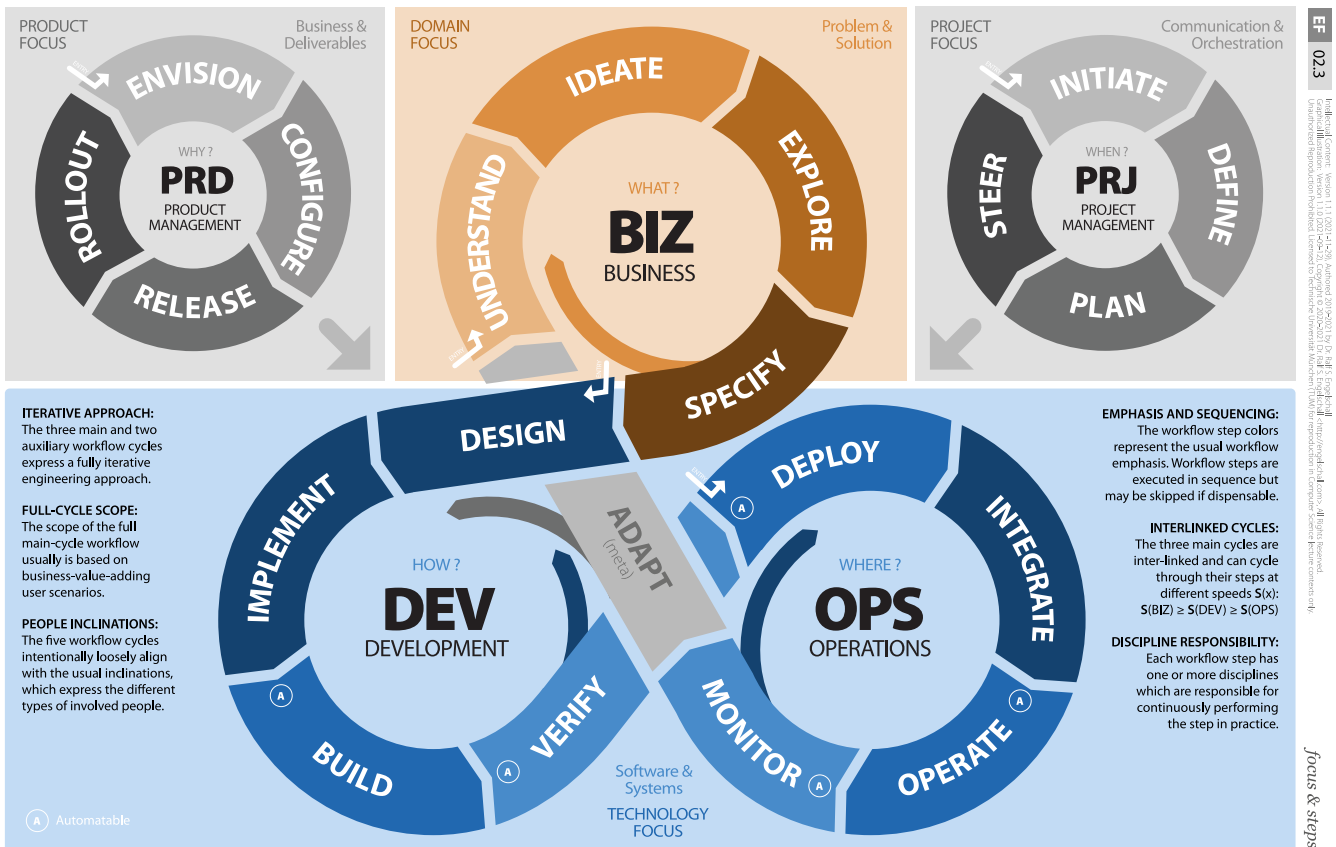




# **Software Engineering in der industriellen Praxis (SEIP)**

**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.

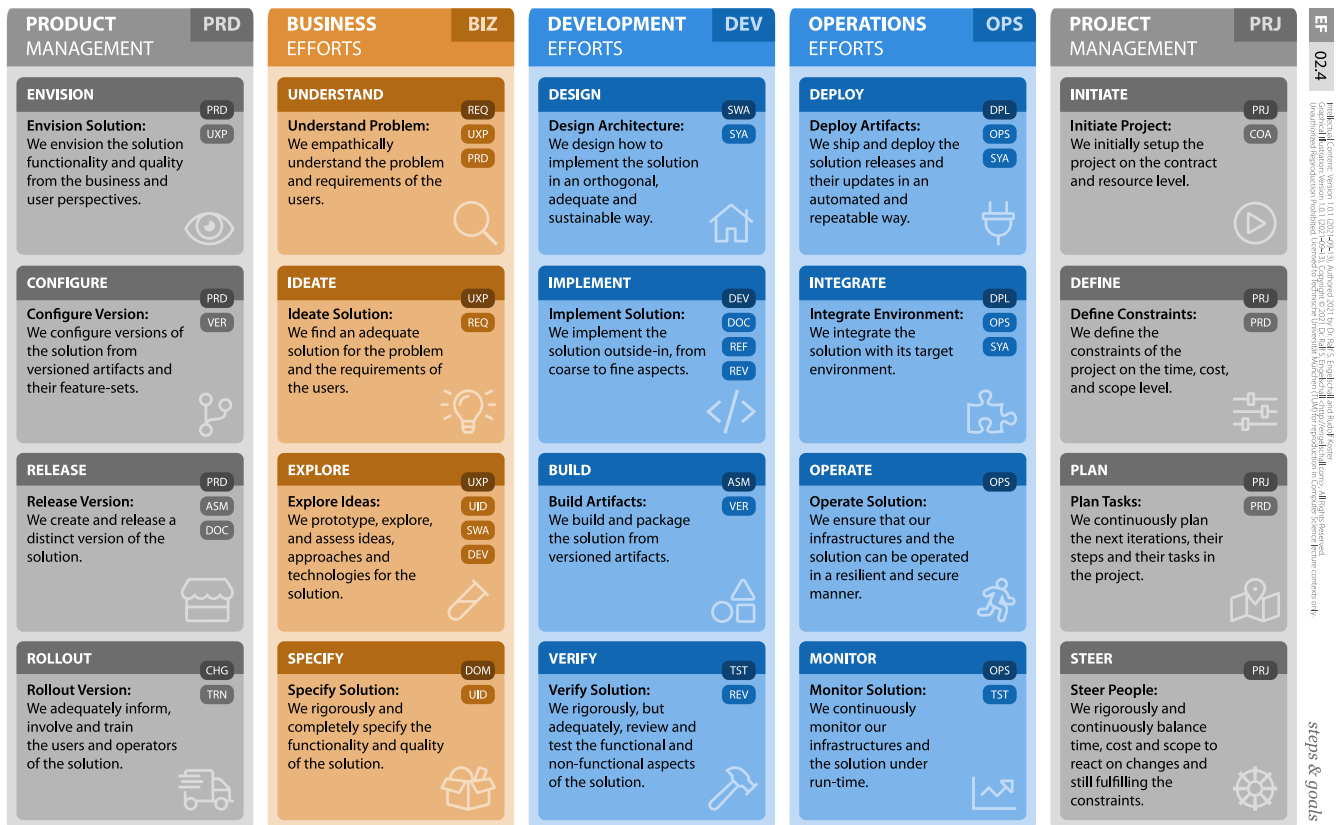
Die drei Hauptzyklen sind miteinander verknüpft und können ihre Schritte mit unterschiedlichen Geschwindigkeiten  $S(x)$  durchlaufen, wobei  $S(BIZ)$  größer oder gleich  $S(DEV)$  ist, welches wiederum größer oder gleich  $S(OPS)$  ist. Denn die Zyklen mit früheren Schritten sollten die Zyklen nicht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

Die Schrittfarben repräsentieren den Arbeitsablaufschiwerpunkt. 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 beschreibt das Workflow-Modell im Software Engineering?



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.

