lies** in the **combination of views** and that view models **favour holistic as opposed to particularistic perspectives**.

System Architecture
Architecture relevance

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

According to Gerald **Weinberg** ([Weinberg 2001]), **Architecture deals** with the following **types of problems**:

"[. . .] **too complex for automation** and **too organised for statistics**."



Unorganised complexity is the segment of problems that must be **understood** and **processed statistically**.

Problems of this type can be understood and processed by machines (**organised simplicity**).

Finally, **organised complexity** is the segment that **architecture** disciplines (e.g., Enterprise Architecture) **primarily deal with**.

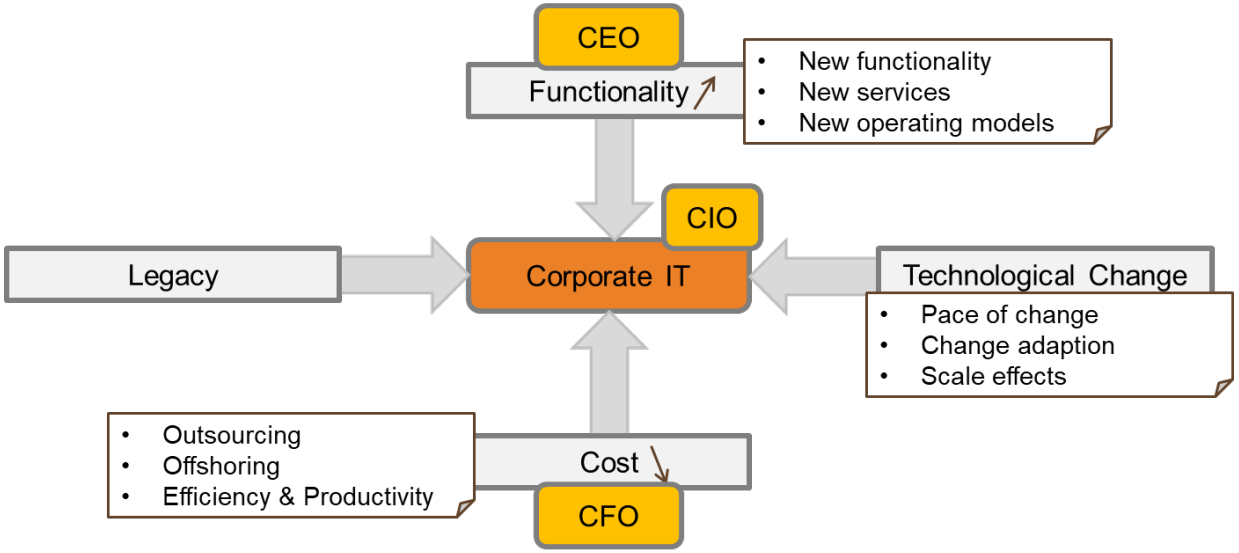
System Architecture
Architecture contribution in the enterprise

