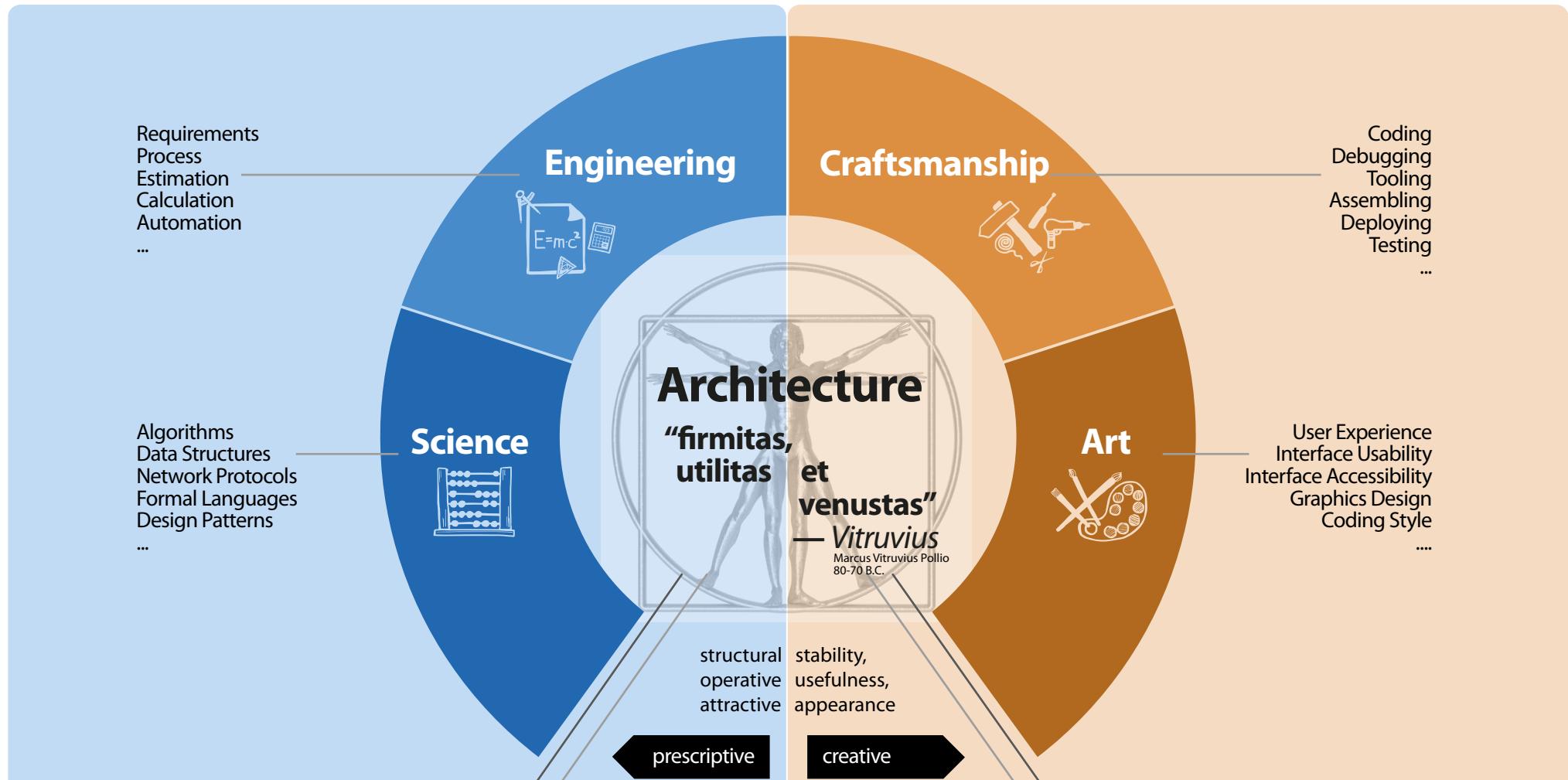




TECHNISCHE  
UNIVERSITÄT  
MÜNCHEN

# Software Engineering in der industriellen