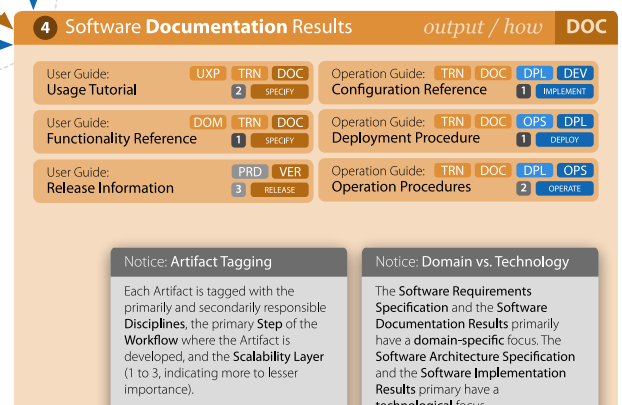
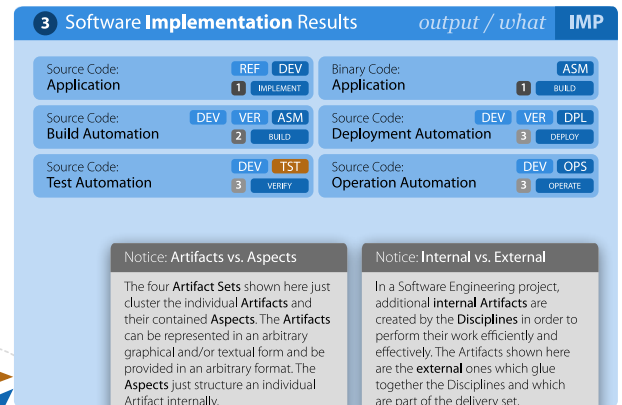
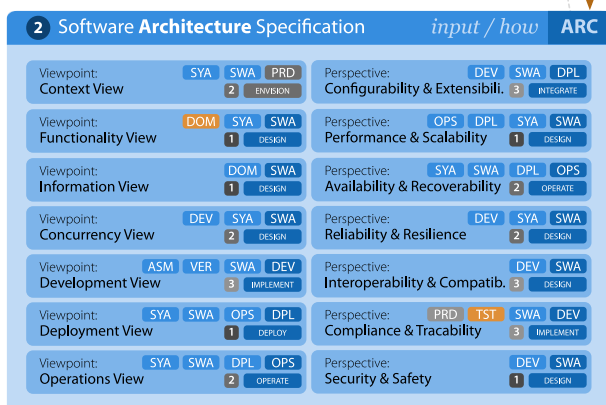
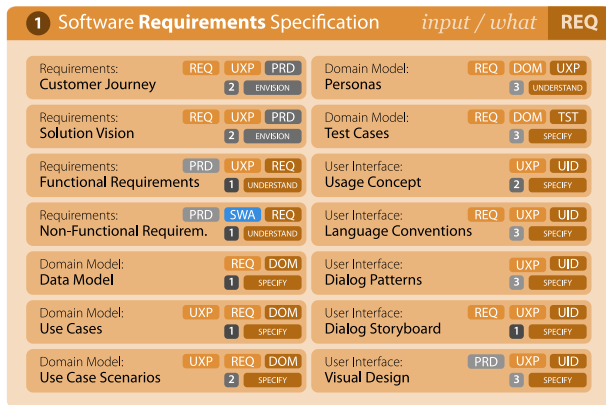
Workflow Steps sind das geeignete Konzept, um zu verstehen, welche Aktivitäten in jeder Iteration eines **Software Engineering Prozesses** durchgeführt werden müssen.

## Fragen

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

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

no/oo 0-400/00



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 **Artifact** 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.

