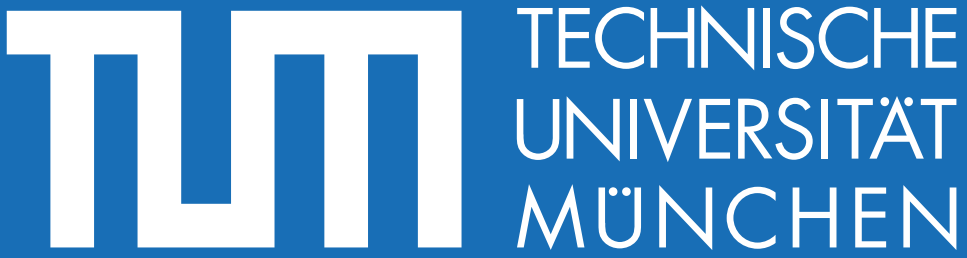
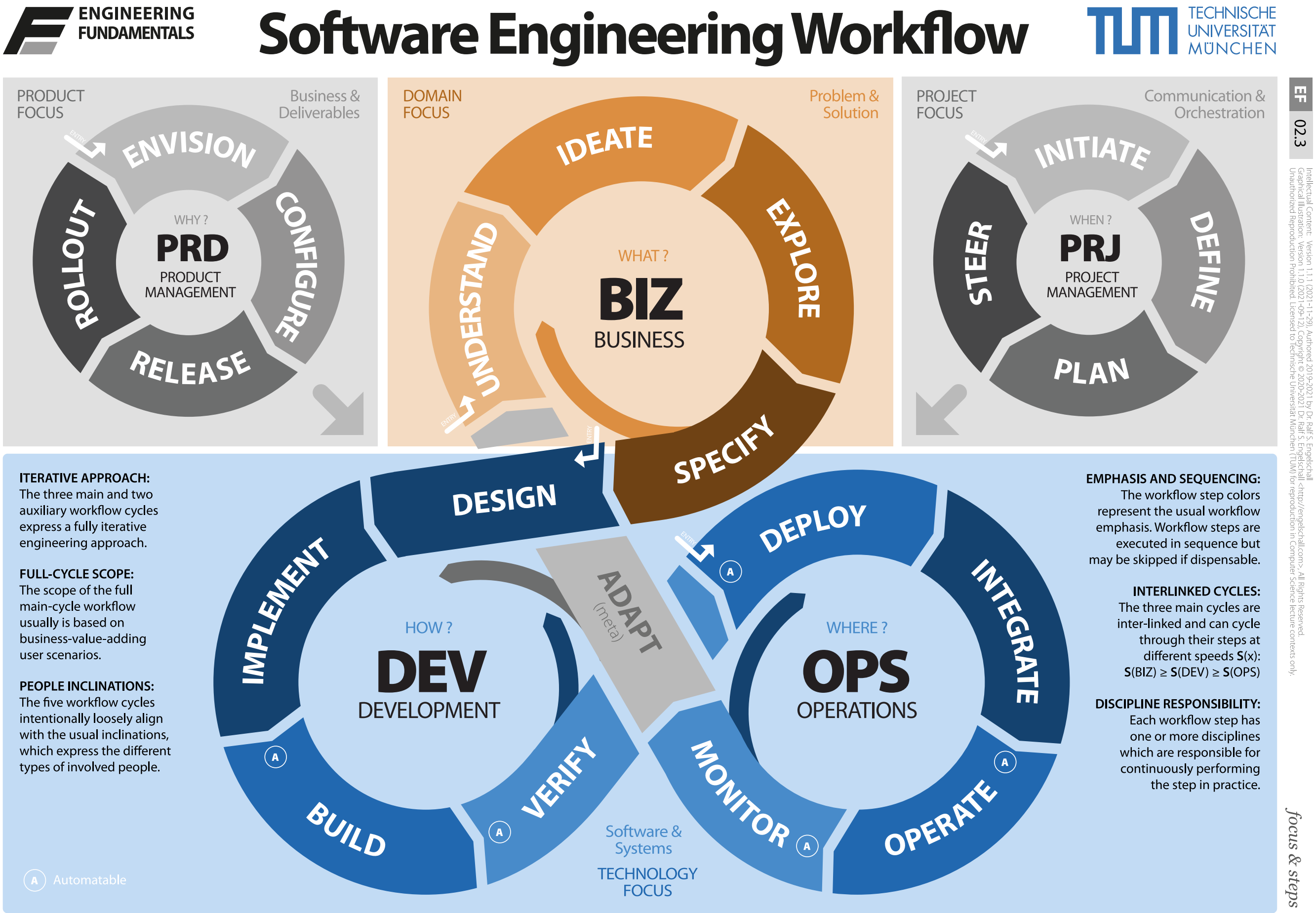


Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall



Das Workflow-Modell beschreibt die Arbeitsteilung im Software Engineering. Im **Workflow** drücken drei Haupt- und zwei Neben-Workflow-Zyklen das iterative Vorgehen aus. Der vollständige Haupt-Workflow-Zyklus basiert in der Regel auf Geschäfts-wertschöpfenden Anwenderszenarien.

Die drei Hauptzyklen sind miteinander verknüpft und können ihre Schritte mit unterschiedlichen Geschwindigkeiten S(x) durchlaufen, wobei S(BIZ) größer oder gleich S(DEV) ist, welches wiederum größer oder gleich S(OPS) ist. Denn die Zyklen mit früheren Schritten sollten die Zyklen nicht späteren Schritten nicht ausbremsen.

Die Schrittfarben repräsentieren den Arbeitsablaufschwerpunkt. Arbeitsablaufschritte werden der Reihe nach ausgeführt, können aber übersprungen werden, wenn sie entbehrlich sind.

Die zehn Disziplinen des Software Engineering drücken die unterschiedlichen Rollen der beteiligten Personen aus. Die fünf Neigungen drücken die verschiedenen Typen der beteiligten Personen aus. Deshalb besteht der Workflow aus genau fünf Zyklen.

Fragen

- ❓ Was bescheibt das Workflow-Modell im Software Engineering?

ENGINEERING FUNDAMENTALS

Software Engineering Steps

TECHNISCHE UNIVERSITÄT MÜNCHEN

EF 02.4

steps & goals

Software Engineering kann auf operativer Ebene alternativ durch 20 verschiedene **Steps** verstanden werden, die innerhalb des **Software Engineering Workflow** kontinuierlich durchgeführt werden. Jeder Step gehört zu einer primär verantwortlichen Disziplin und keiner oder mehreren sekundär verantwortlichen Disziplinen.

Workflow Steps sind das geeignete Konzept, um zu verstehen, welche Aktivitäten in jeder Iteration eines **Software Engineering Prozesses** durchgeführt werden müssen.

Fragen

- Welches Konzept erlaubt einem am besten zu verstehen, welche Aktivitäten in einem Software Engineering Prozess durchgeführt werden müssen?

Software Engineering Process

1. WORKFLOW CYCLES
The workflow has five cycles which continuously iterate through their steps. Workflow steps are executed in each cycle in sequence, but may be skipped if dispensable in a particular iteration of the process. The length of an iteration is arbitrary, but can be e.g. about 1/3 of a Scrum sprint.

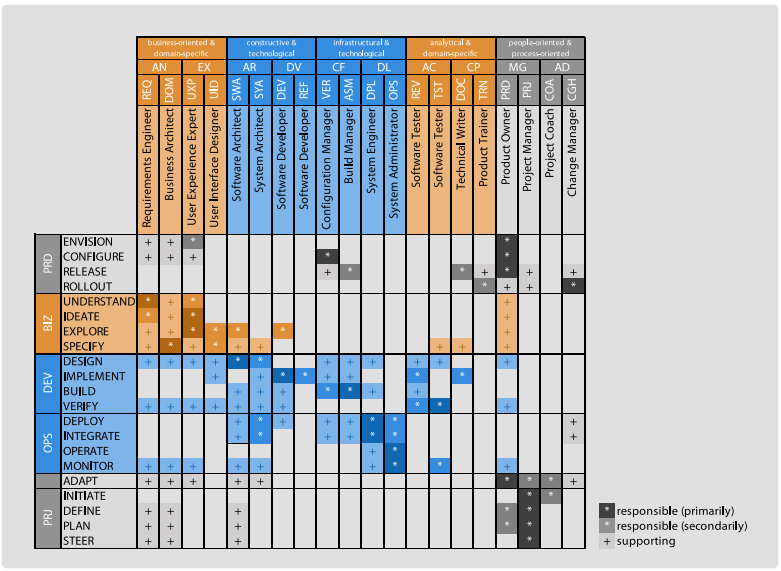
2. WORKFLOW STEPS:
The workflow steps describe a logical activity which has to be performed. Each step relates to one or more discipline areas and their corresponding disciplines, which express the operative responsibilities for each workflow step. In each discipline individual roles act.