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

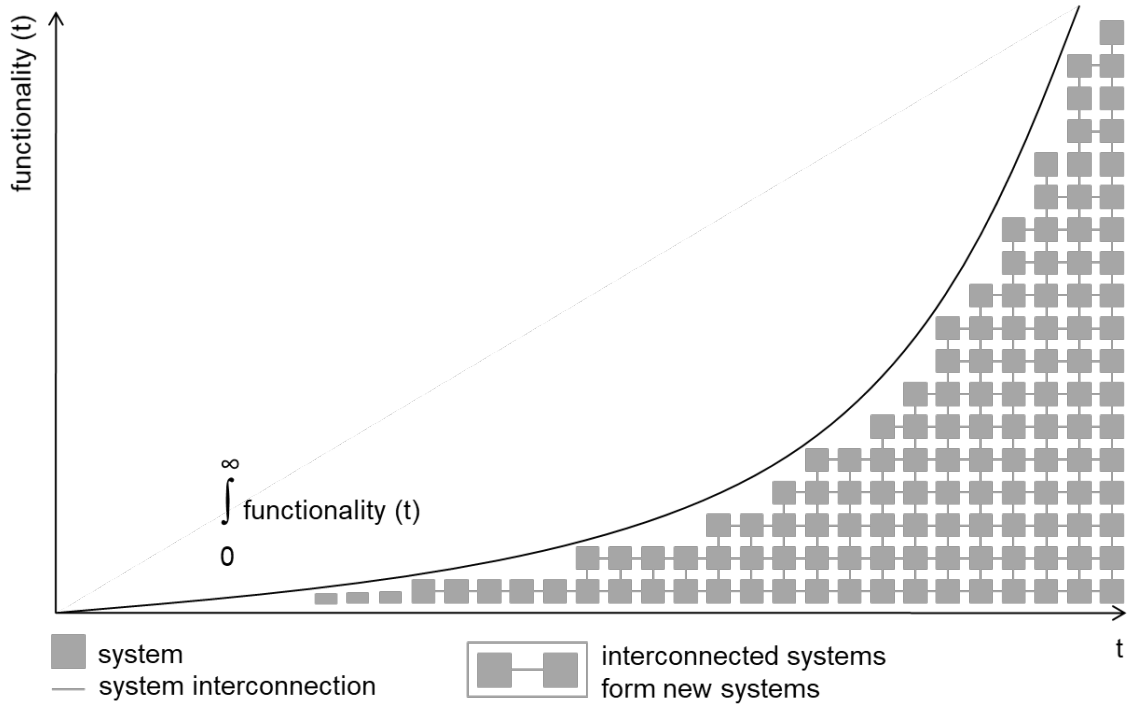
System Architecture
Architecture contribution in the enterprise

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.



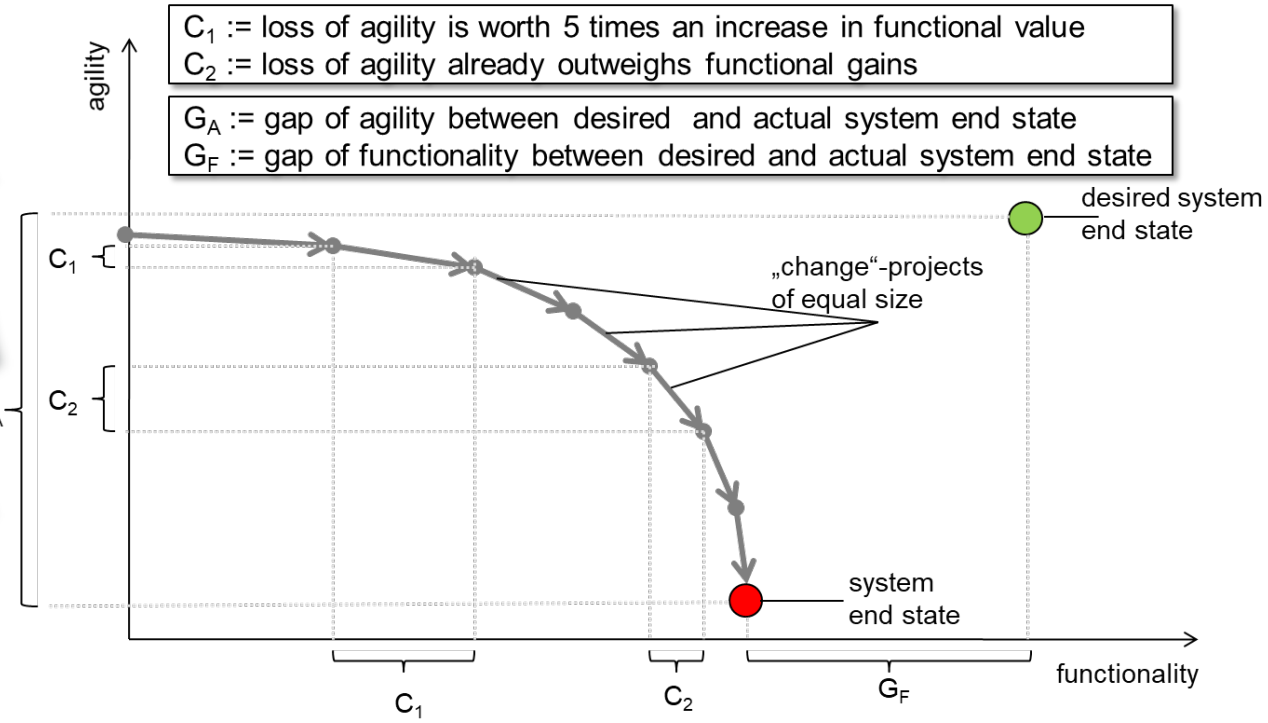
System Architecture
Architecture contribution in the enterprise

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]).

