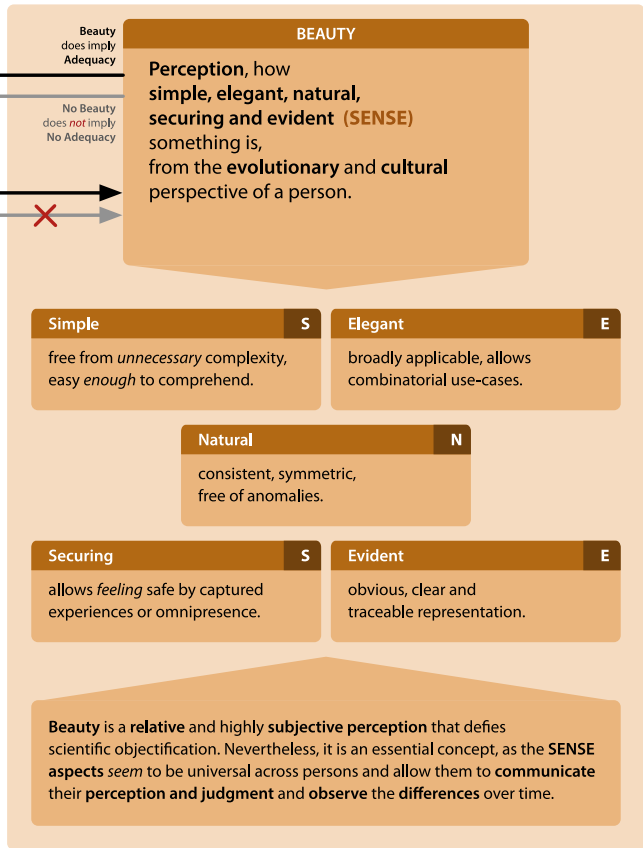
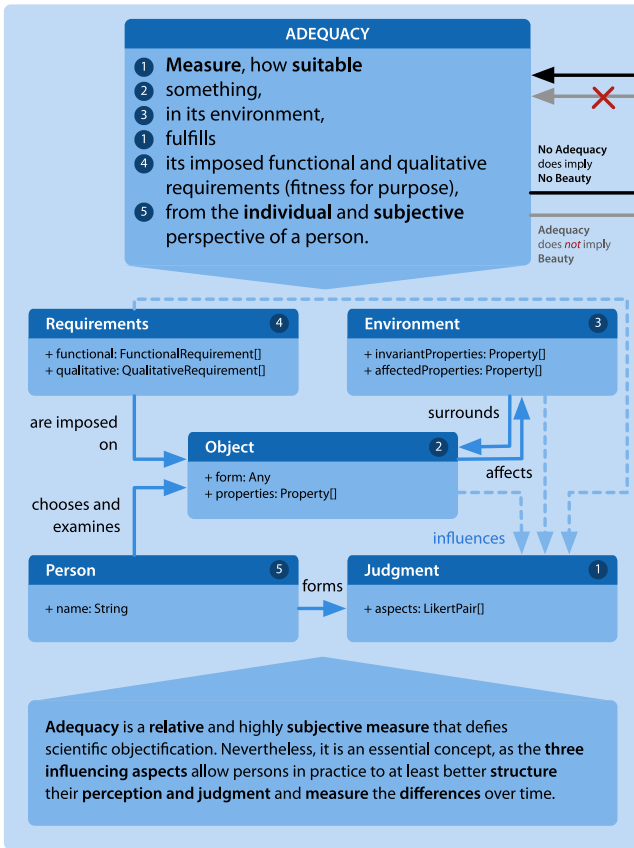




**Software Engineering  
in der industriellen Praxis  
(SEIP)**

Dr. Ralf S. Engelschall





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.