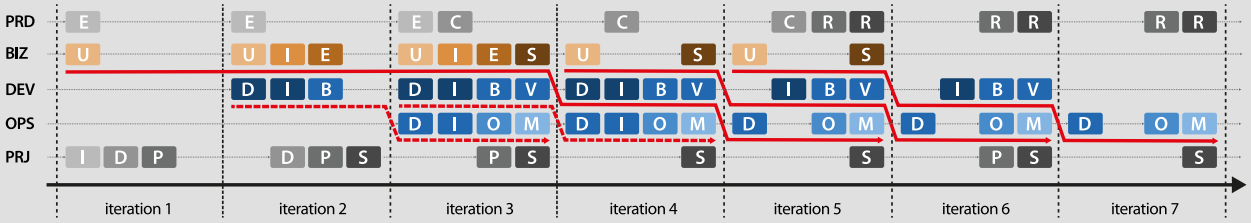
3. WORKFLOW ROLES:
The workflow roles are held by individual persons. Each role is primarily responsible for a particular workflow step. In addition, each role can be secondarily responsible for other workflow steps or at least actively support those steps.

4. PROJECT SCHEDULE:
To create a particular project execution schedule, the five cycles, their iterations and their steps have to be mapped onto a timeline. The cycles are mapped onto (horizontal) timeline tracks, the iterations are mapped onto (vertical) timeline phases, and the steps are mapped onto timeline activities.

5. PROCESS FLOWS (THE CRUX):
The activities across the cycles can (and should) be linked into individual (diagonal) waterfall-like flows, although the execution schedule, from the perspective of the cycles, is fully iterative. There are multiple such flows in parallel and they are usually highly interleaved on the project timeline in order to maximally utilize the team.

6. PROCESS ADAPTION:
In the meta-step ADAPT, the process is adapted by choosing which workflow steps are required for the next iteration. The major input for this decision is the current solution state and the feedback on it by the customer.





Der Arbeitsablauf hat fünf Hauptzyklen, die ihre Schritte kontinuierlich durchlaufen. Die Schritte des Arbeitsablaufs werden in jedem Zyklus der Reihe nach ausgeführt, können aber übersprungen werden, wenn sie entbehrlich sind.

Die Arbeitsablauf-Schritte sind mit Disziplinbereichen annotiert, um operative Verantwortlichkeiten auszudrücken. In jedem Bereich agieren mehrere Rollen.

Die Arbeitsablauf-Rollen werden von einzelnen Personen besetzt. Jede Rolle ist primär für einen bestimmten Arbeitsschritt verantwortlich. Darüber hinaus kann jede Rolle sekundär für andere Arbeitsschritte verantwortlich sein oder zumindest diese Schritte aktiv unterstützen.

Um einen bestimmten Projektablaufplan zu erstellen, müssen die fünf Zyklen, ihre Iterationen und ihre Schritte auf einer Zeitachse abgebildet werden. Die Zyklen werden auf (horizontale) Zeitleistenspuren abgebildet, die Iterationen werden auf (vertikale) Zeitleistenphasen abgebildet, und die Schritte werden auf Zeitleisten-Aktivitäten abgebildet.

Die Aktivitäten über die Zyklen hinweg können (und sollten) zu einzelnen (diagonalen) wasserfallartigen Flüssen verknüpft werden, obwohl der Ausführungsplan aus der Perspektive der Zyklen vollständig iterativ ist. Es gibt mehrere solcher Flüsse parallel und sie sind normalerweise stark auf der Projektzeitachse verschränkt, um das Team maximal auszulasten.

