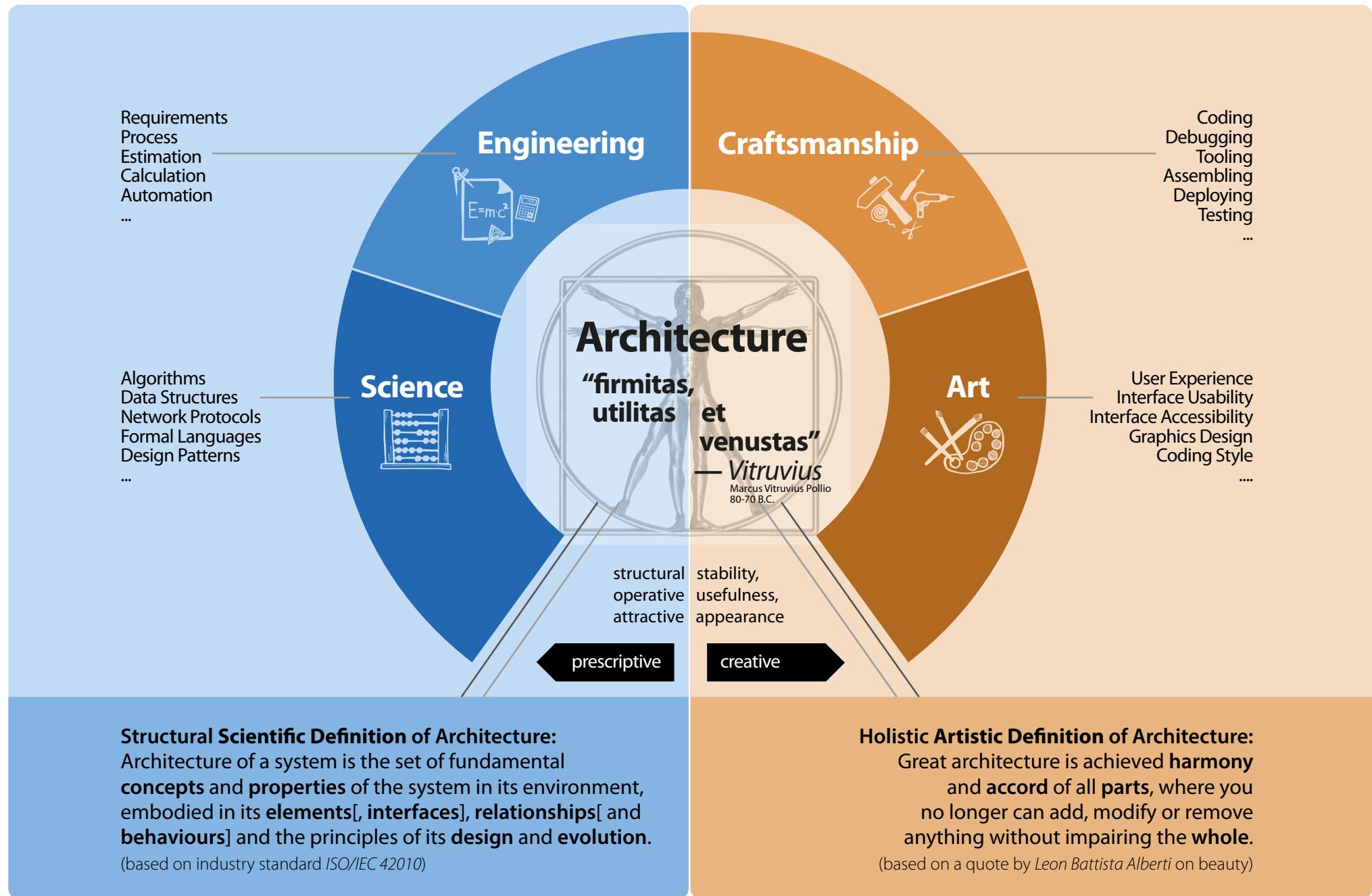




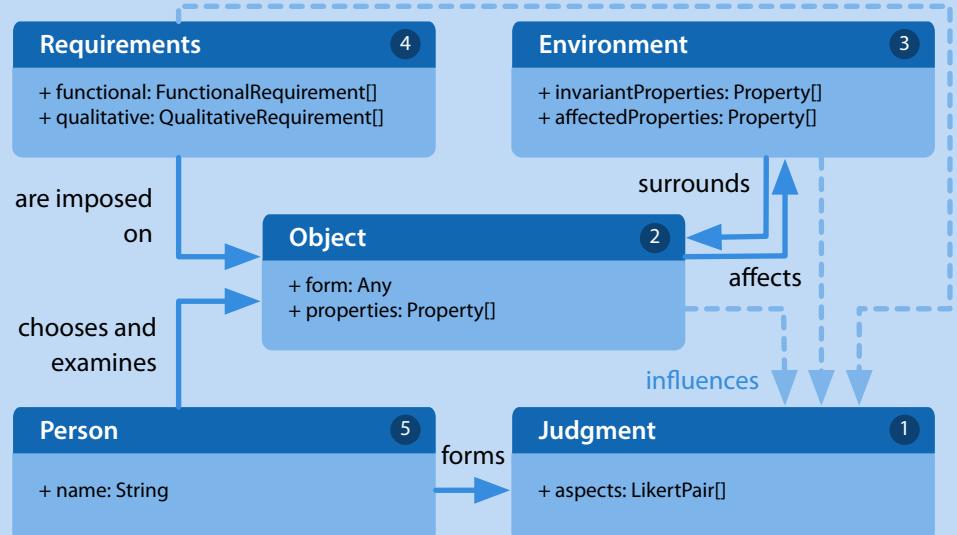
