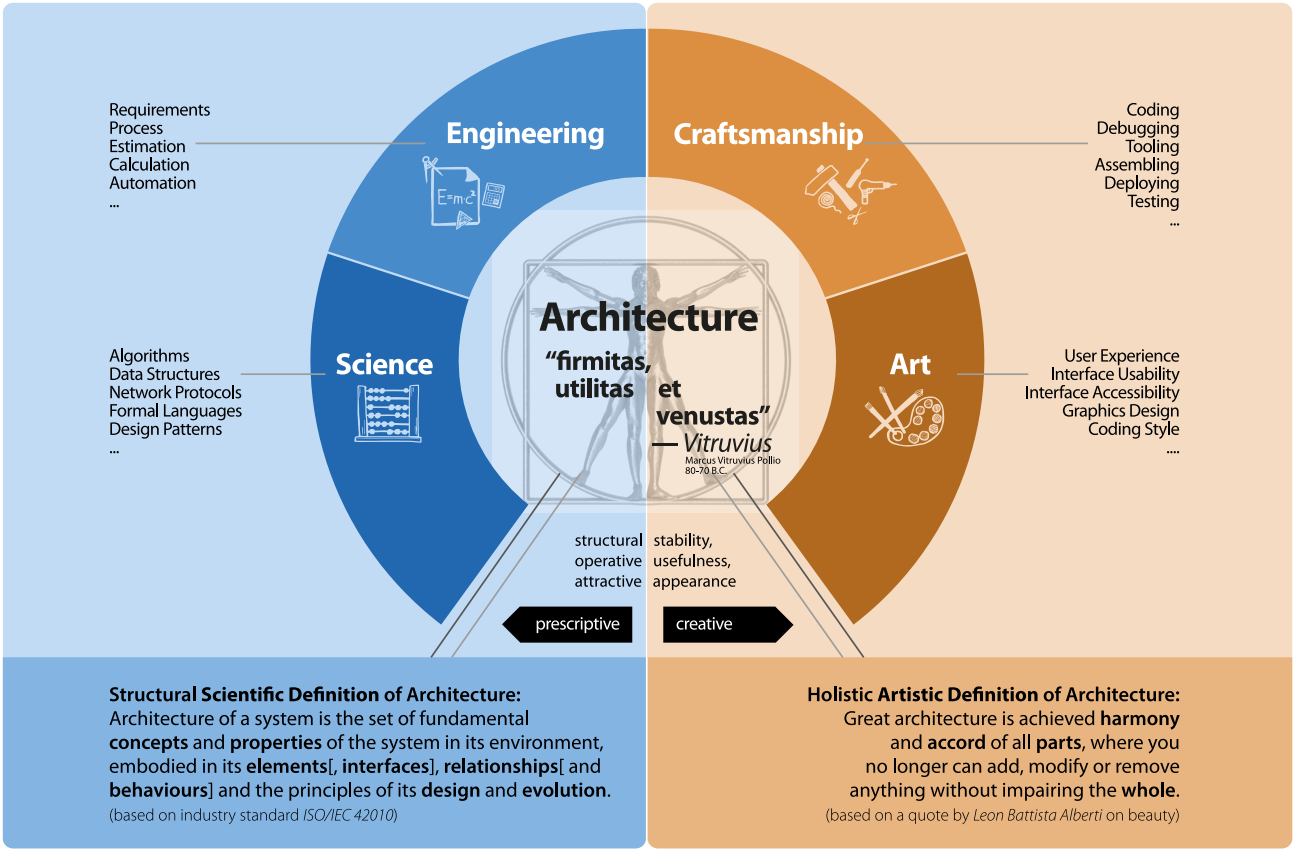
TECHNISCHE UNIVERSITÄT MÜNCHEN

Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

Architecture Stargate



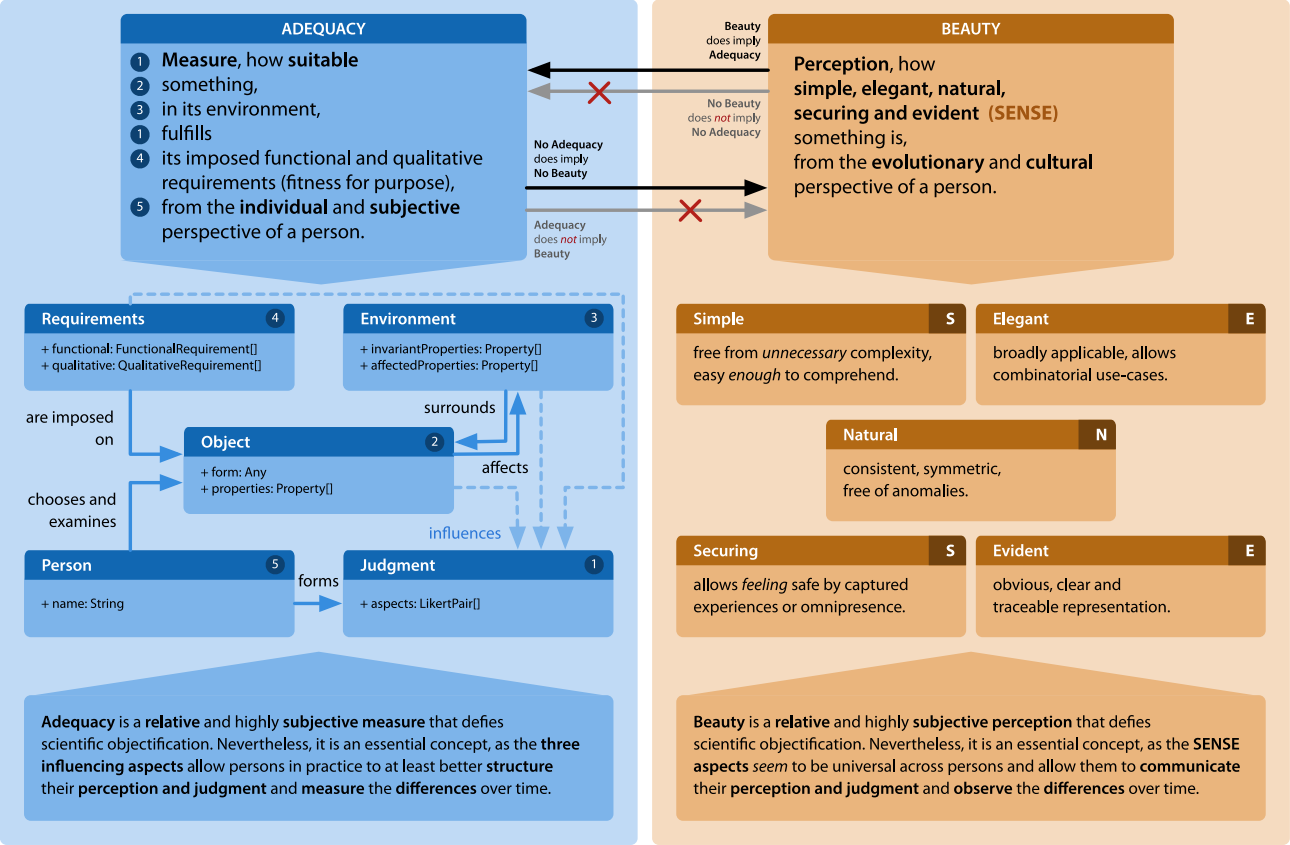
Architektur ist nicht einfach zu definieren. Man kann Architektur sowohl **strukturell wissenschaftlich** über messbare Elemente (elements), Schnittstellen (interfaces) und Beziehungen (relationships), als auch **ganzheitlich künstlerisch** über "die Harmonie und den Einklang aller Teile" definieren. Die "Wahrheit" liegt in der Praxis irgendwo dazwischen, denn die beiden Extrema spannen einen breiten Raum auf, in dem alle Lösungen in der Praxis liegen.

Auf der strukturell wissenschaftlichen Seite definiert sich Architektur über die Aspekte **Science** (insbesondere Computer Science) und **Engineering** (insbesondere Software Engineering). Auf der ganzheitlich künstlerischen Seite definiert sich Architektur über die Aspekte **Craftsmanship** (Handwerk, insbesondere Programmieren) und **Art** (Kunst, insbesondere User Experience).

Fragen

- Wie kann man **Architektur** definieren?
- Über welche vier **Aspekte** kann man **Architektur** aufspannen?

Adequacy and Beauty



Angemessenheit ist definiert als das Maß, wie geeignet etwas, in seiner Umgebung, die gestellten funktionalen und qualitativen Anforderungen erfüllt (Eignung für den Einsatzzweck), aus der individuellen Perspektive einer Person.