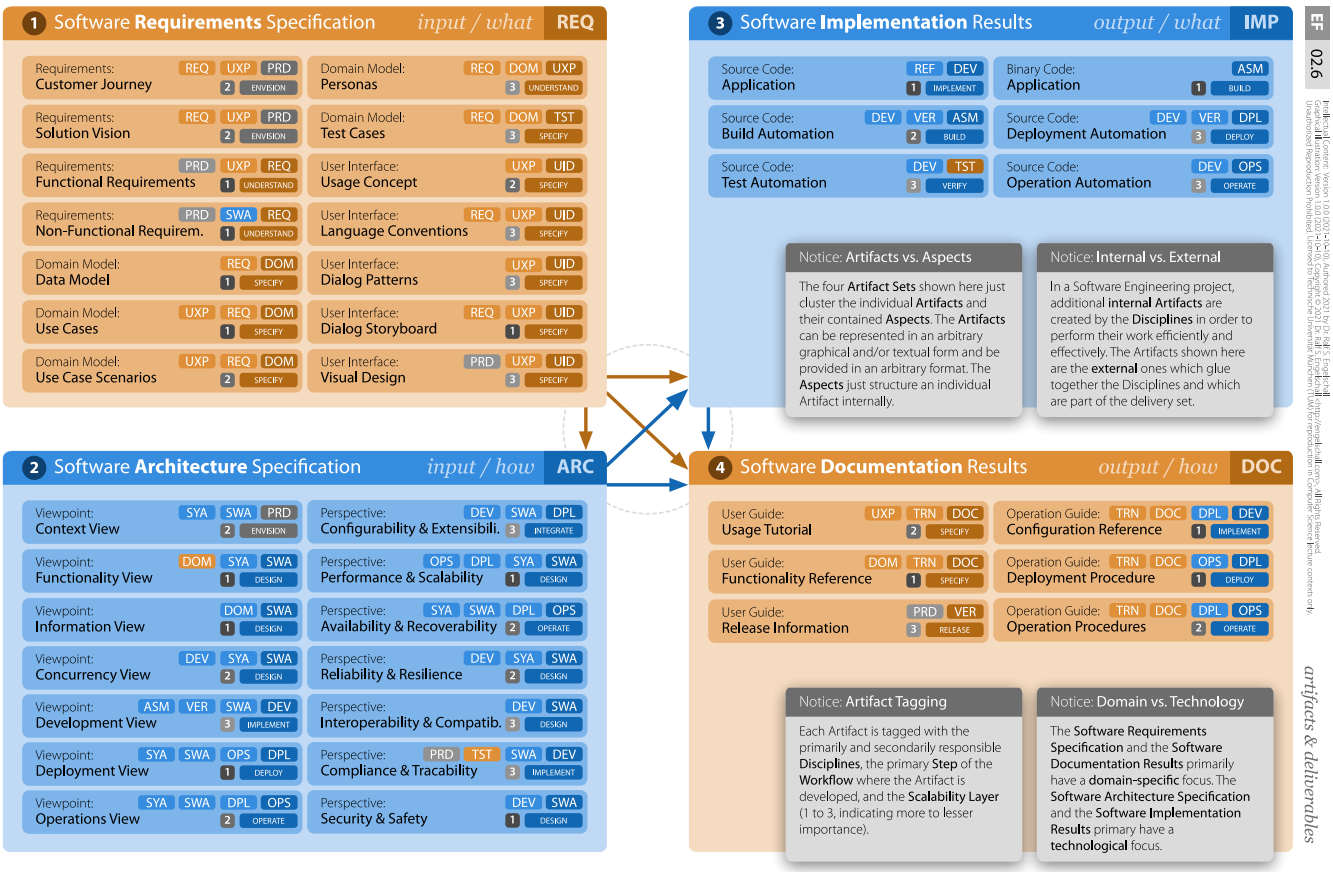
Fragen

- Wie wird trotz Arbeitsteilung eine maximale Auslastung des Teams im Software Engineering erreicht?

ENGINEERING FUNDAMENTALS

Software Engineering Artifacts

TECHNISCHE UNIVERSITÄT MÜNCHEN



Die vier **Artifact Sets** bündeln lediglich die einzelnen **Artifacts** und ihre enthaltenen **Aspects**. Die **Artifacts** können in beliebiger grafischer und/oder textueller Form dargestellt werden und in einem beliebigen Format bereitgestellt werden. Die **Aspects** strukturieren lediglich ein einzelnes Artifacts intern.

In einem Software-Engineering-Projekt werden zusätzliche **interne** Artifacts von den **Disziplinen** erstellt, um ihre Arbeit effizient und effektiv durchzuführen. Die gezeigten Artifacts sind nur die **externen** Artifacts, welche die Disziplinen verbinden und die Teil des Lieferumfangs sind.

Jedes **Artifact** ist mit der primären und den sekundären **Disciplines**, dem primären **Step** des Workflows in dem das Artifact entwickelt wird, und der Skalierbarkeitsstufe (1 bis 3, von mehr bis weniger wichtig) annotiert.

