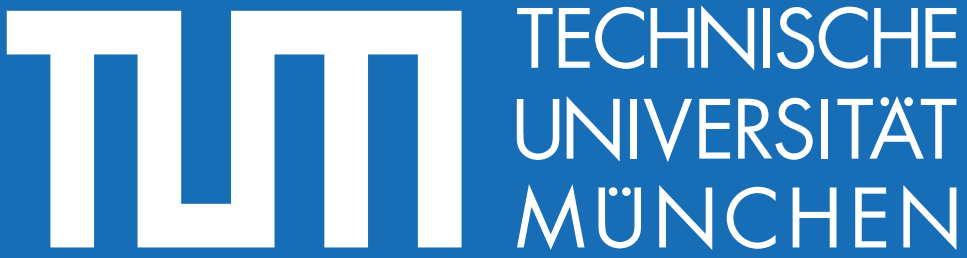


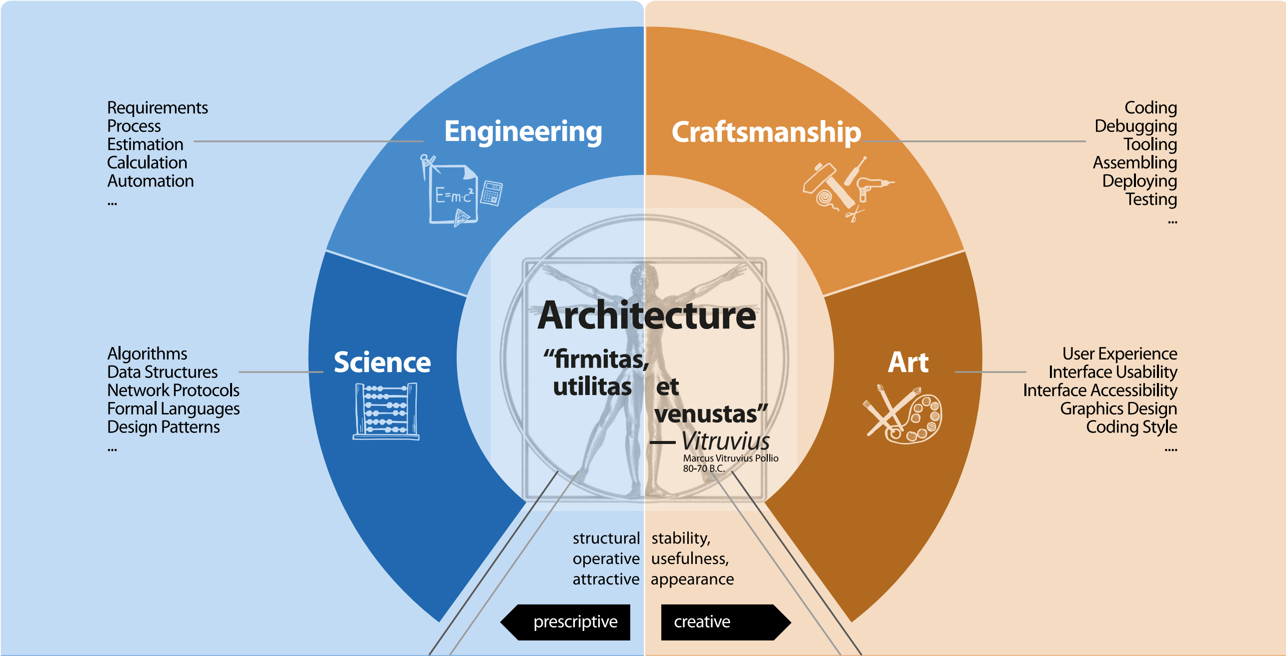
Software Engineering in Industrial Practice (SEIP)

Dr. Ralf S. Engelschall

ARCHITECTURE FUNDAMENTALS

Architecture Stargate

TECHNISCHE UNIVERSITÄT MÜNCHEN



Structural Scientific Definition of Architecture:
Architecture of a system is the set of fundamental **concepts** and **properties** of the system in its environment, embodied in its **elements**[, **interfaces**], **relationships**[and **behaviours**] and the principles of its **design** and **evolution**.
(based on industry standard *ISO/IEC 42010*)

Holistic Artistic Definition of Architecture:
Great architecture is achieved **harmony** and **accord** of all **parts**, where you no longer can add, modify or remove anything without impairing the **whole**.
(based on a quote by *Leon Battista Alberti* on beauty)