Angemessenheit ist ein relatives und höchst subjektives Maß, das sich einer wissenschaftlichen Objektivierung entzieht. Dennoch ist Angemessenheit ein wichtiges Konzept, da die drei beeinflussenden Aspekte (Anforderungen, Umgebung, Objekt) es den Personen in der Praxis ermöglichen, ihre Wahrnehmung und ihr Urteil zumindest besser zu strukturieren und die Unterschiede über die Zeit zu messen.

Schönheit ist definiert als die Wahrnehmung, wie einfach, elegant, natürlich, sicher und offensichtlich (SENSE) etwas ist, aus der evolutionären Perspektive eines Menschen.

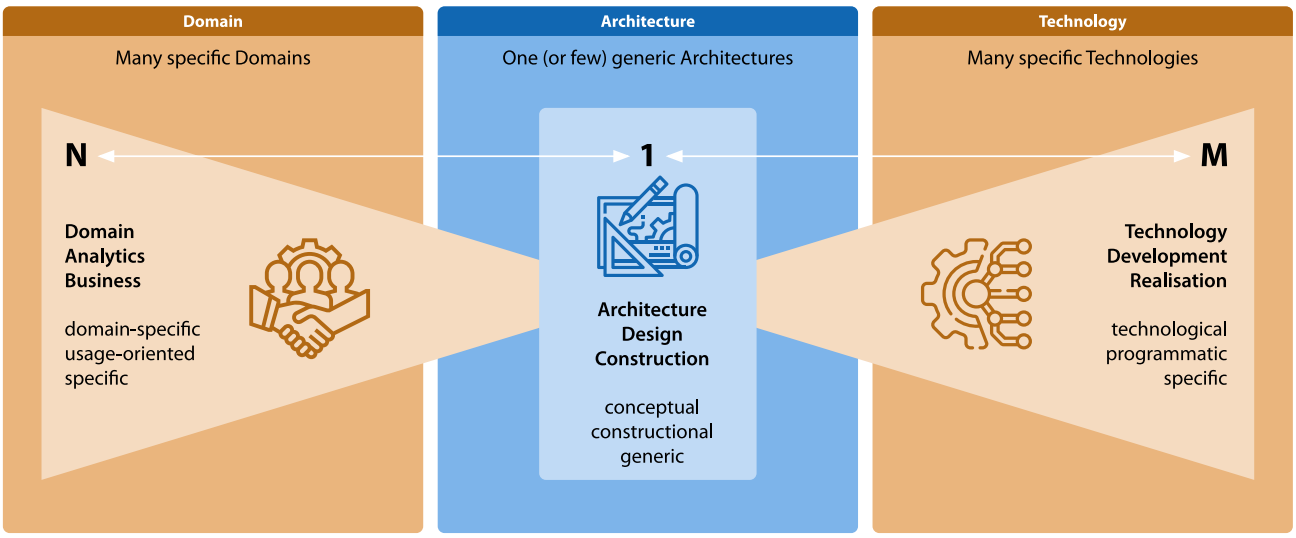
Schönheit ist eine absolute und höchst subjektive Wahrnehmung, die sich einer wissenschaftlichen Objektivierung entzieht. Dennoch ist Schönheit ein wichtiges Konzept, da die SENSE-Aspekte universell über Personen hinweg zu sein scheinen und es ihnen erlauben ihre Wahrnehmung und ihr Urteil mitzuteilen und die Unterschiede über die Zeit zu messen.

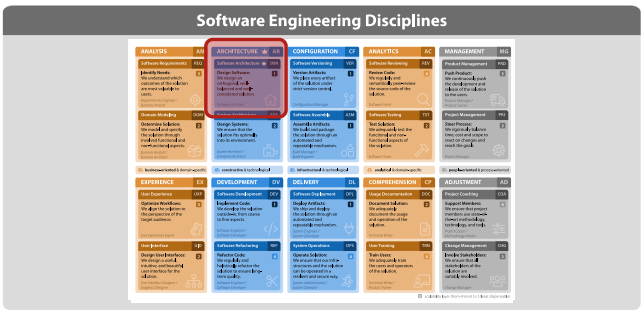
Fragen

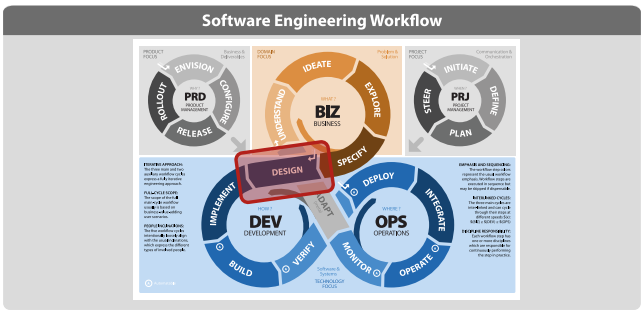
- Kann man Angemessenheit oder Schönheit allgemein messen?
- Kann man Angemessenheit oder Schönheit im Kontext einer einzelnen Person messen?