Die **Software Requirements Specification** und die **Software Documentation Results** haben primär einen **fachlichen** Fokus. Die **Software Architecture Specification** und die **Software Implementation Results** haben primär einen **technologischen** Fokus.

Fragen

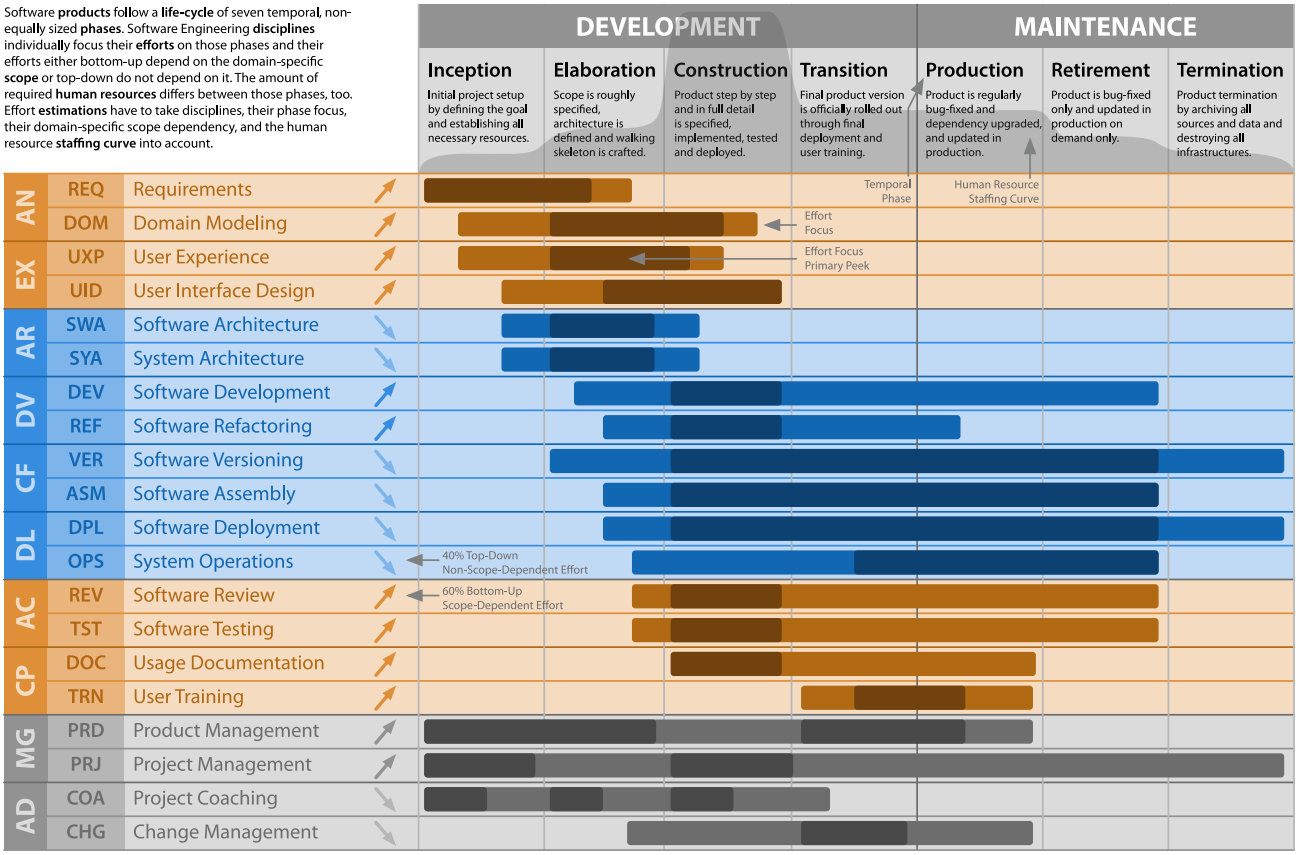
- Welchen Fokus hat die **Software Requirements Specification**?

ENGINEERING FUNDAMENTALS

Software Engineering Efforts

TECHNISCHE UNIVERSITÄT MÜNCHEN

EF 03.1



phases & efforts

Softwareprodukte folgen einem **Lebenszyklus** von sieben zeitlichen, nicht gleich großen **Phasen**. Die einzelnen **Disziplinen** des Software-Engineering fokussieren ihre **Aufwände** auf diese Phasen und ihre Aufwände sind entweder von unten nach oben vom fachlichen Umfang abhängig, oder von oben nach unten nicht davon abhängig. Der Umfang der erforderlichen **personellen Ressourcen** unterscheidet sich ebenfalls zwischen den Phasen.