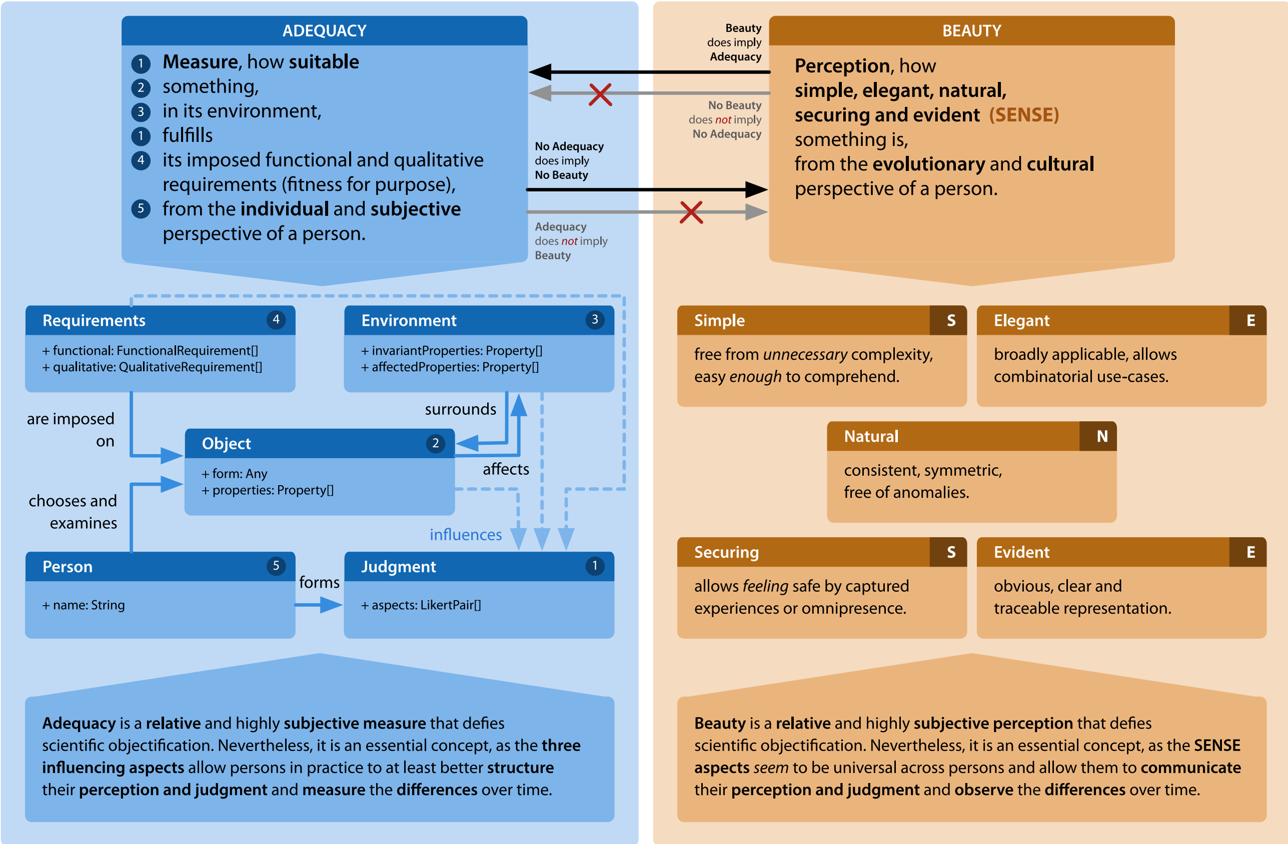
Architecture is not easy to define. You can define architecture both **structurally scientific** through measurable elements, interfaces and relationships, or also **wholely artistic** through "the harmony and the accord of all parts." The "truth" lies somewhere in practice in between, because the two extremes span a broad space, in which all solutions are located in practice.

On the structurally scientific side, architecture defines itself through the aspects **Science** (in particular Computer Science) and **Engineering** (especially Software Engineering). On the holistic artistic side, architecture defines itself through the aspects of **Craftsmanship** (especially programming) and **Art** (especially User Experience).

Questions

- How to define **Architecture**?
- What are the four **Aspects** of **Architecture**?

Adequacy and Beauty



Adequacy is defined as the measure, how suitable something, in its environment, fulfills its imposed functional and qualitative requirements (fitness for purpose), from the individual perspective of a person.