When **large systems are evolved** ...

... and these **systems' evolutionary fitness** is **not taken into account** ...

… sooner or later they will be in **system crisis**.



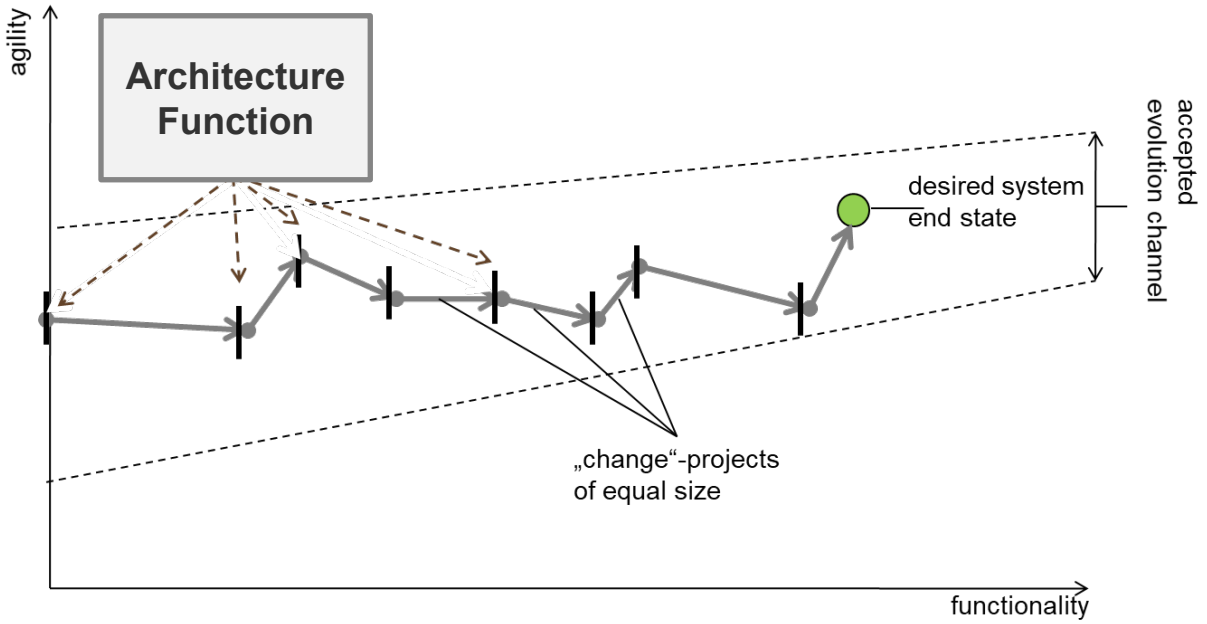
System Architecture
Architecture contribution in the enterprise

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

The **approach** to **mitigating** this **risk** is to ...

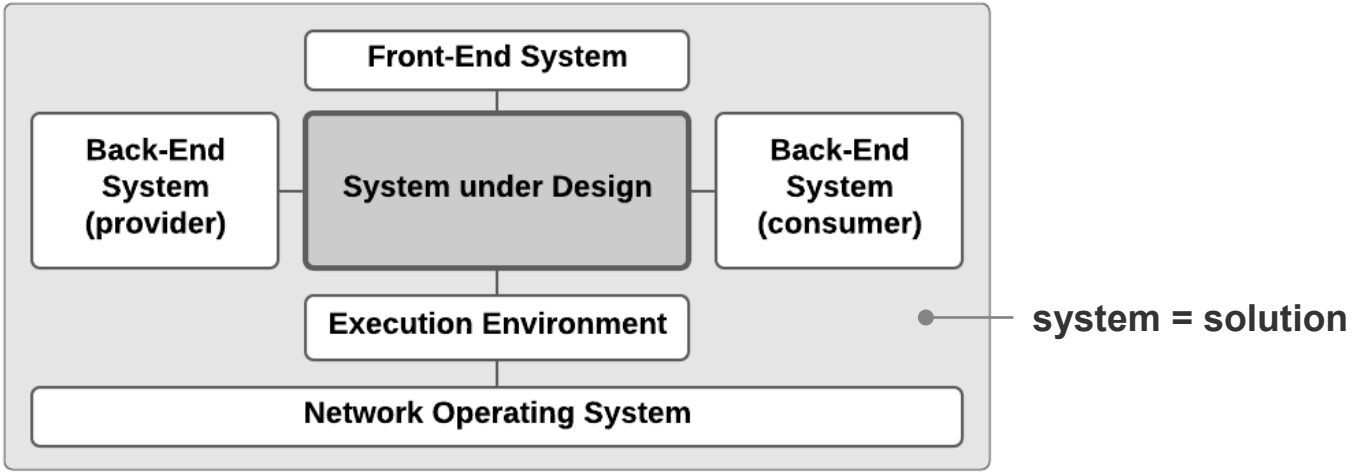
... **define** an **evolution corridor** for **each large system** ...