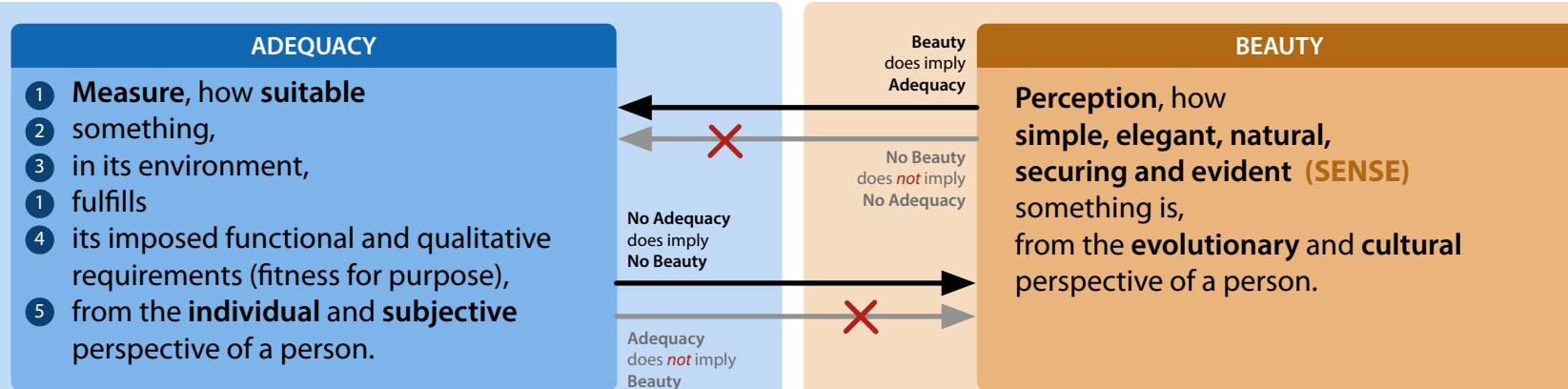
TECHNISCHE
UNIVERSITÄT
MÜNCHEN

Software Engineering in Industrial Practice (SEIP)