Adequacy is a relative and highly subjective measure that defies scientific objectification. Nevertheless, adequacy is an essential concept, as the three influencing aspects (Requirements, Environment, Object) allow persons in practice to at least better structure their perception and judgment and measure the differences over time.

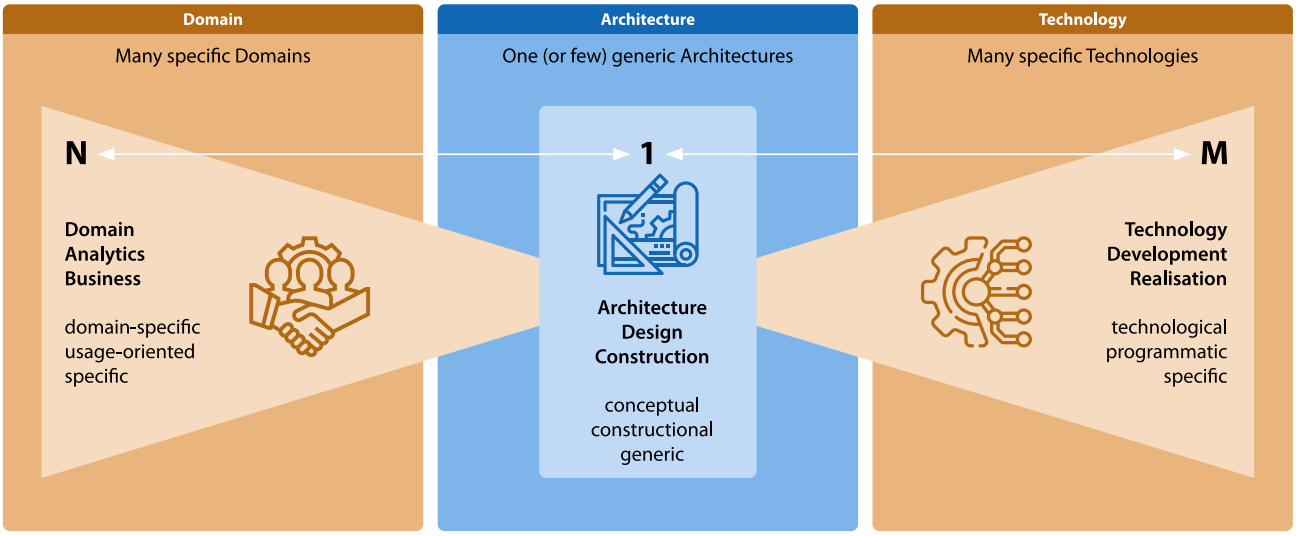
Beauty is defined as the perception, how simple, elegant, natural, securing and evident (SENSE) something is, from the evolutionary perspective of a person.

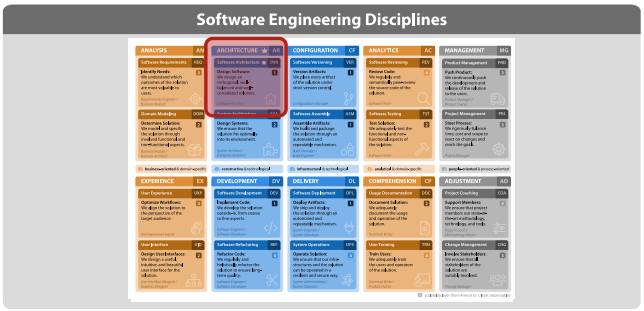
Beauty is an absolute and highly subjective perception that defies scientific objectification. Nevertheless, beauty is an essential concept, as the SENSE aspects seem to be universal across persons and allow them to communicate their perception and judgment and observe the differences over time.

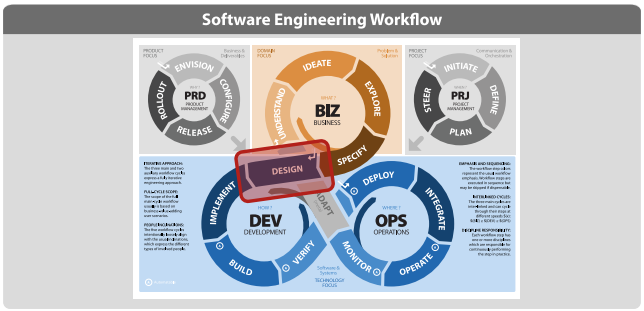
Questions

- Is it possible to measure Adequacy or Beauty in general?
- Is it possible to measure Adequacy or Beauty in the context of a single person?