Praxis (SEIP)

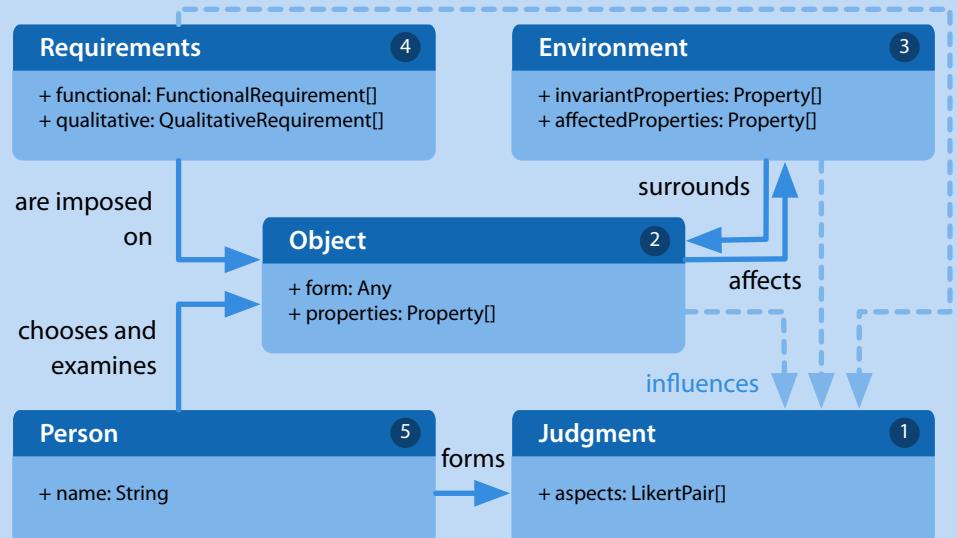
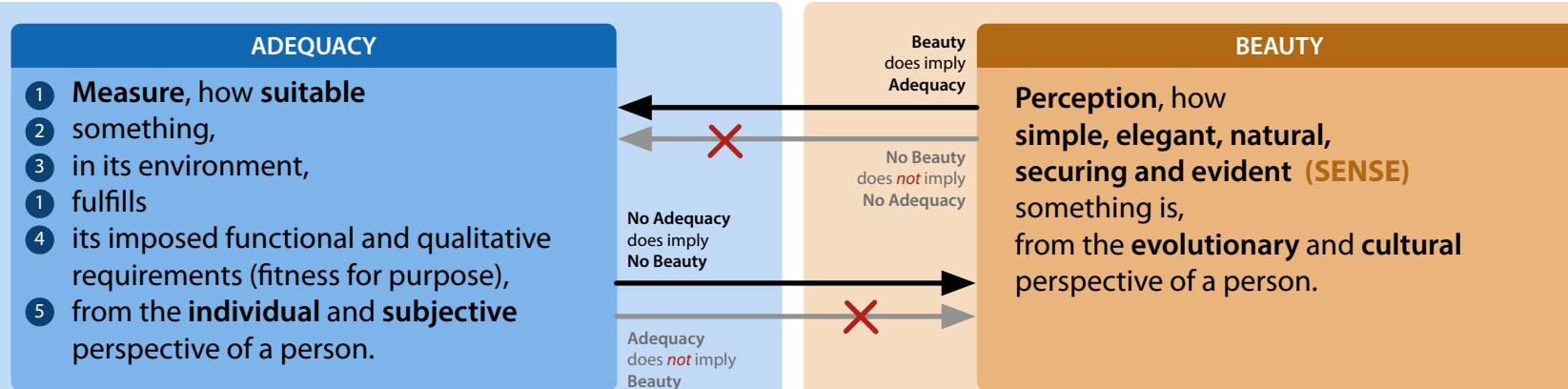
Dr. Ralf S. Engelschall