Intellectual Content: Version 1.0.2 (2022-10-09), Authored 2021-2022 by Dr. Ralf S. Engelschall, based on pre-work of Stefan Bratsch
Graphical Illustration: Version 1.0.2 (2022-10-08), Copyright © 2021-2022 Dr. Ralf S. Engelschall <http://engelschall.com>, All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

King Discipline Architecture







(Software) **Architecture** gilt als die "Königsdisziplin" im **Software Engineering**, da es das zentrale, allgemeine und konzeptuelle Bindeglied zwischen den vielen, potentiellen, spezifischen, realisierten **Fachlichkeiten** und den vielen, potentiellen, spezifischen, realisierenden **Technologien** ist. Die architekturelle Konstruktion einer Anwendung findet im logischen Schritt "Design" innerhalb des BizDevOps-Workflows im Software Engineering statt.

Fragen

? Wieso gilt **Architektur** als die "Königsdisziplin" im Software Engineering?

Intellectual Content: Version 1.0.0 (2023-10-20), Authored 2022 by Dr. Ralf D. Engelschall
Graphical Illustration: Version 1.0.0 (2024-04-06), Copyright © 2022 Dr. Ralf D. Engelschall <http://engelschall.com>, All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (LMU) for reproduction in Computer Science lecture contexts only.

Architecture Manifesto

Das **Manifesto for IT Architecture** ist eine Grundsatzerklärung für IT-Architektur. Sie besagt zuallererst, daß kontinuierlich die Meßlatte zu erhöhen ist ("Continuously Raising the Bar"), da nach gerade Mal 50 Jahren Software Engineering und Software Architecture wir zwar bereits etliche Best Practices kennen, aber die Disziplin sich sicherlich noch sehr lange weiterentwickeln muss.

Der Auftrag (Mission) an IT-Architekten ist, die Konstruktion, die Implementierung und die Weiterentwicklung/Wartung von IT-Lösungen zu leiten. Der Anspruch (Entitlement) ist es, dabei die Messlatte kontinuierlich zu erhöhen und anderen zu helfen, das "Handwerk" zu erlernen. Selbstredend ist die Tatsache, daß durch die Arbeit von Architekten maximaler Mehrwert für die Kunden erzielt wird.

Die Basis-Werte (Values), welche in diesem Handwerk eine zentrale Rolle spielen und sehr geschätzt werden, sind: **Latest Technologies**, **Theoretical Consideration**, **Analytical Engineering**, **Archieved Industrialization**, **Reactive Correction**, **Tested Robustness** und **Useful Functionality**.

Darüber hinaus gibt es aber zusätzliche Werte, welche ebenfalls eine zentrale Rolle spielen und noch sehr viel mehr geschätzt werden: **Sustainable Concepts** (der Inhalt von **Architecture Fundamentals**!), **Pragmatic Making**, **Constructive Craftsmanship**, **Accredited Creativity**, **Proactive Improvement**, **Inherent Quality** und **Operational Delight**.

Fragen

? (keine)

Intellectual Content: Version 1.0.2 (2011-11-02), Authored 2013 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.1.1 (2018-11-06), Copyright © 2013-2018 Dr. Ralf S. Engelschall <http://engelschall.com>, All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contents only.

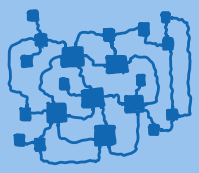
ARCHITECTURE FUNDAMENTALS

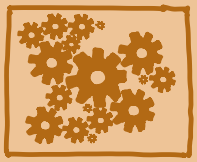
Complex vs. Complicated

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 01.5

	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.

"Komplex" bezieht sich auf die extrinsische Schwierigkeit eines Systems auf der höheren Ebene bzw. der Makro-Ebene, da das System aus vielen verschiedenen, miteinander verbundenen Teilen besteht. Es braucht deshalb Zeit, um dieses zu begreifen und als Ganzes zu meistern, obwohl jedes Teil des Systems meist leicht zu erklären ist.