Aufwandsschätzungen müssen die Disziplinen, ihre Phasenschwerpunkte, ihre Abhängigkeit vom fachlichen Umfang und die Personalbesetzungskurve berücksichtigen.

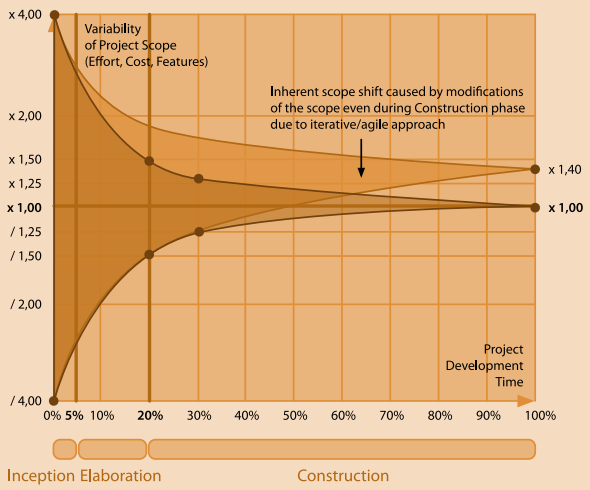
Die sieben sequenziellen Phasen stehen außerdem in keinerlei Konflikt mit agilen Vorgehensmodellen: agile zeitliche Perioden (in Scrum “Sprints” genannt) unterteilen nämlich lediglich die einzelnen Phasen.

Fragen

- Wie wird die Software Engineering **Phase** genannt, welche den größten Personalbedarf besitzt und in der primär die fachlichen Funktionalitäten realisiert werden?

Uncertainty & Elaboration

Cone of Uncertainty



The ***Cone of Uncertainty*** (*Steve McConnell*, 2006) tells how the variability of the project scope (measured in Effort, Cost or Features) in Software Development changes over time. Initially, it usually is within the range of +/- 400% of the final scope.

The early development phases Inception and Elaboration especially have to ensure that within the first 20% of the project, the variability is reduced noticeably to just +/- 50%. During the initial iterations of the Construction phase within the first 30% of the project, the variability usually can be further reduced to about +/- 25%.

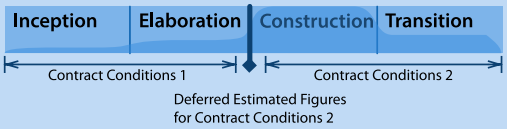
For iterative/agile approaches, experience showed that during the Construction phase inherently the final scope further shifts by about + 40% due to the just step-by-step learned required details of the required solution. This especially has to be taken into account for estimations.

Essential Elaboration Phase

Walking Skeleton:

The ***Walking Skeleton*** (or *Technical Breakthrough*) is the design and implementation of the bare technical foundation of an application, still *without* any domain-specific functionalities. It is made during the Elaboration phase with the primary purpose to establish a stable integration of all technical aspects (libraries, frameworks, build procedures, etc) onto which the domain-specific functionalities later can be successively put onto.

Agile Fixed-Price Contracts:



The ***Agile Fixed-Price*** is an agile variant of a fixed-price contract, *not* a fixed-price project with an agile development process.

There are two important inherent aspects:

First, the contract contains two types of conditions: one (usually *Time & Material* but fixed duration based) for the Inception and Elaboration phases in order to make experiences and to gather necessary figures, and one (usually Fixed-User-Story and/or Fixed-Price based) for the Construction and Transition phases based on deferred estimated figures, gathered in the Elaboration phase.

Second, the Fixed-Price aspect of the contract is actually based on an amount of User-Stories (resulting in costs by multiplying them with either an average hourly rate of an engineer or individual rates based on engineer job levels), which the customer can 1:1 *exchange* during the project for different deliverables.

The crux of an Agile Fixed-Price contract is: first, during the Inception and Elaboration phases the supplier can shrink the *Cone of Uncertainty* and this way its risks dramatically, and second, during the Construction and Transition phases the customer still remains flexible in scope.