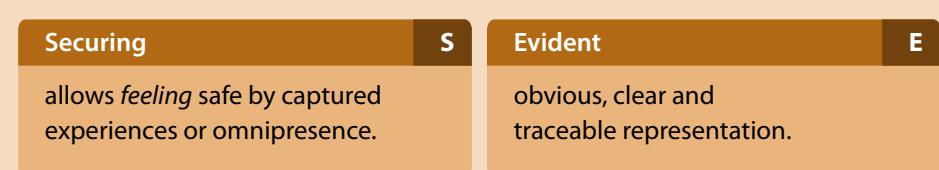
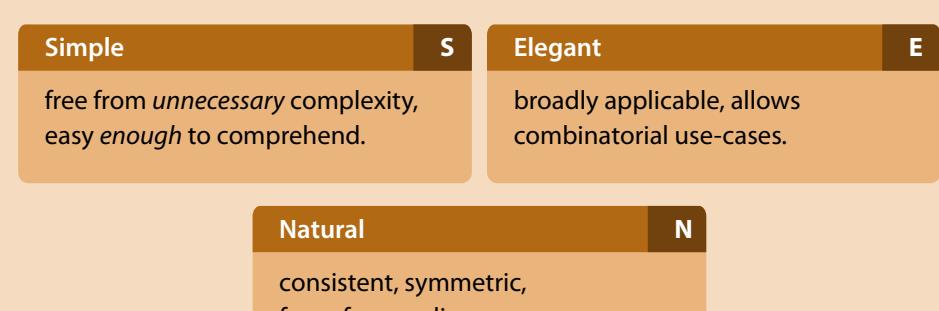
**Structural Scientific Definition of Architecture:**  
 Architecture of a system is the set of fundamental **concepts** and **properties** of the system in its environment, embodied in its **elements**[, **interfaces**], **relationships**[ and **behaviours**] and the principles of its **design** and **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)