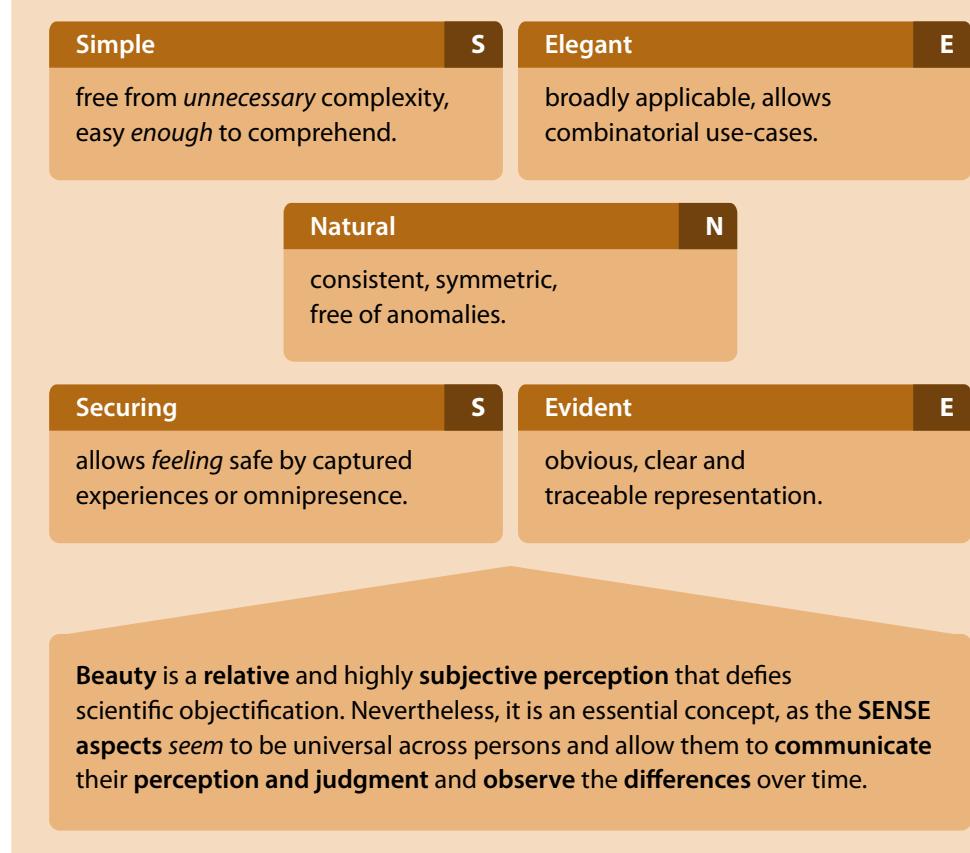
Dr. Ralf S. Engelschall



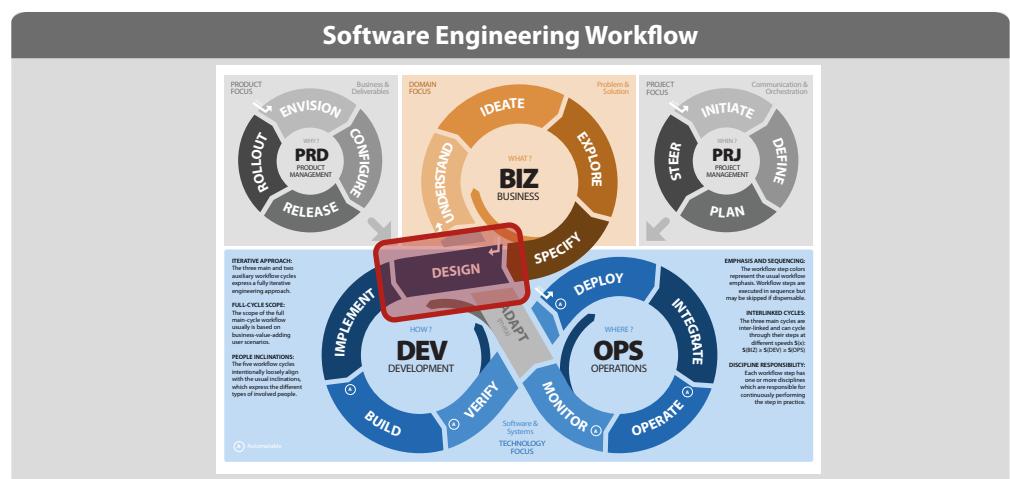