King Discipline Architecture







(Software) **Architecture** is considered the "King Discipline" in **Software Engineering**, since it is the central, general, and conceptual link between the many, potential, specific, realized **Domains** and the many, potential, specific, realizing **Technologies**. The architectural construction of an application takes place in the logical step "Design" within the BizDevOps-workflow of Software Engineering.

Questions

- Why is **Architecture** considered the "King Discipline" of Software Engineering?

Architecture Manifesto

The **Manifesto for IT Architecture** is a policy statement for IT architecture. First and foremost, it says to "Continuously Raising the Bar", since after just 50 years of Software Engineering and Software Architecture even though we already know a number of best practices, the discipline will certainly have to continue to develop for a very long time.

The Mission for IT architects is the design, implementation, and maintenance of IT solutions. The Entitlement is to continuously raise the bar and help others to learn the "craft." Naturally is the fact that through the work of architects, the maximum added value is achieved for customers.

The basic values, which play a central role in this craft and which are greatly appreciated are: **Latest Technologies**, **Theoretical Consideration**, **Analytical Engineering**, **Achieved Industrialization**, **Reactive Correction**, **Tested Robustness** and **Useful Functionality**.

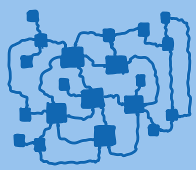
In addition, there are additional values, which also play a central role and are even more appreciated: **Sustainable Concepts** (the content of **Architecture Fundamentals**!), **Pragmatic Making**, **Constructive Craftsmanship**, **Accredited Creativity**, **Proactive Improvement**, **Inherent Quality** and **Operational Delight**.

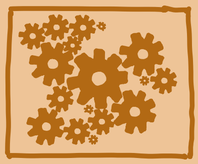
Questions

(none)

Complex vs. Complicated

	complex	complicated
FOCUS	refers to the **extrinsic** and **higher**- or **macro**-level difficulty of a system,	refers to the **intrinsic** and **lower**- or **micro**-level difficulty of a system,
RATIONALE	because the system involves many different and **connected** parts	because the system involves many different and **difficult** aspects
CHALLENGE	which take time to **comprehend** and **master** in **total**,	which take time to **understand** and **learn** in **detail**,
INSIGHT	and which nevertheless are **easy** to explain.	and which usually are **hard** to explain.





	NOTICE	NOTICE
	Simple (non-complicated) systems can be **complex**.	**Clear** (non-complex) systems can be **complicated**.

	RECOGNIZE	RECOGNIZE
	Architecture primarily has to master the **complex** aspects of a system.	**Development** primarily has to master the **complicated** aspects of a system.

"Complex" refers to the extrinsic and higher- or macro-level difficulty of a system because the system involves many different and connected parts which take time to comprehend and master in total, and which nevertheless are easy to explain.

"Complicated" refers to the intrinsic and lower- or micro-level difficulty of a system because the system involves many different and difficult aspects which take time to understand and learn in detail, and which usually are hard to explain.