... and to **take** the **corridor into account** when making **investment/project alignment decisions**.



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).

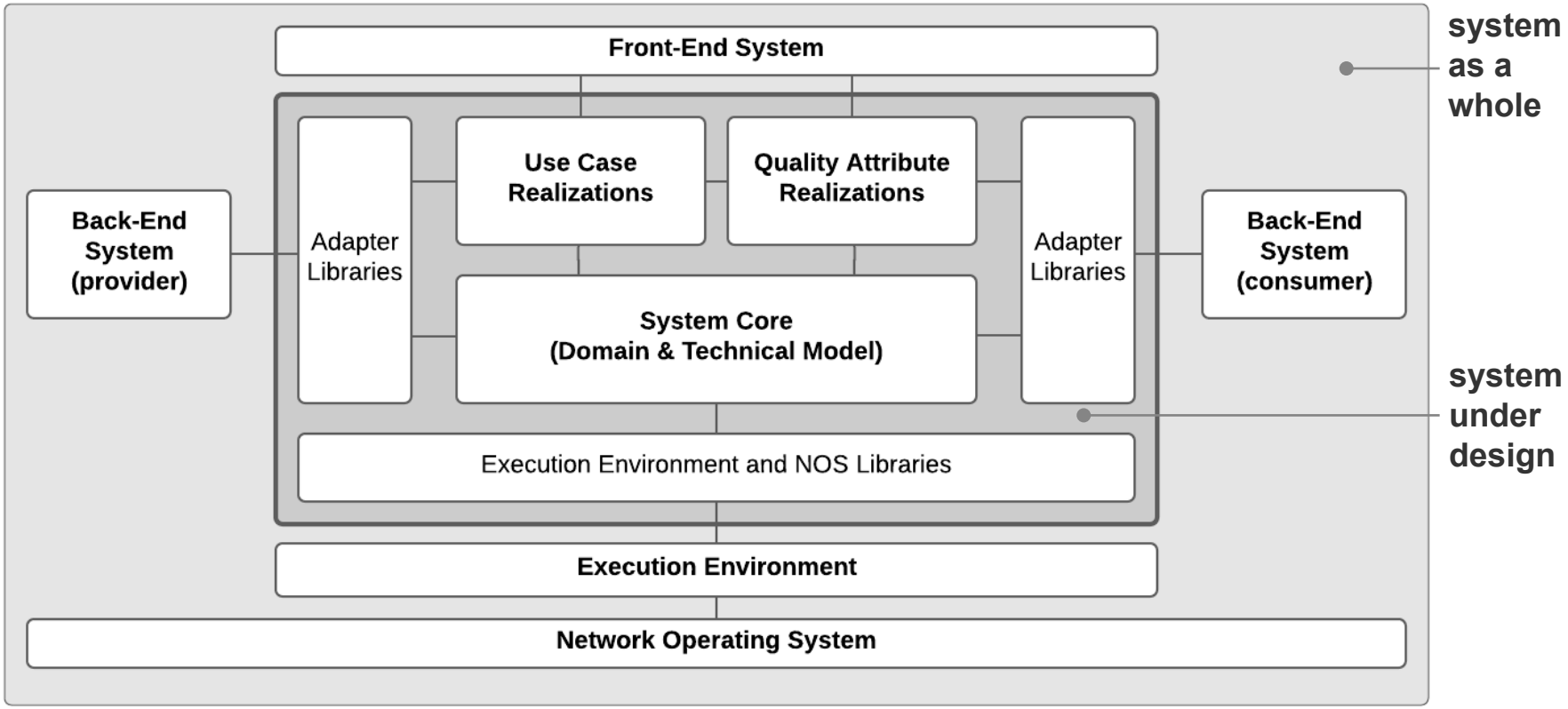
System Architecture
Architecture differentiation