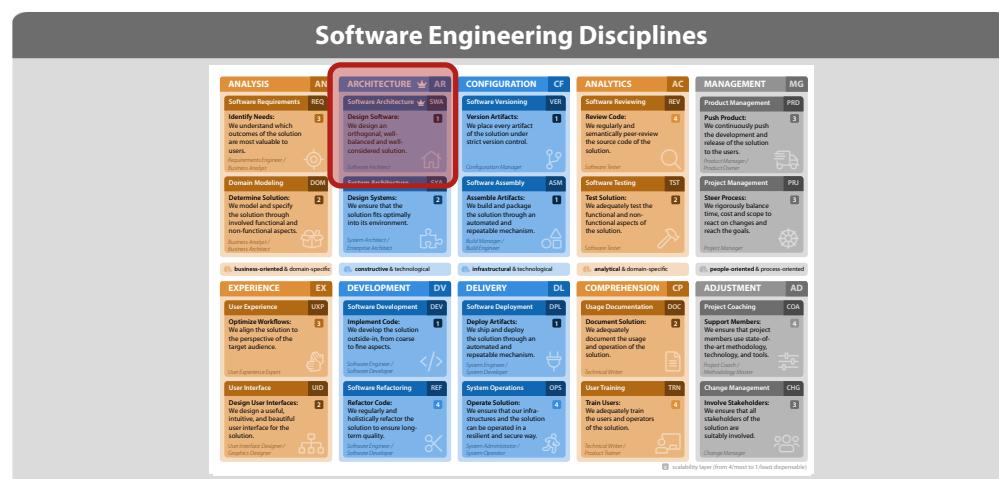
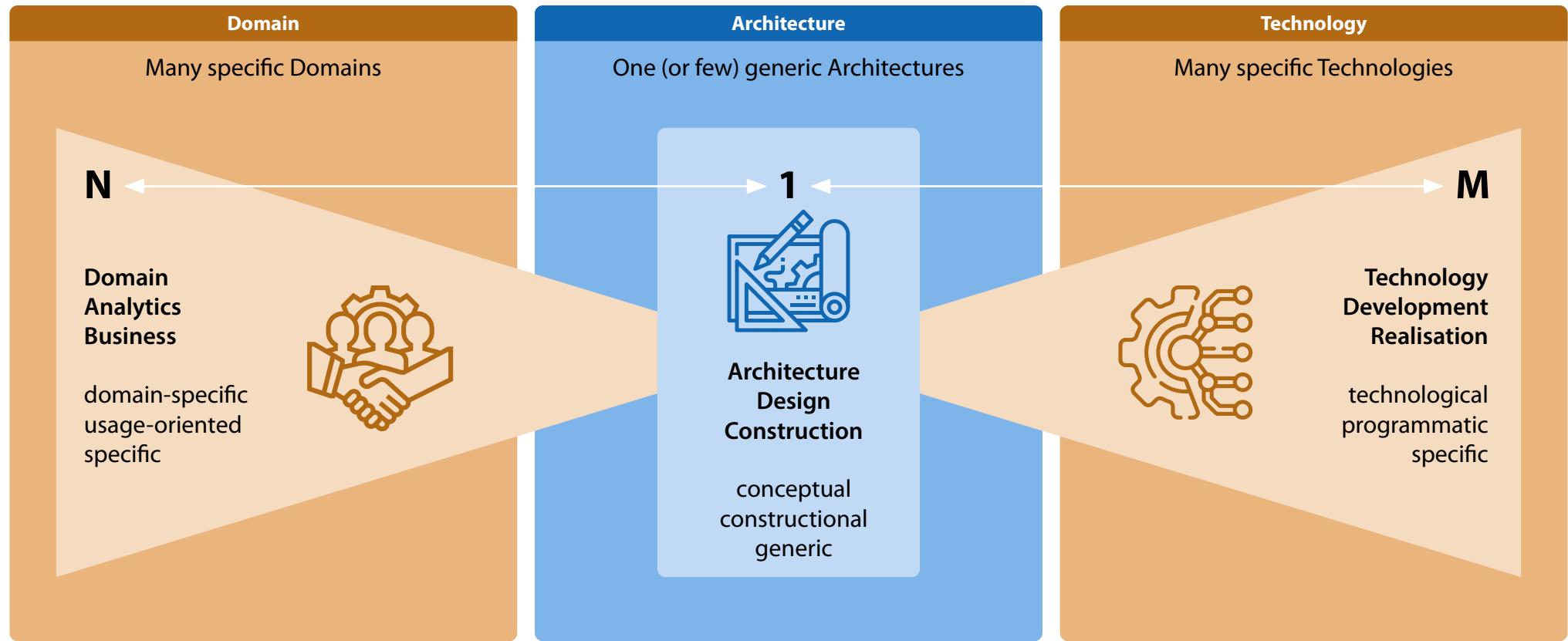
Adequacy and Beauty