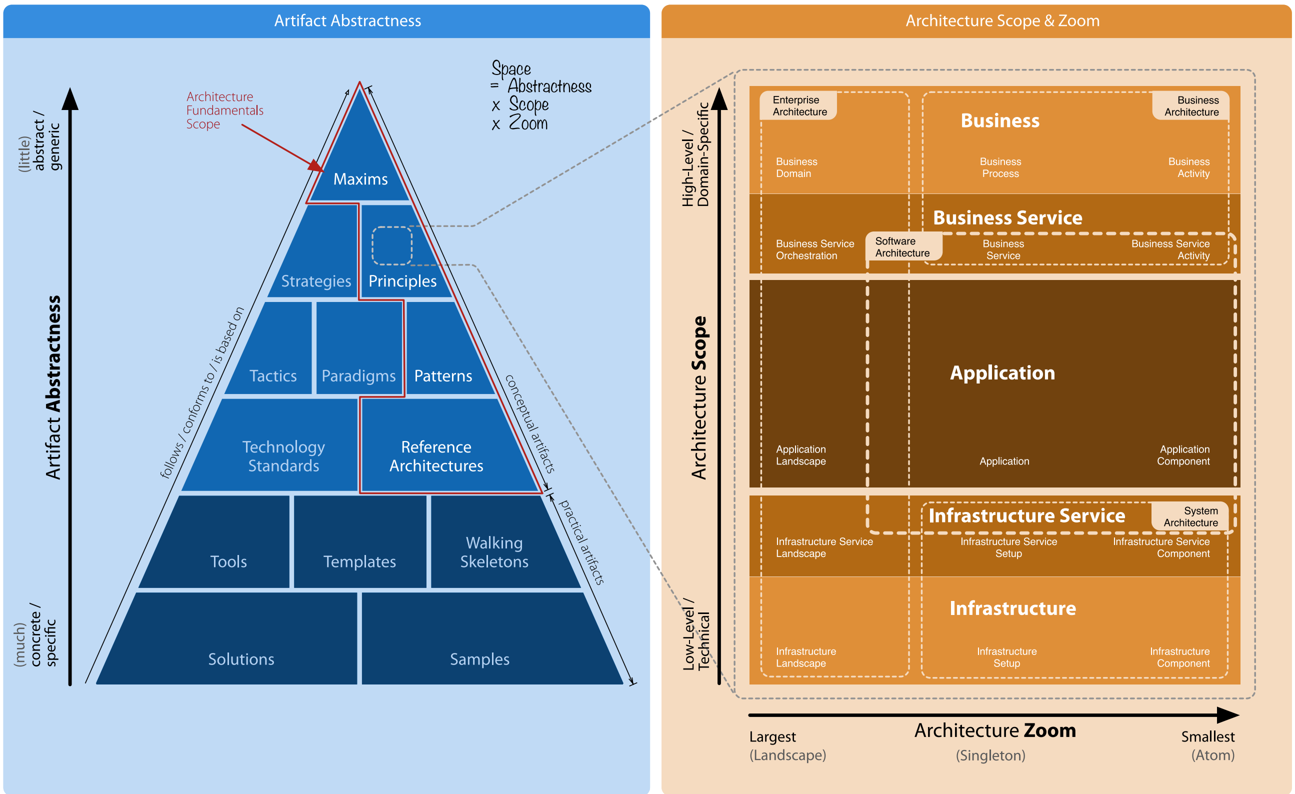
Note: Simple (non-complicated) systems can be complex – clear (non-complex) systems can be complicated.

The crucial difference is: The architecture or the construction has to master the complex aspects of a system. The development or realization has to master the complicated aspects of a system.

Questions

- Does **Architecture** primarily have to deal with **complex** or **complicated** aspects of a system?

Architecture Space



The **IT Architecture Space** consists of three dimensions: the **Artifact Abstractness** (a degree of abstraction of all artifacts that are known by the architect), the **Architecture Scope** (the field of architecture, in which one acts) and the **Architecture Zoom** (the detail level of architecture in which one acts).

One primarily distinguishes between three Architecture Scopes: (high-level/domain-specific) **Business**, **Application** and (low-level/technical) **Infrastructure**. In addition, the two secondary Architecture Scopes **Business Service** and **Infrastructure service** are used to reduce in practice the "mental leap" from Business to Application and from Application to Infrastructure.

On each Architecture Scope, one can choose from at least three different **Architecture Zooms**: from **Landscape** (largest), over **Singleton** to **Atom** (smallest).

In the space of Architecture Scope and Zoom, one differentiates between four types of **IT architecture**: **Business Architecture**, **Software Architecture**, **System Architecture** and **Enterprise Architecture**.

The higher/lower the level of abstraction of the artifacts, the larger/smaller usually is the coverage of the room made from Architecture Scope and Architecture Zoom.