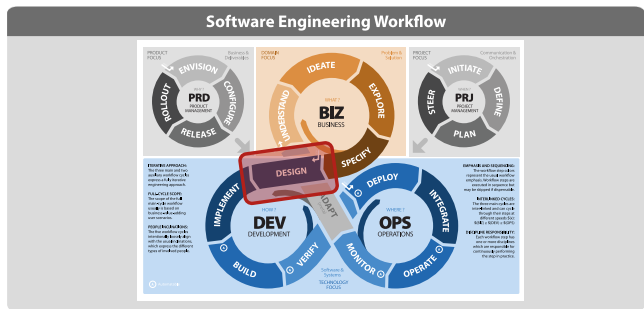
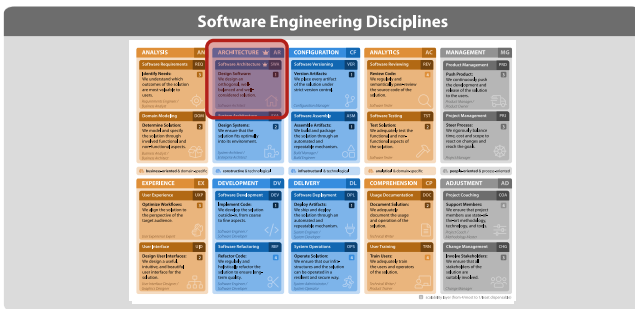
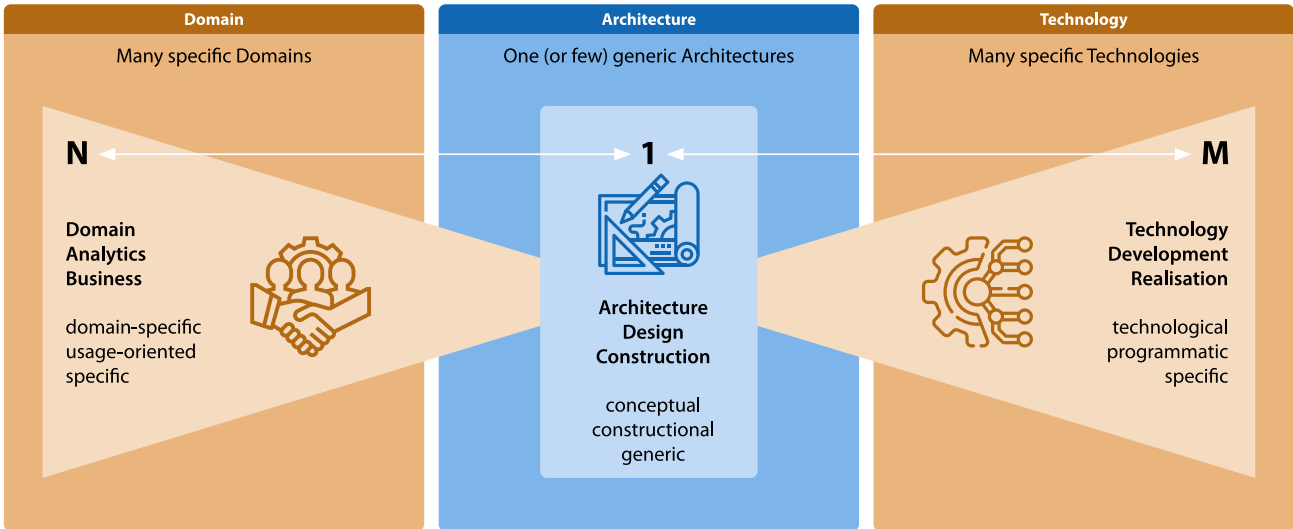
Adequacy is a relative and highly subjective measure that defies scientific objectification. Nevertheless, adequacy is an essential concept, as the three influencing aspects (Requirements, Environment, Object) allow persons in practice to at least better structure their perception and judgment and measure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



(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?

# Manifesto for IT Architecture

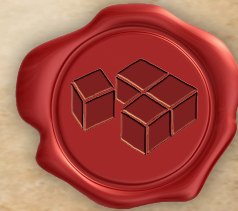
## Continuously Raising the Bar

**Mission** As IT Architects we guide the design, implementation and evolution of IT solutions.

**Entitlement** We continuously strive to raise the bar of professional IT architecture by practicing it and helping others to learn our craft. We achieve maximum value for our clients through our work.

**Values** Through this work we have come to value aspects of our craft. While we acknowledge the beneficial values in the items on the right, we appreciate the stronger values in the items on the left even more.