"Kompliziert" bezieht sich auf die intrinsische Schwierigkeit eines Systems auf der unteren Ebene bzw. der Micro-Ebene, da das System viele verschiedene, schwierige Aspekte umfasst. Es braucht deshalb Zeit, um die Teile im Details zu verstehen und zu lernen und jedes Teil des Systems ist meist auch schwer zu erklären.

Beachte: Einfache (nicht-komplizierte) Systeme können komplex sein – offensichtliche (nicht-komplexe) Systeme können kompliziert sein.

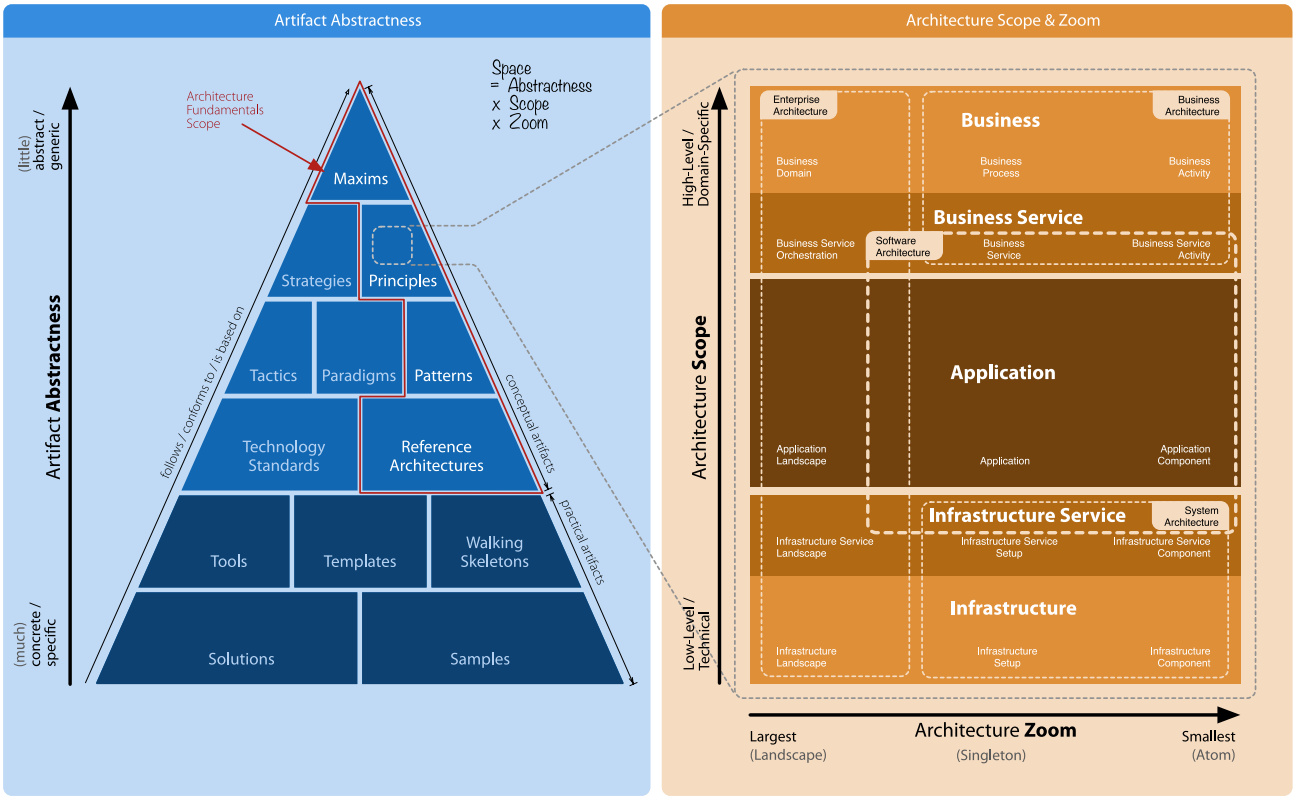
Der entscheidende Unterschied ist: Die Architektur bzw. die Konstruktion muss vor allem die komplexen Aspekte eines Systems meistern. Die Entwicklung bzw. die Realisierung muss vor allem die komplizierten Aspekte eines Systems meistern.