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.



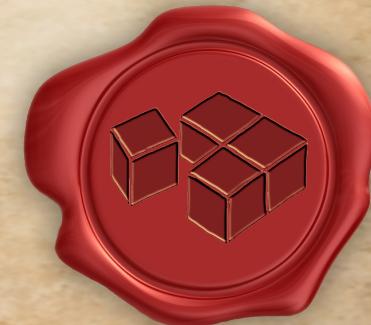
King Discipline Architecture



Manifesto for IT Architecture

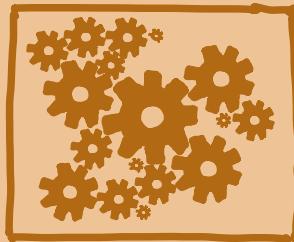
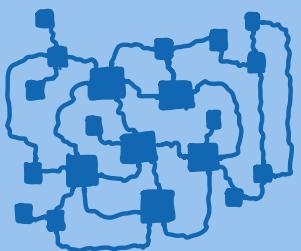
Continuously Raising the Bar

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 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

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

RECOGNIZE

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

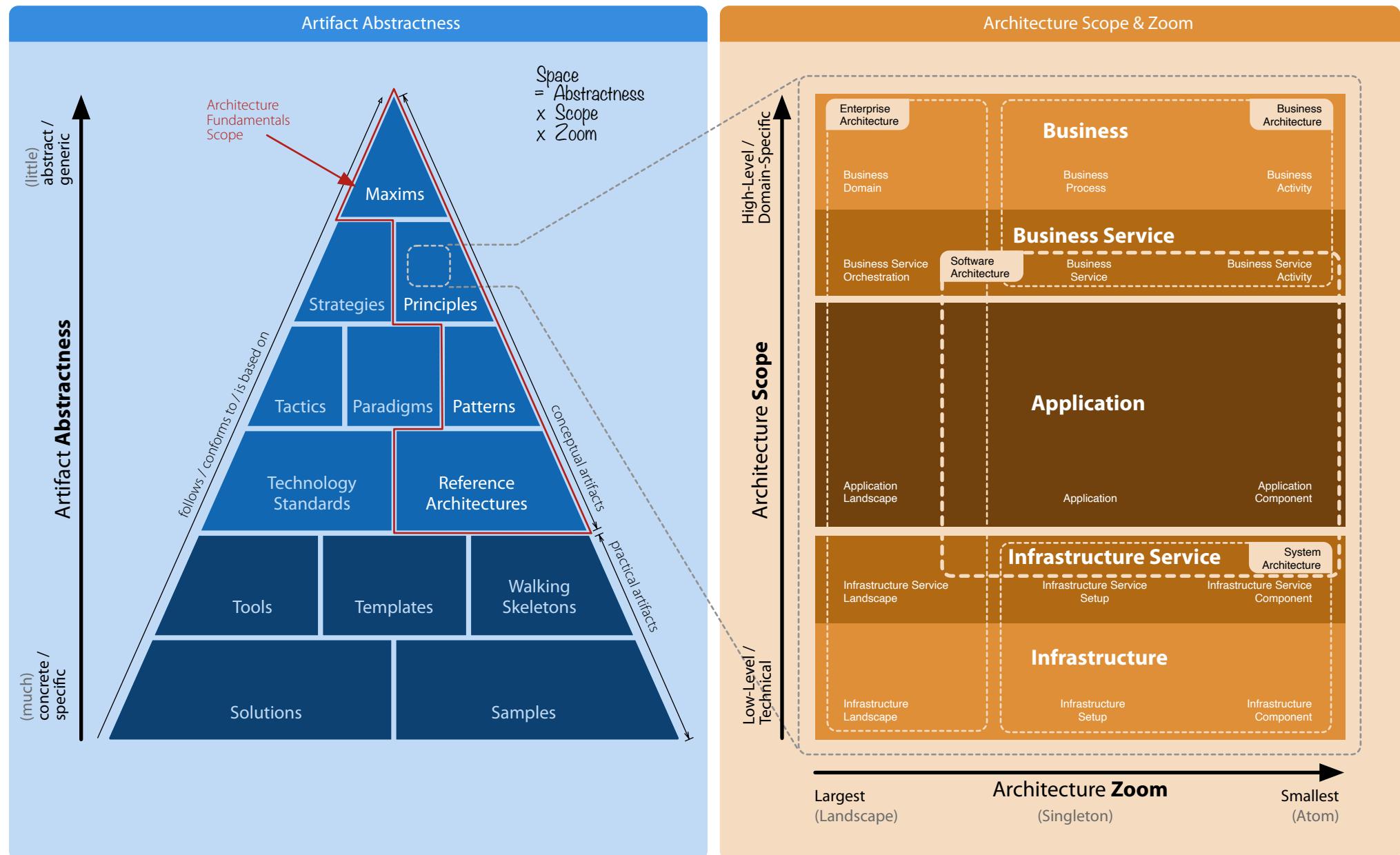
NOTICE

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

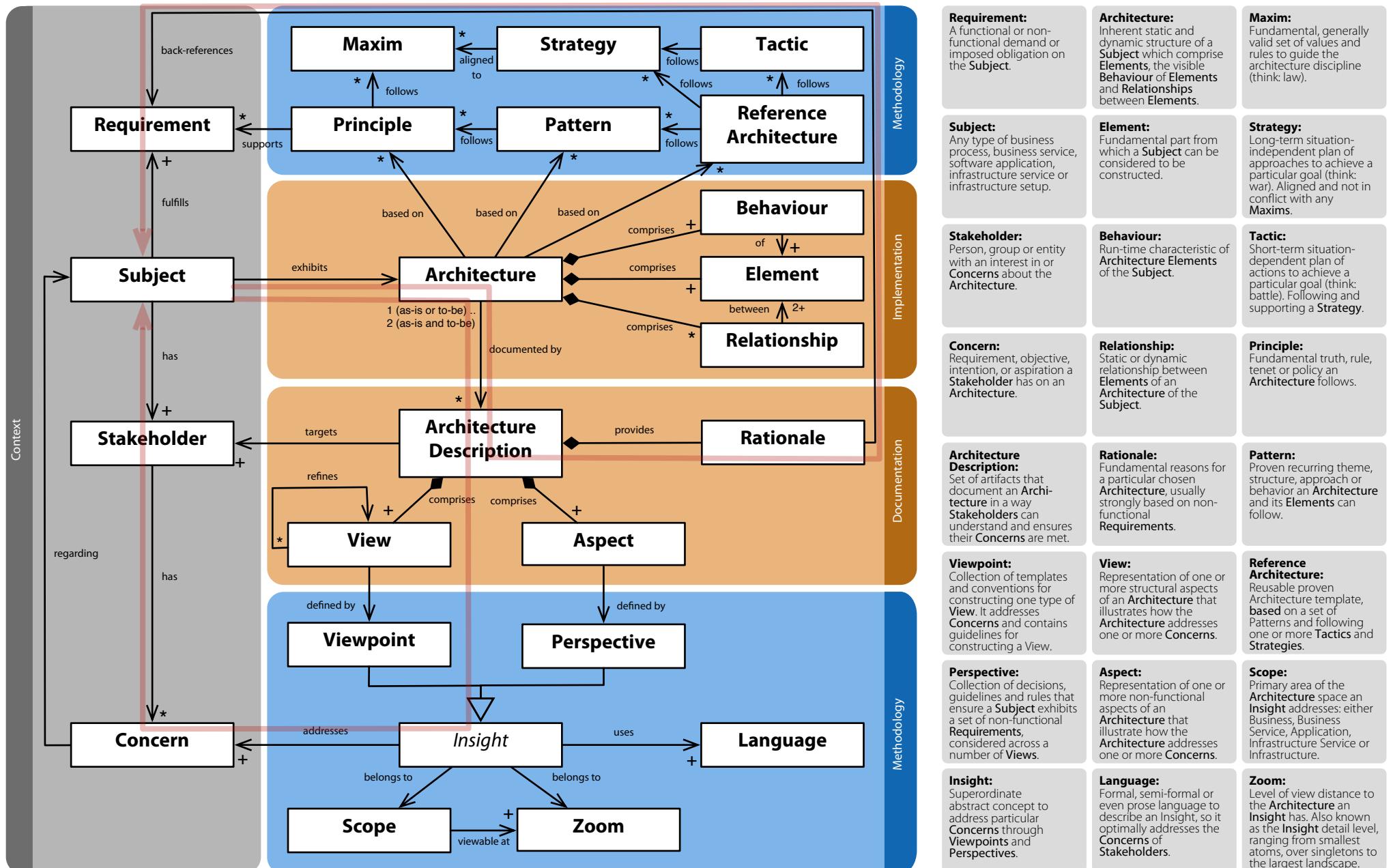
RECOGNIZE

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

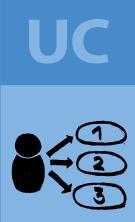
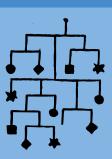
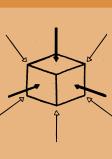
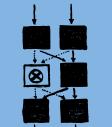
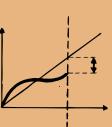
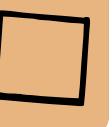
Architecture Space



Architecture Ontology



Architecture Maxims

Business Drives	BD	Component Orientation	CO	Separation of Business and Technology	BT	Adequate Description	AD
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.	
Use-Case Driven Design	UC	Analytical and Creative Act	AC	Balance Principles Against Requirements	PR	Insights through Views & Aspects	VA
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 (structure) and creative artistic (beauty) 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.	
Proven Basis	PB	Don't Be Too Clever	TC	Design for Failure Case	DF	Continuous Compliance	CC
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.	
No Silver Bullet	SB	Simplicity Trumps	ST	Design to Change	DC	Integration-Figure Architect	IF
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.	
Stepwise Refinement	SR	Perfect is the Enemy of Good Enough	GE	Explicit Decisions	ED	Eat Your Own Dog-Food	OF
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.	