<b>Sustainable Concepts</b>	over <b>Latest Technologies</b>
<b>Pragmatic Making</b>	over <b>Theoretical Consideration</b>
<b>Constructive Craftsmanship</b>	over <b>Analytical Engineering</b>
<b>Accredited Creativity</b>	over <b>Achieved Industrialization</b>
<b>Proactive Improvement</b>	over <b>Reactive Correction</b>
<b>Inherent Quality</b>	over <b>Tested Robustness</b>
<b>Operational Delight</b>	over <b>Useful Functionality</b>



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.

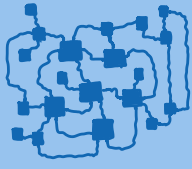
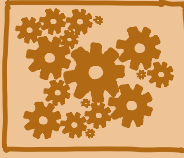
The Mission for IT architects is the design, implementation, and maintenance of IT solutions. The Entitlement is to continuously raise the bar and help others to learn the “craft.” Naturally is the fact that through the work of architects, the maximum added value is achieved for customers.

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	complicated
FOCUS	refers to the <b>extrinsic</b> and <b>higher-</b> or <b>macro-</b> level difficulty of a system,	refers to the <b>intrinsic</b> and <b>lower-</b> or <b>micro-</b> level difficulty of a system,
RATIONALE	because the system involves many different and <b>connected</b> parts	because the system involves many different and <b>difficult</b> aspects
CHALLENGE	which take time to <b>comprehend</b> and <b>master in total</b> ,	which take time to <b>understand</b> and <b>learn in detail</b> ,
INSIGHT	and which nevertheless are <b>easy</b> to explain.	and which usually are <b>hard</b> to explain.
		
	<b>NOTICE</b> <b>Simple</b> (non-complicated) systems can be <b>complex</b> .	<b>NOTICE</b> <b>Clear</b> (non-complex) systems can be <b>complicated</b> .
	<b>RECOGNIZE</b> <b>Architecture</b> primarily has to master the <b>complex</b> aspects of a system.	<b>RECOGNIZE</b> <b>Development</b> primarily has to master the <b>complicated</b> 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.

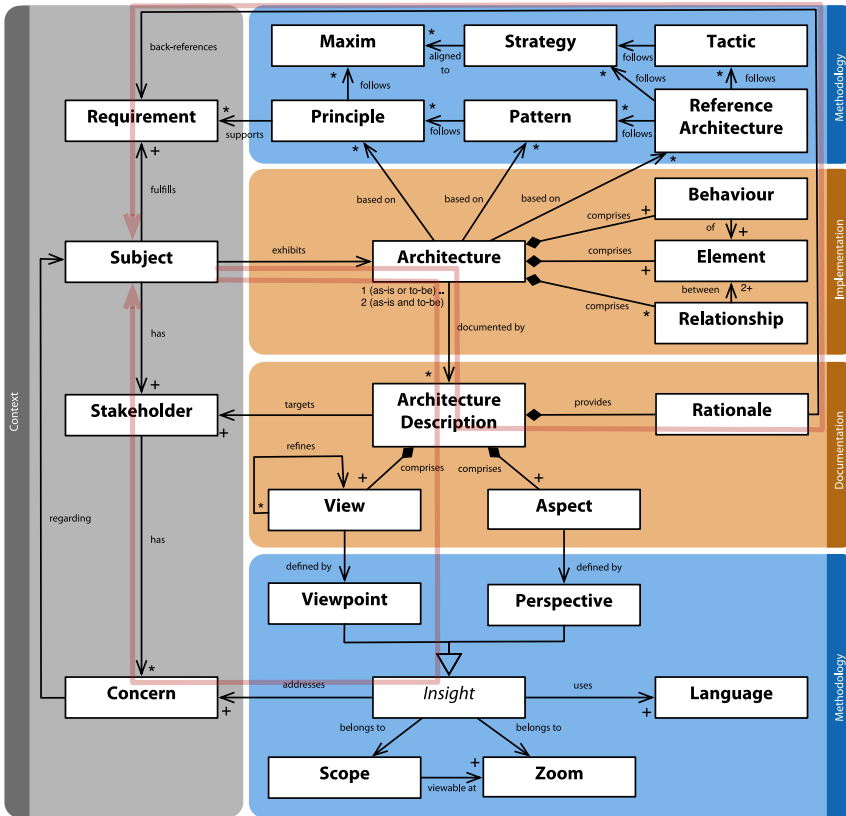
The crucial difference is: The architecture or the construction has to master the complex aspects of a system. The development or realization has to master the complicated aspects of a system.