Der **Konus der Unsicherheit** zeigt auf, wie sich die Variabilität des Projektumfangs (gemessen in Aufwand, Kosten oder Features) in der Softwareentwicklung im Laufe der Zeit ändert. Die frühen Entwicklungsphasen Inception und Elaboration müssen insbesondere dafür sorgen, daß sich die Variabilität merklich reduziert.

Bei iterativen/agilen Ansätzen zeigt die Erfahrung, daß sich in der Phase Construction naturgemäß der endgültige Umfang weiter verschiebt, weil die erforderlichen Details der gewünschten Lösung erst schrittweise gelernt werden.

Die Phase Elaboration is insbesondere wichtig für die Erstellung des Technischen Durchstichs (**Walking Skeleton**), bei dem alle technischen Integrationen von Libraries, Frameworks, Build-Prozeduren, etc., durchgeführt wird, ohne bereits fachliche Funktionalitäten zu implementieren.

In **Agile Fixed-Price** Projektverträgen wird aufgrund des **Konus der Unsicherheit** in der Regel zwischen den frühen Phasen Inception und Elaboration und den Hauptphasen Construction und Transition unterschieden. Die Vertragsbedingungen der letzteren hängen in der Regel von Zahlen ab, die erst am Ende der Phase Elaboration seriös geschätzt werden können.

Fragen

- Was wird insbesondere in der Projektphase “Elaboration” entwickelt?

Requirements Basics

Requirements Specification

A binding document that specifies the requirements for a solution, by focusing on the WHAT and WHY of the solution — and *not* giving instructions for the HOW.

The documented set of requirements has to be: correct, unambiguous, complete, consistent, ranked, verifiable, modifiable, and traceable.

Requirement Classes

FR Functional (Shall Do)

A condition or capability that a solution must have to provide its service in terms of its behaviour and information. Think: Functionality.

NFR Non-Functional (Shall Be)

A condition, property or quality that a solution must have to satisfy a contract, standard, or other formally imposed obligation. Think: Constraints and "*-ilities".

Requirement Interdependencies

POS Positive (Backing)

One requirement supports the other (e.g. for NFRs: Maintainability and Comprehensibility usually support Adaptability, Portability, Modifiability, etc., and Scalability usually supports Availability, etc.)

NEG Negative (Trade-Off)

One requirement interferes with the other (e.g. for NFRs: Security usually interferes with Efficiency, Usability, Performance, etc., and Orthogonality can interfere with Usability)

Requirement Characteristics

S Specific

The requirement is precise, unambiguous, and clear on what should be done.

M Measurable

The requirement can be verified when it has been achieved by use of a particular test.

A Achievable

The requirement is achievable given existing circumstances and feasible and viable solutions.

R Relevant

The requirement is relevant to the goals of the context.

T Time-Bound

The requirement can be achieved within a reasonable time frame.

Requirement Life-Time

E Enduring

The requirement lasts forever, as it is derived from core activities and organisational structures.

V Volatile

The requirement can be temporary, as it might change over time.

Requirement Expression

[<req-id>] <req-name>:
<subject/actor>
SHALL
<result/action/condition>
BECAUSE
<rationale>

Die **Requirements Specification** (auch Anforderungsspezifikation, Fachkonzept, oder Pflichtenheft) ist ein bindendes Dokument, in dem primär das WAS und das WARUM der Lösung spezifiziert wird, allerdings nicht das konkrete technische WIE. Die Menge an Anforderungen müssen korrekt, eindeutig, vollständig, konsistent, priorisiert, nachprüfbar, änderbar und nachvollziehbar sein.

Es gibt zwei Arten von Requirements: **Functional Requirement** ("Shall Do", Funktionalität) und **Non-Functional Requirements** ("Shall Be", Bedingungen, im Englischen oft über die Wörter mit dem Ende "-ility" ausgedrückt). Der Architekt kümmert sich primär um letztere.

Requirements können außerdem sich gegenseitig positiv (Backing) oder negativ (Trade-Off) beeinflussen. Der Architekt kümmert sich auch hier primär um letztere.