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 ...

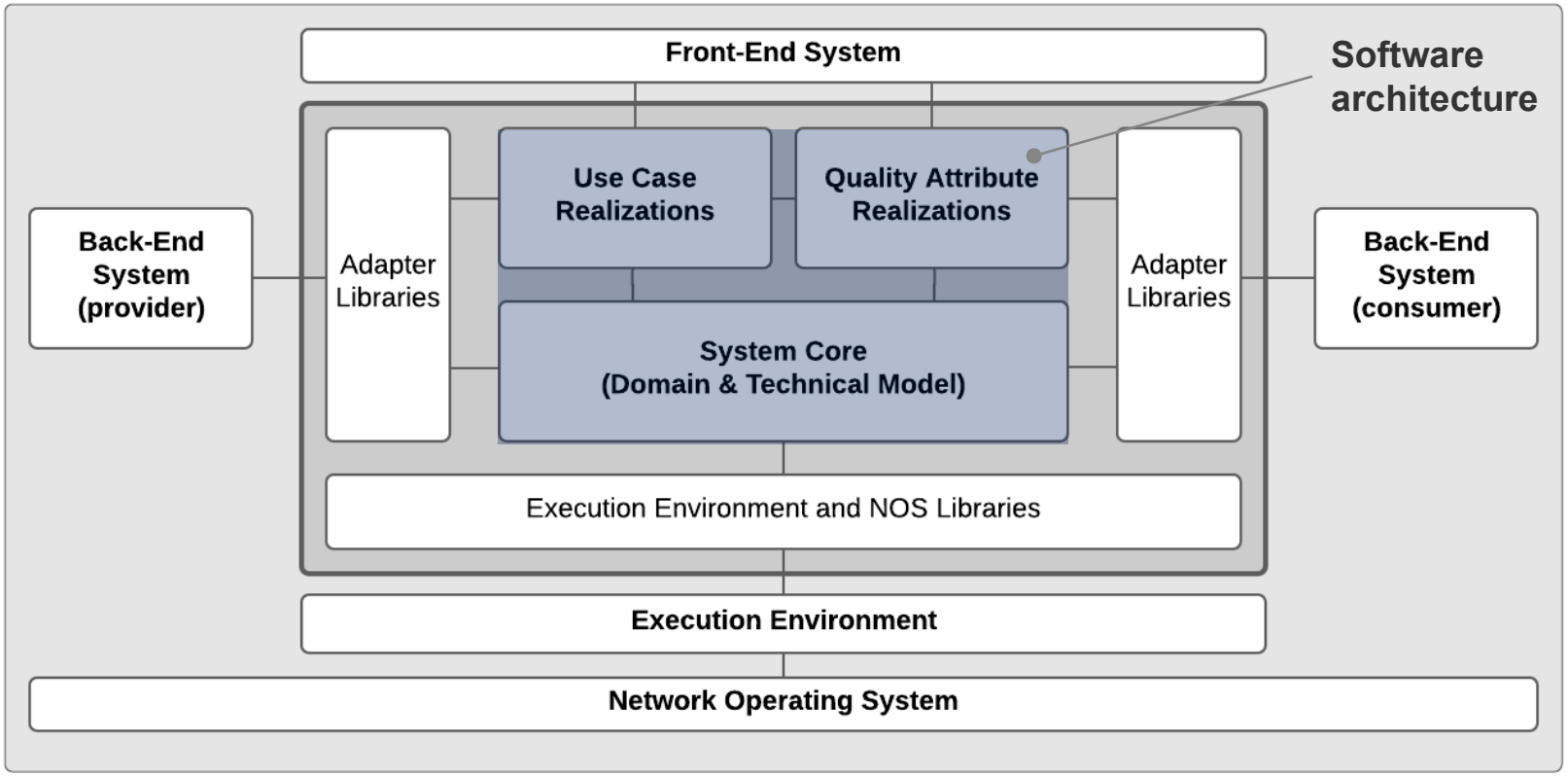
- **Use Case Realizations**. Components and component collaboration to realize required system functionality.
- **Quality Attribute Realizations**. Components and component collaboration to realize required system quality (e.g., security, performance, availability, modifiability).
- **System Core (Domain & 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).