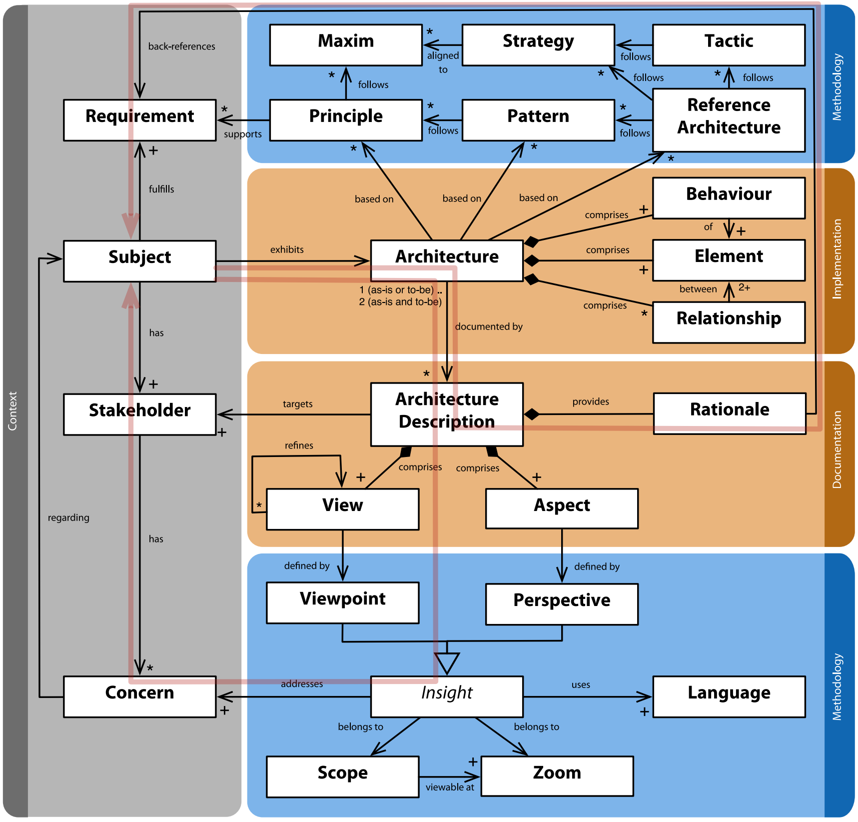
Questions

- The IT **Architecture Space** consists of which three dimensions?
- Which four types of Architecture are known in IT?

ARCHITECTURE FUNDAMENTALS

Architecture Ontology

TECHNISCHE UNIVERSITÄT MÜNCHEN



Requirement: A functional or non-functional demand or imposed obligation on the **Subject**.

Architecture: Inherent static and dynamic structure of a **Subject** which comprise **Elements**, the visible **Behaviour** of **Elements** and **Relationships** between **Elements**.

Maxim: Fundamental, generally valid set of values and rules to guide the architecture discipline (think: law).

Subject: Any type of business process, business service, software application, infrastructure service or infrastructure setup.

Element: Fundamental part from which a **Subject** can be considered to be constructed.

Strategy: Long-term situation-independent plan of approaches to achieve a particular goal (think: war). Aligned and not in conflict with any **Maxims**.

Stakeholder: Person, group or entity with an interest in or **Concerns** about the **Architecture**.

Behaviour: Run-time characteristic of **Architecture Elements** of the **Subject**.

Tactic: Short-term situation-dependent plan of actions to achieve a particular goal (think: battle). Following and supporting a **Strategy**.

Concern: Requirement, objective, intention, or aspiration a **Stakeholder** has on an **Architecture**.

Relationship: Static or dynamic relationship between **Elements** of an **Architecture** of the **Subject**.

Principle: Fundamental truth, rule, tenet or policy an **Architecture** follows.

Architecture Description: Set of artifacts that document an **Architecture** in a way **Stakeholders** can understand and ensures their **Concerns** are met.

Rationale: Fundamental reasons for a particular chosen **Architecture**, usually strongly based on non-functional **Requirements**.

Pattern: Proven recurring theme, structure, approach or behavior an **Architecture** and its **Elements** can follow.

Viewpoint: Collection of templates and conventions for constructing one type of **View**. It addresses **Concerns** and contains guidelines for constructing a View.

View: Representation of one or more structural aspects of an **Architecture** that illustrates how the **Architecture** addresses one or more **Concerns**.

Reference Architecture: Reusable proven Architecture template, **based** on a set of Patterns and following one or more **Tactics** and **Strategies**.

Perspective: Collection of decisions, guidelines and rules that ensure a **Subject** exhibits a set of non-functional **Requirements**, considered across a number of **Views**.

Aspect: Representation of one or more non-functional aspects of an **Architecture** that illustrate how the **Architecture** addresses one or more **Concerns**.

Scope: Primary area of the **Architecture** space an **Insight** addresses: either Business, Business Service, Application, Infrastructure Service or Infrastructure.

Insight: Superordinate abstract concept to address particular **Concerns** through **Viewpoints** and **Perspectives**.

Language: Formal, semi-formal or even prose language to describe an Insight, so it optimally addresses the **Concerns** of **Stakeholders**.

Zoom: Level of view distance to the **Architecture** an **Insight** has. Also known as the **Insight** detail level, ranging from smallest atoms, over singletons to the largest landscape.

So that Architects can communicate meaningfully in practice, one must agree on a few basic terms and their meaning. The terms are defined in a taxonomy and are described in the **Architecture Ontology** in relation to each other.