# Adequacy and Beauty

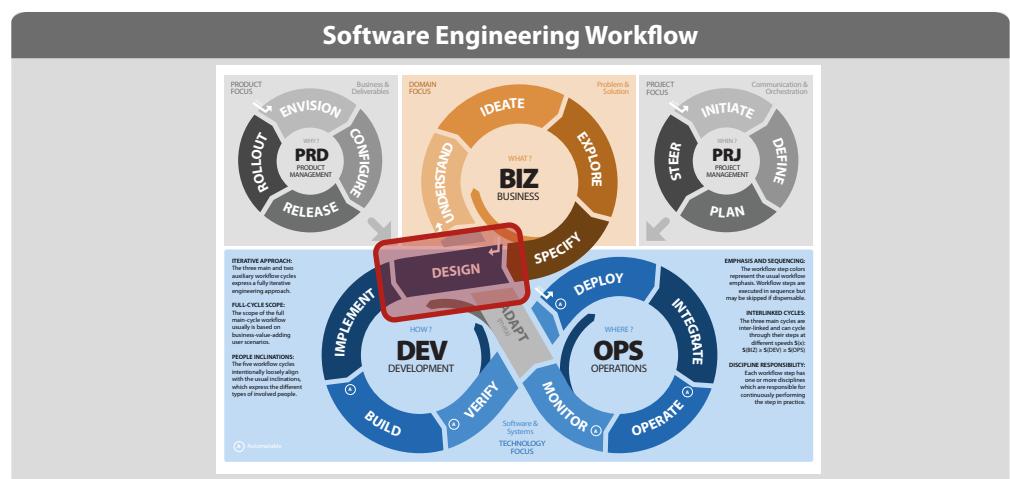
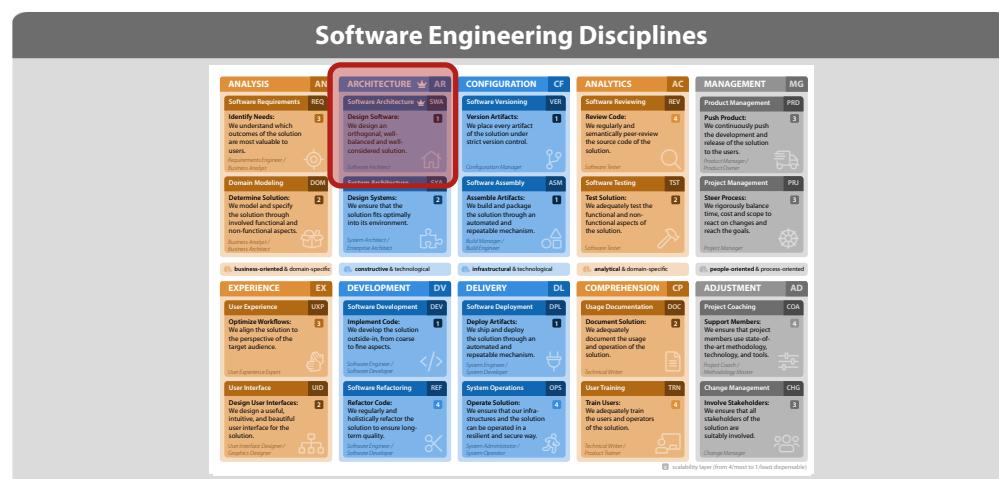
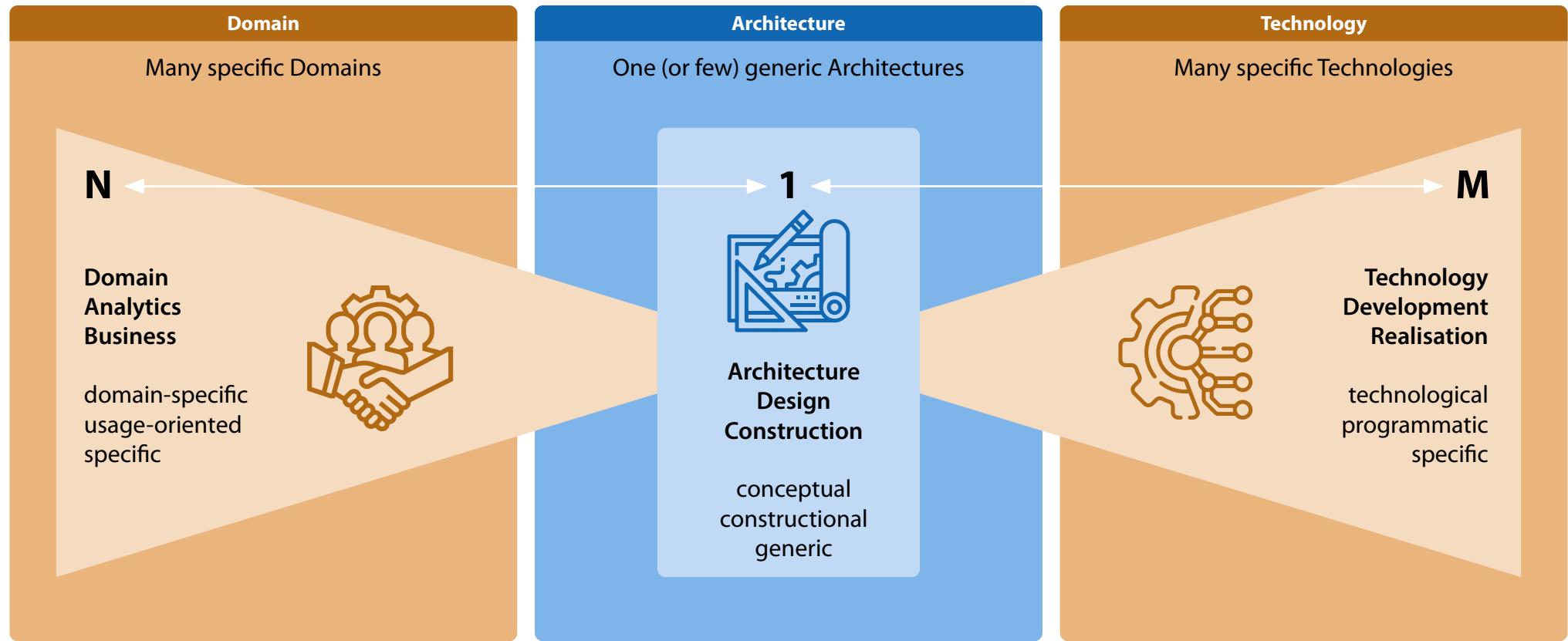


Adequacy is a relative and highly subjective measure that defies scientific objectification. Nevertheless, it is an essential concept, as the **three influencing aspects** allow persons in practice to at least better **structure** their **perception and judgment** and **measure the differences** over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



Beauty is a relative and highly subjective perception that defies scientific objectification. Nevertheless, it 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.