Lecture Agenda

- Classical Architecture
- Enterprise Business and Operating Model
- System
- Architecture Disciplines
- System Architecture
- Software Architecture

Software Architecture
Architecture differentiation

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



Software Architecture
Software Architecture Definition

There are many different answers to the question of **what exactly software architecture is** — wickedly put, as many as there are software architects.

One world-renowned institution that has long been involved in software architecture is the **Software Engineering Institute (SEI)** at Carnegie Mellon University in Pittsburgh, Pennsylvania.

Various such **software architecture definitions** have been collected and made available to the general public ([SEI Software Architecture 2021])

Software Architecture
Software Architecture Definition

However, the SEI also offers its own definition[1] (text 1):

"The software architecture of a program or computing system is a depiction of the system that aids in the understanding of how the system will behave. Software architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by design and implementation teams. The architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. Architecture is an artifact for early analysis to make sure that a design approach will yield an acceptable system. By building effective architecture, you can identify design risks and mitigate them early in the development process."

[1] http://www.sei.cmu.edu/architecture/

Software Architecture
Software Architecture Definition

Does this definition contradict common agile process attitude and models? Architecture (as defined by the SEI) sounds like a lot of *planning ahead*. Some thoughts worth reading on the role of architecture and architects in the article *who needs an architect* ([Fowler 2003]) in which Martin Fowler embeds postings by Ralph Johnson (text 2):

"In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called *architecture*. This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers."

Software Architecture
Software Architecture Definition

What are the relationships between architecture and code? What significance does the architecture still have when a system has been completely programmed? In this context, the abstract of a presentation at the Java User Group Switzerland (JUGS) by Simon Brown (software architecture vs. code[1]) is worth reading (text 3):

"Software architecture and coding are often seen as mutually exclusive disciplines, despite us referring to higher level abstractions when we talk about our software. You've probably heard others on your team talking about components, services and layers rather than objects when they're having discussions. Take a look at the codebase though. Can you clearly see these abstractions or does the code reflect some other structure? If so, why is there no clear mapping between the architecture and the code? Why do those architecture diagrams that you have on the wall say one thing whereas your code says another?"

[1] https://www.jug.ch/html/events/2015/software_architecture_vs_code.html

Software Architecture
Software Architecture Definition

Finally, another interesting angle via which Ralph Johnson marks an elementary difference between traditional architecture or engineering disciplines and software architecture in Martin Fowler's article ([Fowler 2003]) (text 4):

"Software is not limited by physics, like buildings are. It is limited by imagination, by design, by organization. In short, it is limited by properties of people, not by properties of the world. We have met the enemy, and he is us."

An angle Melvin Conway had already picked up in *how do committees invent?* ([Conway 1968]) (text 5):

"The basic thesis of this article is that organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations."

Questions

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

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

[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

[Gehry 2024]
Frank Gehry (Architect), https://de.wikipedia.org/wiki/Frank_Gehry

Bibliography
Lecture

[La Roche 2024]
Emanuel La Roche, https://de.wikipedia.org/wiki/Emanuel_La_Roche

[Luhmann 2024]
Social Systems Theory, https://en.wikipedia.org/wiki/Niklas_Luhmann

[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

[Perroud 2013]
Perroud, Thierry; Inversini, Reto, Enterprise Architecture Patterns, Springer-Verlag, Berlin Heidelberg, 2013

[Systems Theory 2024]
Systems Theory, https://en.wikipedia.org/wiki/Systems_theory

[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

[Weinberg 2001]
Weinberg, Gerald M. 2001. An Introduction to General Systems Thinking. New York: Dorset House.