## Questions

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







<b>Requirement:</b> A functional or non-functional demand or imposed obligation on the <b>Subject</b> .	<b>Architecture:</b> Inherent static and dynamic structure of a <b>Subject</b> which comprise <b>Elements</b> , the visible <b>Behaviour</b> of <b>Elements</b> and <b>Relationships</b> between <b>Elements</b> .	<b>Maxim:</b> Fundamental, generally valid set of values and rules to guide the architecture discipline (think: law).
<b>Subject:</b> Any type of business process, business service, software application, infrastructure service or infrastructure setup.	<b>Element:</b> Fundamental part from which a <b>Subject</b> can be constructed.	<b>Strategy:</b> Long-term situation-independent plan of approaches to achieve a particular goal (think: war). Aligned and not in conflict with any <b>Maxims</b> .
<b>Stakeholder:</b> Person, group or entity with an interest in or <b>Concerns</b> about the <b>Architecture</b> .	<b>Behaviour:</b> Run-time characteristic of <b>Architecture Elements</b> of the <b>Subject</b> .	<b>Tactic:</b> Short-term situation-dependent plan of actions to achieve a particular goal (think: battle). Following and supporting a <b>Strategy</b> .
<b>Concern:</b> Requirement, objective, intention, or aspiration a <b>Stakeholder</b> has on an <b>Architecture</b> .	<b>Relationship:</b> Requirement, objective, intention, or aspiration a <b>Stakeholder</b> has on an <b>Architecture</b> .	<b>Principle:</b> Fundamental truth, rule, tenet or policy an <b>Architecture</b> follows.
<b>Architecture Description:</b> Set of artifacts that document an <b>Architecture</b> in a way <b>Stakeholders</b> can understand and ensures their <b>Concerns</b> are met.	<b>Rationale:</b> Fundamental reasons for a particular chosen <b>Architecture</b> , usually strongly based on non-functional <b>Requirements</b> .	<b>Pattern:</b> Proven recurring theme, structure, approach or behavior an <b>Architecture</b> and its <b>Elements</b> can follow.
<b>Viewpoint:</b> Collection of templates and conventions for constructing one type of <b>View</b> . It addresses <b>Concerns</b> and contains guidelines for constructing a <b>View</b> .	<b>View:</b> Representation of one or more structural aspects of an <b>Architecture</b> that illustrates how the <b>Architecture</b> addresses one or more <b>Concerns</b> .	<b>Reference Architecture:</b> Reusable proven <b>Architecture</b> template, based on a set of <b>Patterns</b> and following one or more <b>Tactics</b> and <b>Strategies</b> .
<b>Perspective:</b> Collection of decisions, guidelines and rules that ensure a <b>Subject</b> exhibits a set of non-functional <b>Requirements</b> , considered across a number of <b>Views</b> .	<b>Aspect:</b> Representation of one or more non-functional aspects of an <b>Architecture</b> that illustrate how the <b>Architecture</b> addresses one or more <b>Concerns</b> .	<b>Scope:</b> Primary area of the <b>Architecture</b> space an <b>Insight</b> addresses; either Business, Business Service, Application, Infrastructure Service or Infrastructure.
<b>Insight:</b> Superordinate abstract concept to address particular <b>Concerns</b> through <b>Viewpoints</b> and <b>Perspectives</b> .	<b>Language:</b> Formal, semi-formal or even prose language to describe an <b>Insight</b> , so it optimally addresses the <b>Concerns</b> of <b>Stakeholders</b> .	<b>Zoom:</b> Level of view distance to the <b>Architecture</b> an <b>Insight</b> has. Also known as the <b>Insight</b> 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**?