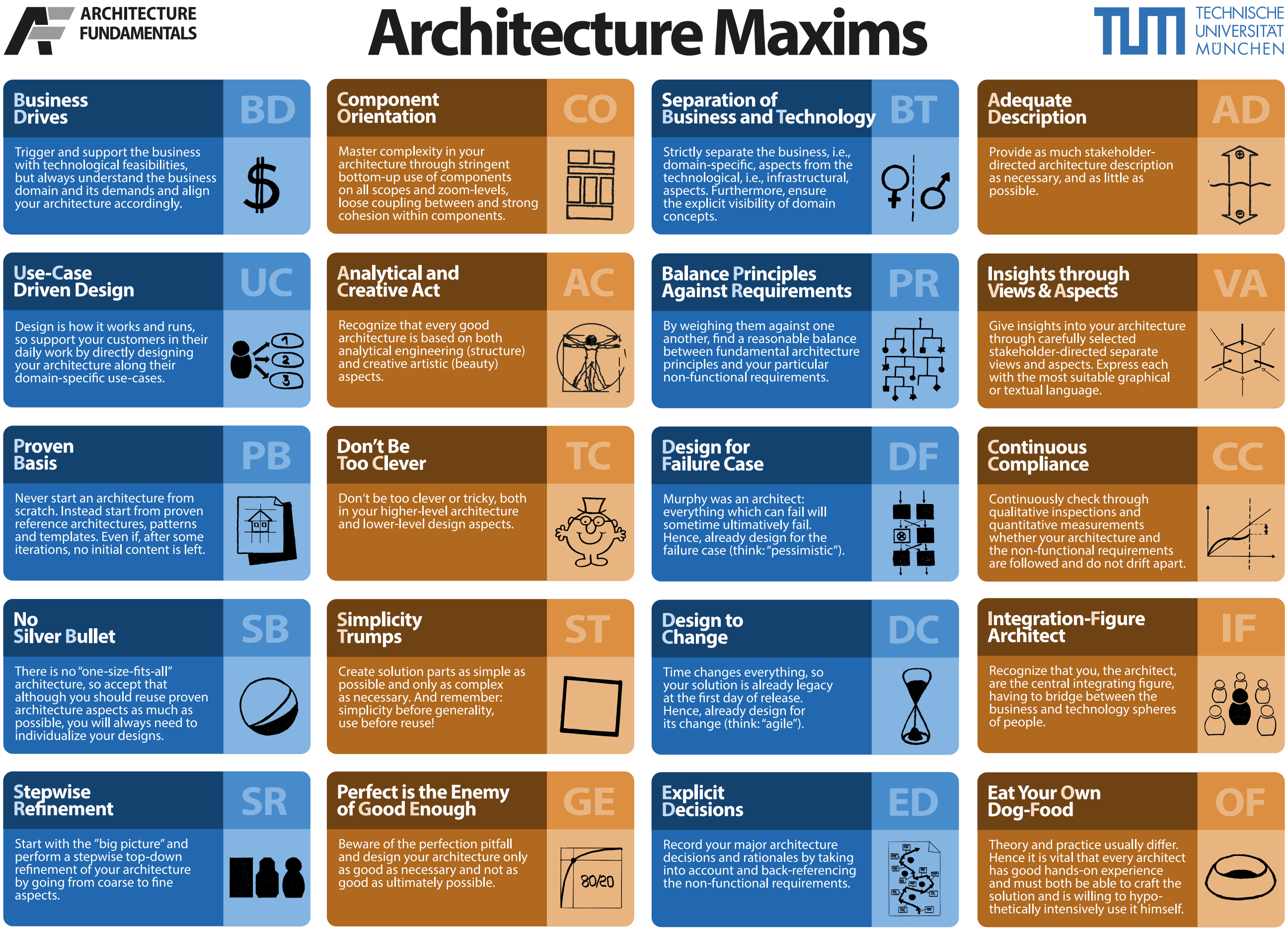
In the Architecture Ontology, there are two main important "loops." Both start at the **Subject**, which has an **Architecture**, which is documented via the **Architecture Description**.

Loop 1: The Architecture Description gives **Rationales** for decisions, which, ideally, should be back-referencing to **Requirements**. Because an Architecture Description should not document the **WHAT** but the **WHY**. Because the WHAT can also be seen in the code, but the WHY not!