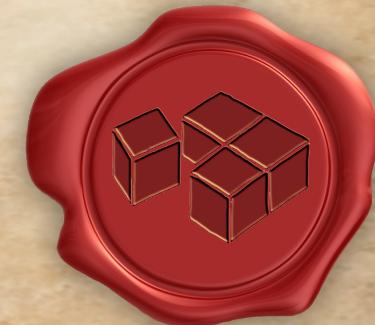
# King Discipline Architecture



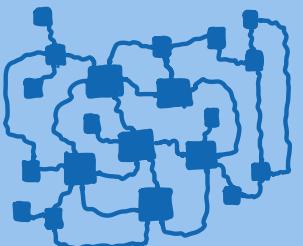
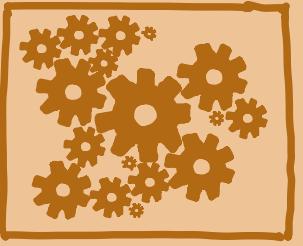
# Manifesto for IT Architecture

## Continuously Raising the Bar

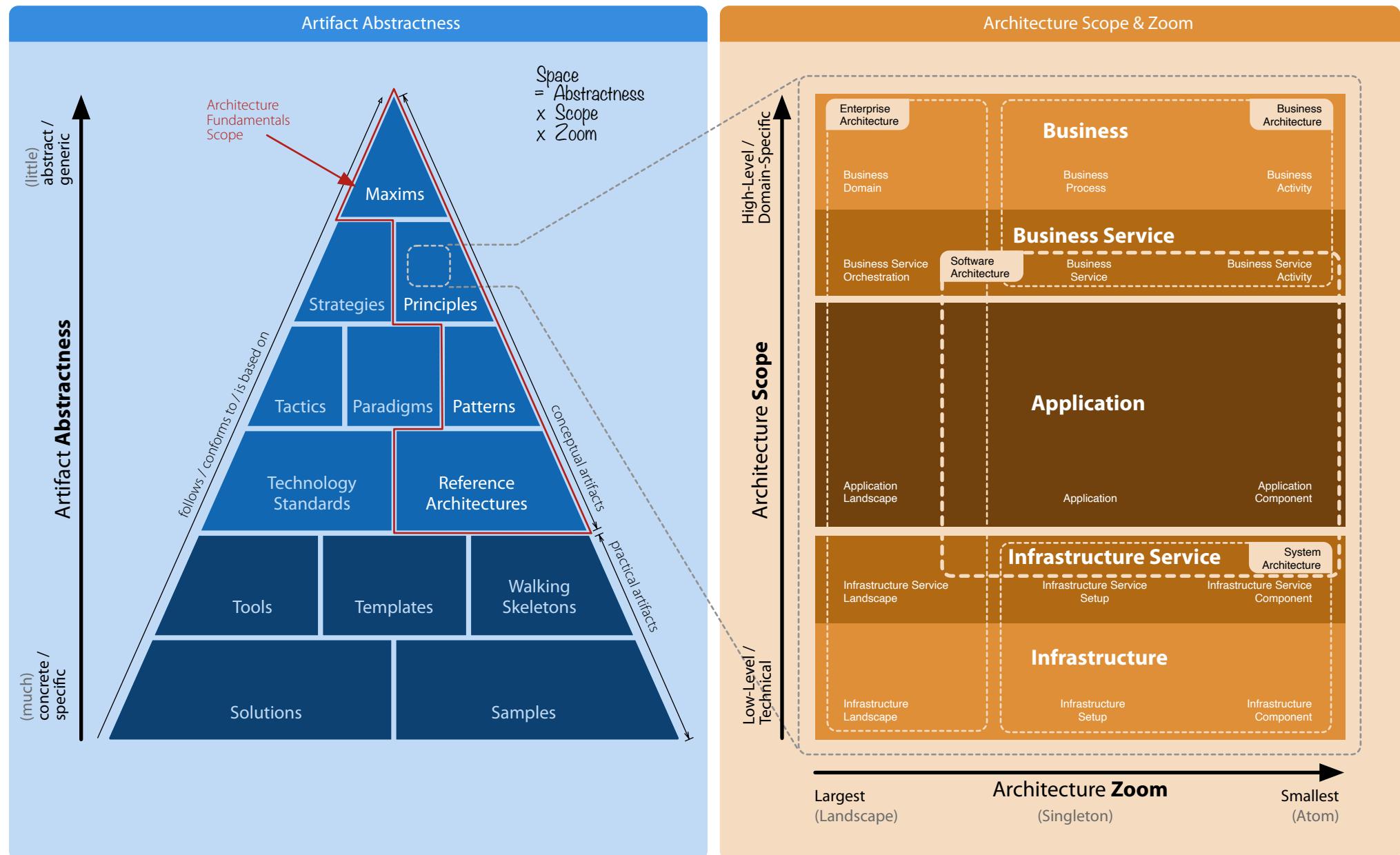
Mission	As IT Architects we guide the design, implementation and evolution of IT solutions.
Entitlement	We continuously strive to raise the bar of professional IT architecture by practicing it and helping others to learn our craft. We achieve maximum value for our clients through our work.
Values	Through 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.
Sustainable Concepts	over Latest Technologies
Pragmatic Making	over Theoretical Consideration
Constructive Craftsmanship	over Analytical Engineering
Accredited Creativity	over Achieved Industrialization
Proactive Improvement	over Reactive Correction
Inherent Quality	over Tested Robustness
Operational Delight	over Useful Functionality