Fragen

? Muss sich **Architektur** primär um **komplexe** oder **komplizierte** Aspekte eines Systems kümmern?

Intellectual Content: Version 1.0.0 (2019-10-12), Authored 2019 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.0.0 (2019-10-12), Copyright © 2019 Dr. Ralf S. Engelschall <http://engelschall.com>, All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

Architecture Space



Der **IT Architecture Space** besteht aus drei Dimensionen: der **Artifact Abstractness** (einem Abstraktheitsgrad aller Artefakte, die der Architekt kennt), dem **Architecture Scope** (dem Bereich von Architektur, in dem man agiert) und dem **Architecture Zoom** (der Detailstufe von Architektur, in der man agiert).

Man unterscheidet primär drei Architecture Scopes: (high-level/domain-specific) **Business**, **Application** und (low-level/technical) **Infrastructure**. Zusätzlich werden die zwei sekundären Architecture Scopes **Business Service** und **Infrastructure Service** genutzt, um den "gedanklichen Sprung" von Business zu Application und von Application zu Infrastructure in der Praxis deutlich kleiner und sinnvoller ausfallen zu lassen.

Auf jedem Architecture Scope kann man mindestens drei verschiedene **Architecture Zooms** unterscheiden: von **Landscape** (largest), über **Singleton** bis zu **Atom** (smallest).

Im Raum aus Architecture Scope und Zoom unterscheidet man vier Arten von **IT Architecture**: **Business Architecture** (Facharchitektur), **Software Architecture** (Software-Architektur), **System Architecture** (System-Architektur) und **Enterprise Architecture** (Unternehmens-Architektur).

Umso höher/niedriger der Abstraktheitsgrad der Artefakte, umso größer/kleiner ist üblicherweise die Abdeckung des Raums aus Architecture Scope und Architecture Zoom.

Fragen

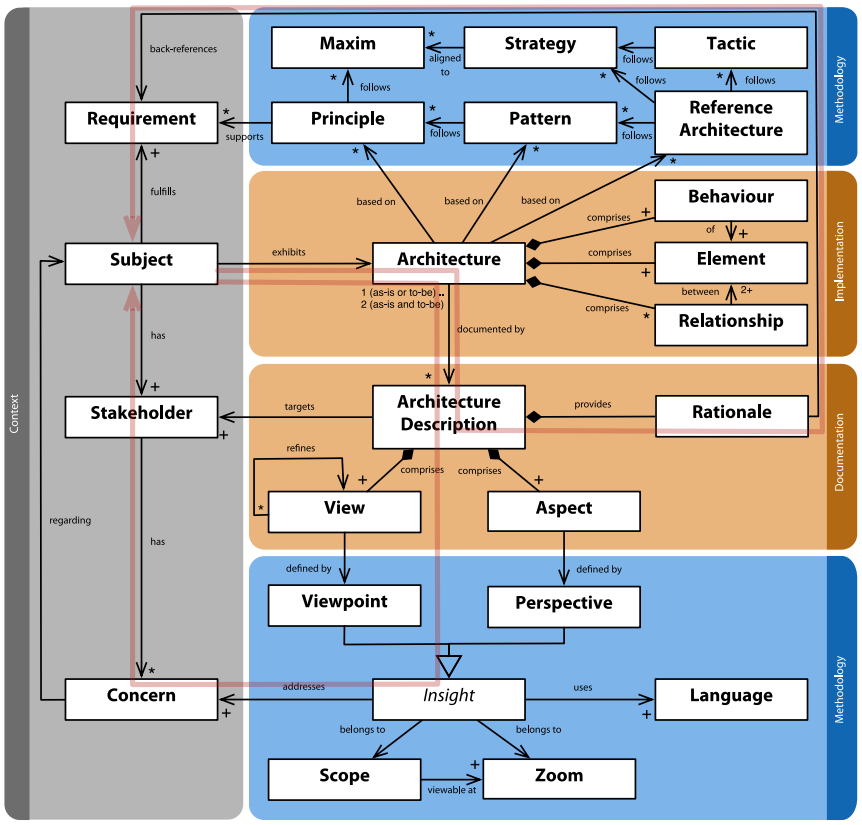
- Aus welchen drei Dimensionen besteht der IT **Architecture Space**?
- Welche vier Arten von Architektur kennt man in der IT?