Loop 2: The Architecture Description consists of **Views** and **Aspects**, which methodically are called **Viewpoints** and **Perspectives**. Both together provide **Insights** at **Scope** and **Zoom Level** (see Architecture Space!) and are documented via a specific (graphical or textual) **Language**. In any case, only those insights will be given which address a **Concern** of a **Stakeholder**. One also doesn't program anything, which one doesn't need!

Questions

- What should an **Architecture Description** document beside the **WHAT**?
- What should an **Architecture Description** especially address through **Insights**?


ARCHITECTURE FUNDAMENTALS
Architecture Maxims
TECHNISCHE UNIVERSITÄT MÜNCHEN
Business Drives BD
Trigger and support the business with technological feasibilities, but always understand the business domain and its demands and align your architecture accordingly.
Component Orientation CO
Master complexity in your architecture through stringent bottom-up use of components on all scopes and zoom-levels, loose coupling between and strong cohesion within components.
Separation of Business and Technology BT
Strictly separate the business, i.e., domain-specific, aspects from the technological, i.e., infrastructural, aspects. Furthermore, ensure the explicit visibility of domain concepts.
Adequate Description AD
Provide as much stakeholder-directed architecture description as necessary, and as little as possible.
Use-Case Driven Design UC
Design is how it works and runs, so support your customers in their daily work by directly designing your architecture along their domain-specific use-cases.
Analytical and Creative Act AC
Recognize that every good architecture is based on both analytical engineering (structure) and creative artistic (beauty) aspects.
Balance Principles Against Requirements PR
By weighing them against one another, find a reasonable balance between fundamental architecture principles and your particular non-functional requirements.
Insights through Views & Aspects VA
Give insights into your architecture through carefully selected stakeholder-directed separate views and aspects. Express each with the most suitable graphical or textual language.
Proven Basis PB
Never start an architecture from scratch. Instead start from proven reference architectures, patterns and templates. Even if, after some iterations, no initial content is left.
Don't Be Too Clever TC
Don't be too clever or tricky, both in your higher-level architecture and lower-level design aspects.
Design for Failure Case DF
Murphy was an architect: everything which can fail will sometime ultimatively fail. Hence, already design for the failure case (think: "pessimistic").
Continuous Compliance CC
Continuously check through qualitative inspections and quantitative measurements whether your architecture and the non-functional requirements are followed and do not drift apart.
No Silver Bullet SB
There is no "one-size-fits-all" architecture, so accept that although you should reuse proven architecture aspects as much as possible, you will always need to individualize your designs.
Simplicity Trumps ST
Create solution parts as simple as possible and only as complex as necessary. And remember: simplicity before generality, use before reuse!
Design to Change DC
Time changes everything, so your solution is already legacy at the first day of release. Hence, already design for its change (think: "agile").
Integration-Figure Architect IF
Recognize that you, the architect, are the central integrating figure, having to bridge between the business and technology spheres of people.
Stepwise Refinement SR
Start with the "big picture" and perform a stepwise top-down refinement of your architecture by going from coarse to fine aspects.
Perfect is the Enemy of Good Enough GE
Beware of the perfection pitfall and design your architecture only as good as necessary and not as good as ultimately possible.
80/20
Explicit Decisions ED
Record your major architecture decisions and rationales by taking into account and back-referencing the non-functional requirements.
Eat Your Own Dog-Food OF
Theory and practice usually differ. Hence it is vital that every architect has good hands-on experience and must both be able to craft the solution and is willing to hypothetically intensively use it himself.

In IT Architecture, one follows **Architecture Maxims**, which are basic guidelines. One knows 20 maxims. The architect should always follow the maxims and never break them.

Note: **Proven Basis** and **No Silver Bullet** say that one has to start an architecture always on a proven basis (e.g., a reference architecture), however, at the same time, it has to be clear that one cannot use these 1:1, but always first have to adjust it.

Note: **Stepwise Refinement** and **Component Orientation** say that regarding time (and for reasons of risk minimization), one always goes from the coarse to the fine, while the results show a stringent component-orientation, where small components are hierarchically integrated into larger components.

Note: while, as an IT architect, one just has to accept all maxims, **Simplicity Trumps** is from another quality: nothing in IT is really easy. When something looks simple, one just doesn't understand enough about it. Or someone really invested a lot just to make it look simple. **Simplicity Trumps** means precisely this: make inherently complex things simple again.

Questions

? For reasons of risk minimization, how should the IT architect at **Stepwise Refinement** always proceed step-by-step?