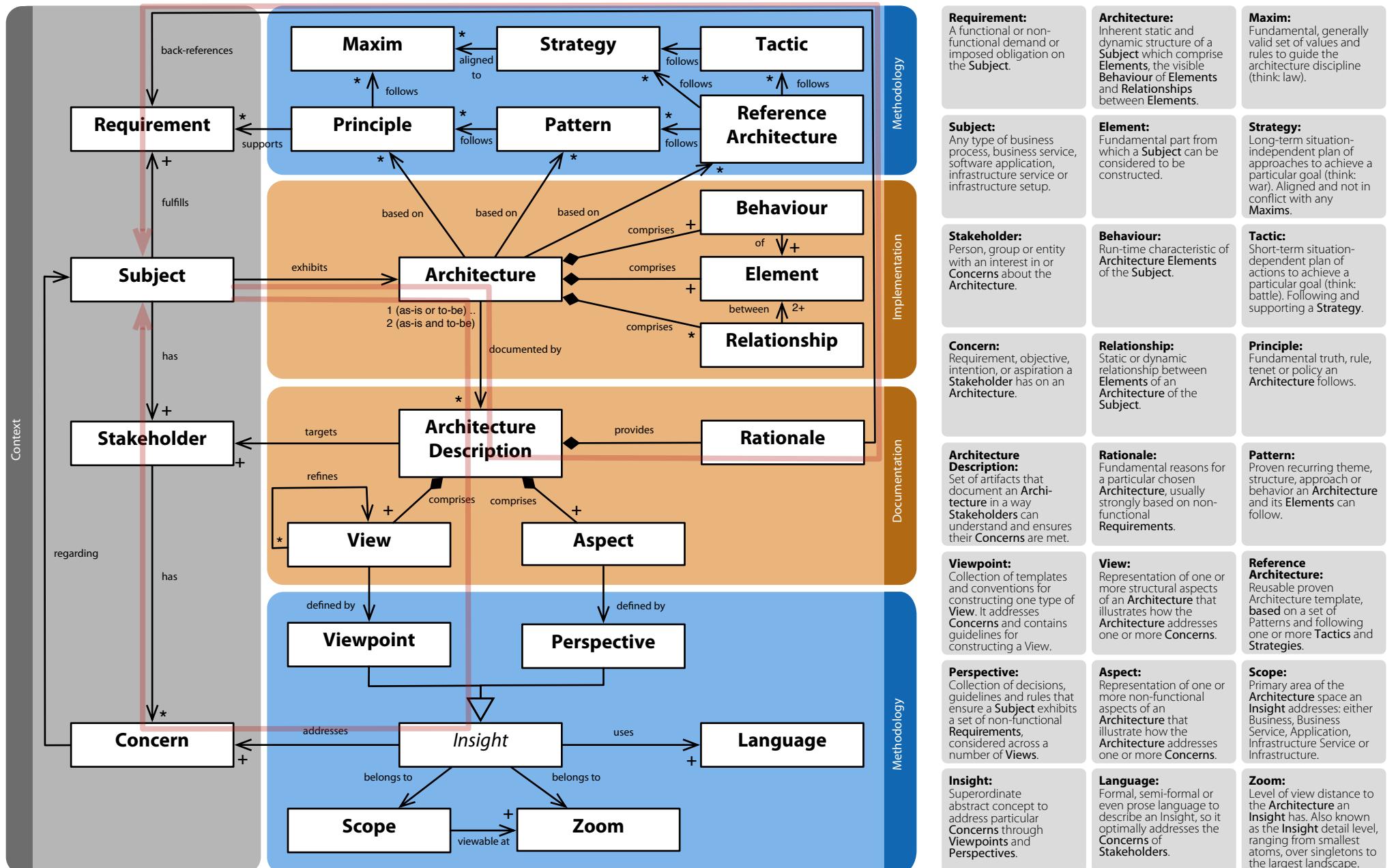
# Complex vs. Complicated

	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</b> in <b>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.
		
NOTICE	<b>Simple</b> (non-complicated) systems can be <b>complex</b> .	<b>Clear</b> (non-complex) systems can be <b>complicated</b> .
RECOGNIZE	<b>Architecture</b> primarily has to master the <b>complex</b> aspects of a system.	<b>Development</b> primarily has to master the <b>complicated</b> aspects of a system.

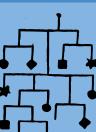
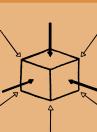
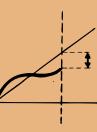
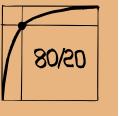
# Architecture Space



# Architecture Ontology



# Architecture Maxims

<b>Business Drives</b>	<b>BD</b>	<b>Component Orientation</b>	<b>CO</b>	<b>Separation of Business and Technology</b>	<b>BT</b>	<b>Adequate Description</b>	<b>AD</b>
Trigger and support the business with technological feasibilities, but always understand the business domain and its demands and align your architecture accordingly.	\$	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.		Strictly separate the business, i.e., domain-specific, aspects from the technological, i.e., infrastructural, aspects. Furthermore, ensure the explicit visibility of domain concepts.		Provide as much stakeholder-directed architecture description as necessary, and as little as possible.	
<b>Use-Case Driven Design</b>	<b>UC</b>	<b>Analytical and Creative Act</b>	<b>AC</b>	<b>Balance Principles Against Requirements</b>	<b>PR</b>	<b>Insights through Views &amp; Aspects</b>	<b>VA</b>
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.		Recognize that every good architecture is based on both analytical engineering and creative artistic aspects.		By weighing them against one another, find a reasonable balance between fundamental architecture principles and your particular non-functional requirements.		Give insights into your architecture through carefully selected stakeholder-directed separate views and aspects. Express each with the most suitable graphical or textual language.	