Intellectual content: Version 1.0.0 (20120420), Authored 2011-2020 by Dr. Ralf S. Engelschall and Marc Mutz, based on presentations of mgi Architect Technology Research. Graphical illustration: Version 1.1.2 (20190405), Copyright © 2011-2019 Dr. Ralf S. Engelschall <http://engelschall.com>. All rights reserved. Unauthorized reproduction prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contents only.

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.

AF 02.2

Fundamental Content, Version 1.1.8 (2012-06-18), Authored 2010-2012 by Dr. Ralf S. Engelschall, inspired by ISO/IEC/IEEE 42010:2011, IEEE 1471:2000 and The Open Group TOGAF 9
Graphical Illustration, Version 1.1.11 (2012-06-21), Copyright © 2011-2012 Dr. Ralf S. Engelschall (http://engelschall.com), All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

Damit Architekten sinnvoll in der Praxis kommunizieren können, muss man sich auf ein paar wenige Grundbegriffe und deren Bedeutung einigen. Die Begriffe werden in einer Taxonomie definiert und in der **Architecture Ontology** in Beziehung zueinander gesetzt.

In der Architecture Ontology gibt es vor allem zwei wichtige "Schleifen". Beide starten beim **Subject**, welches eine **Architecture** hat, welche über die **Architecture Description** (im Deutschen "IT-Konzept") beschrieben wird.

Schleife 1: Die Architecture Description liefert **Rationales** (Begründungen) für Entscheidungen, welche im Idealfall auf **Requirements** zurück referenzieren sind. Denn eine Architecture Description soll nicht das **WAS** sondern vor allem das **WARUM** dokumentieren. Denn das WAS sieht man auch im Code, aber das WARUM nicht!

Schleife 2: Die Architecture Description besteht aus **Views** (Sichten) und **Aspects** (Aspekten), welche methodisch **Viewpoints** und **Perspectives** genannt werden. Beide zusammen liefern **Insights** (Einblicke) auf **Scope** und **Zoom Level** (siehe Architecture Space!) und werden über eine bestimmte (graphische oder textuelle) **Language** dokumentiert. In jedem Fall werden nur solche Insights gegeben, welche einen **Concern** (Anliegen) eines **Stakeholders** (Interessenvertreter) adressieren. Denn man programmiert auch nichts, was nicht gebraucht wird!

Fragen

- Was sollte eine **Architecture Description** (IT-Konzept) neben dem **WAS** vor allem dokumentieren?
- Was sollte eine **Architecture Description** (IT-Konzept) über **Insights** (Einblicke) vor allem addressieren?

ARCHITECTURE FUNDAMENTALS

Architecture Maxims

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 04.1

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 and creative artistic 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.

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.

Icon-Portal Content: Version 1.0.13.0200(24/09/19), Authored 2009-2020 by Dr. Ralf S. Engelschall
Graphic-Illumination: Version 0.8.8.0200-1-1280, Copyright © 2012-2020 Dr. Ralf S. Engelschall <http://engelschall.com>, All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

In der IT Architecture folgt man **Architecture Maxims**, welche grundlegende Richtlinien sind. Man kennt 20 Maxime. Der Architekt sollte den Maximen immer folgen und sie nie brechen.

Beachte: **Proven Basis** und **No Silver Bullet** sagen, daß bei einer Architektur immer auf einer bewährten Basis (z.B. eine Referenz-Architektur) aufgesetzt werden muss, es aber gleichzeitig klar sein muss, daß man diese nicht 1:1 verwenden kann, sondern immer erst anpassen muss.

Beachte: **Stepwise Refinement** und **Component Orientation** sagen, daß vom zeitlichen Prozess her (und aus Gründen der Risikominimierung) man immer vom Groben zum Feinen geht, während die Ergebnisse eine stringente Komponenten-Orienterung aufweisen, wo kleine Komponenten hierarchisch in größeren Komponenten integriert sind.

Beachte: während man als IT-Architekt alle anderen Maxime eigentlich nur akzeptieren muss, ist **Simplicity Trumps** von einer anderen Qualität: nichts in der IT ist wirklich einfach. Wenn etwas einfach wirkt, versteht man meist nur noch nicht genügend davon. Oder jemand hat richtig viel investiert, um es einfach wirken zu lassen. **Simplicity Trumps** bedeutet genau dies zu tun: inhärent komplexe Dinge erst wieder einfach werden zu lassen.

Fragen

? Aus Gründen der Risikominimierung sollte der IT-Architekt beim **Stepwise Refinement** schrittweise immer wie vorgehen?