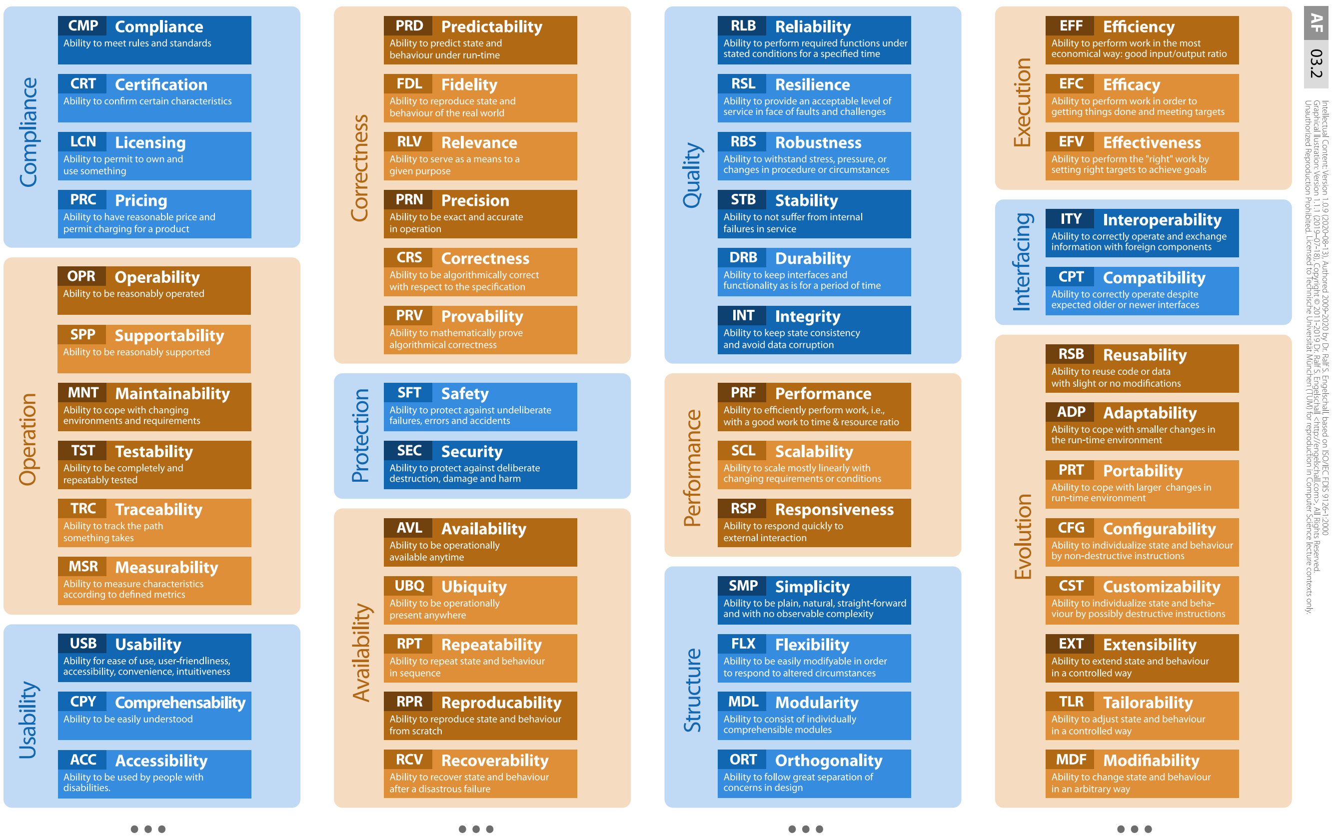
Requirements sollten "SMART" sein: **Specific**, **Measureable**, **Achieveable**, **Relevant** und **Time-Bound**.

Außerdem sind Requirements entweder **Enduring** (feststehend) oder **Volatile** (unbeständig). Der Architekt sollte auf letztere achten.

Fragen

❓ Welche Art von **Requirements** sollte der Architekt primär im Auge haben und sollten von ihm explizit in der Lösungsfindung addressiert werden?

Non-Functional Requirements



Non-Functional Requirements gibt es potenziell sehr viele. Für jede Lösung muss deshalb erst die tatsächliche Menge an solchen Requirements festgestellt werden. Diese Menge ist unbedingt vom Architekten zu minimieren!

Bei jedem vertraglich festgeschriebenen Requirements sollte darauf geachtet werden, daß es klar definiert ist, da es große Ähnlichkeiten zwischen Requirements gibt und die Unterschiede teilweise sehr subtil sind.

Ein paar der in der Praxis fast immer zu berücksichtigenden Non-Functional Requirements sind **Maintainability**, **Usability**, **Security**, **Availability**, **Reliability**, **Performance**, **Responsiveness** und **Adaptability**.

Fragen

? Nennen Sie 3 in der Praxis häufig zu berücksichtigende **Non-Functional Requirements**!