<b>Proven Basis</b>	<b>PB</b>	<b>Don't Be Too Clever</b>	<b>TC</b>	<b>Design for Failure Case</b>	<b>DF</b>	<b>Continuous Compliance</b>	<b>CC</b>
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 or tricky, both in your higher-level architecture and lower-level design aspects.		Murphy was an architect: everything which can fail will sometime ultimately fail. Hence, already design for the failure case (think: "pessimistic").		Continuously check through qualitative inspections and quantitative measurements whether your architecture and the non-functional requirements are followed and do not drift apart.	
<b>No Silver Bullet</b>	<b>SB</b>	<b>Simplicity Trumps</b>	<b>ST</b>	<b>Design to Change</b>	<b>DC</b>	<b>Integration-Figure Architect</b>	<b>IF</b>
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.		Create solution parts as simple as possible and only as complex as necessary. And remember: simplicity before generality, use before reuse!		Time changes everything, so your solution is already legacy at the first day of release. Hence, already design for its change (think: "agile").		Recognize that you, the architect, are the central integrating figure, having to bridge between the business and technology spheres of people.	
<b>Stepwise Refinement</b>	<b>SR</b>	<b>Perfect is the Enemy of Good Enough</b>	<b>GE</b>	<b>Explicit Decisions</b>	<b>ED</b>	<b>Eat Your Own Dog-Food</b>	<b>OF</b>
Start with the "big picture" and perform a stepwise top-down refinement of your architecture by going from coarse to fine aspects.		Beware of the perfection pitfall and design your architecture only as good as necessary and not as good as ultimately possible.		Record your major architecture decisions and rationales by taking into account and back-referencing the non-functional requirements.		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.	