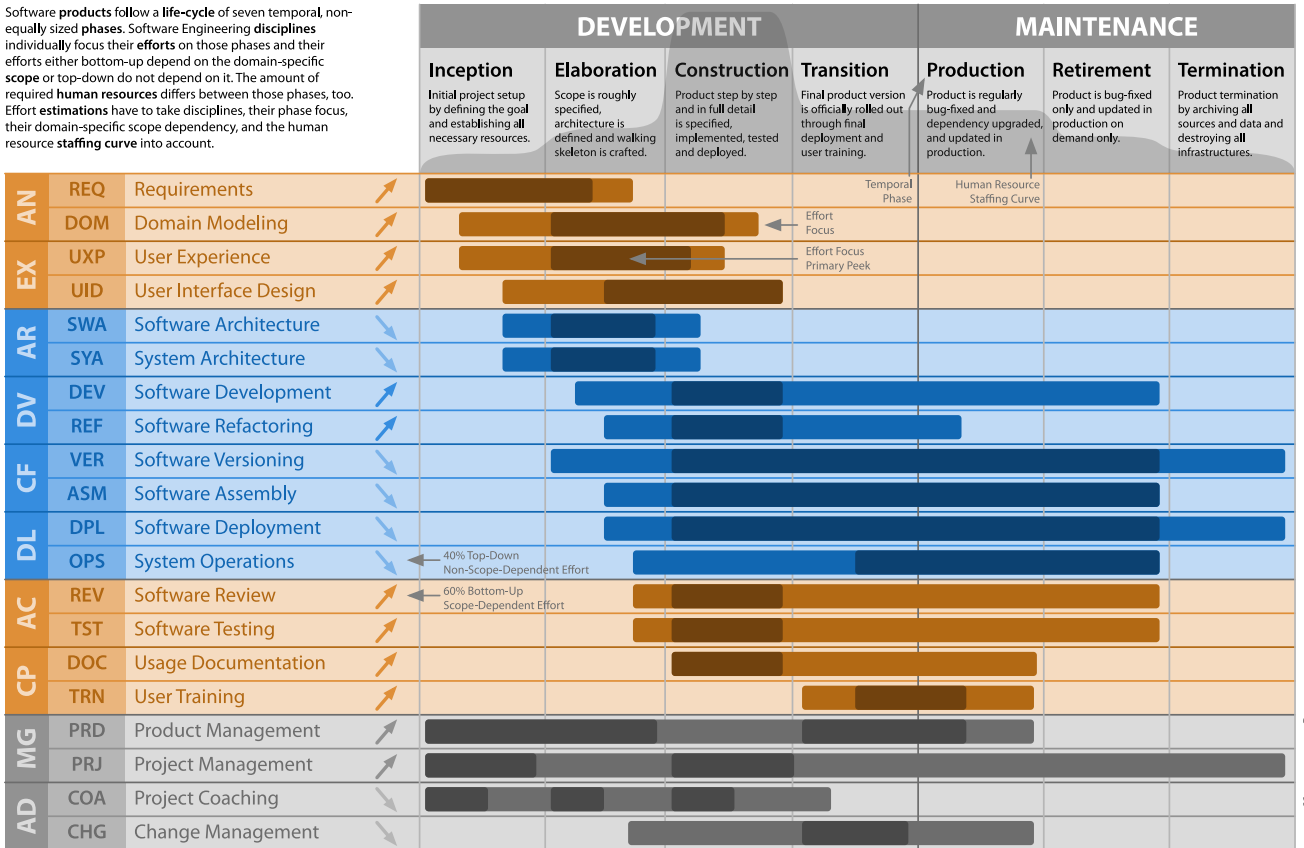
Jedes **Artifact** ist mit der primären und den sekundären **Disciplines**, dem primären **Step** des Workflows in dem das **Artifact** entwickelt wird, und der Skalierbarkeitsstufe (1 bis 3, von mehr bis weniger wichtig) annotiert.

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**?

Software products follow a life-cycle of seven temporal, non-equally sized **phases**. Software Engineering **disciplines** individually focus their **efforts** on those phases and their efforts either bottom-up depend on the domain-specific **scope** or top-down do not depend on it. The amount of required **human resources** differs between those phases, too. **Effort estimations** have to take disciplines, their phase focus, their domain-specific scope dependency, and the human resource **staffing curve** into account.



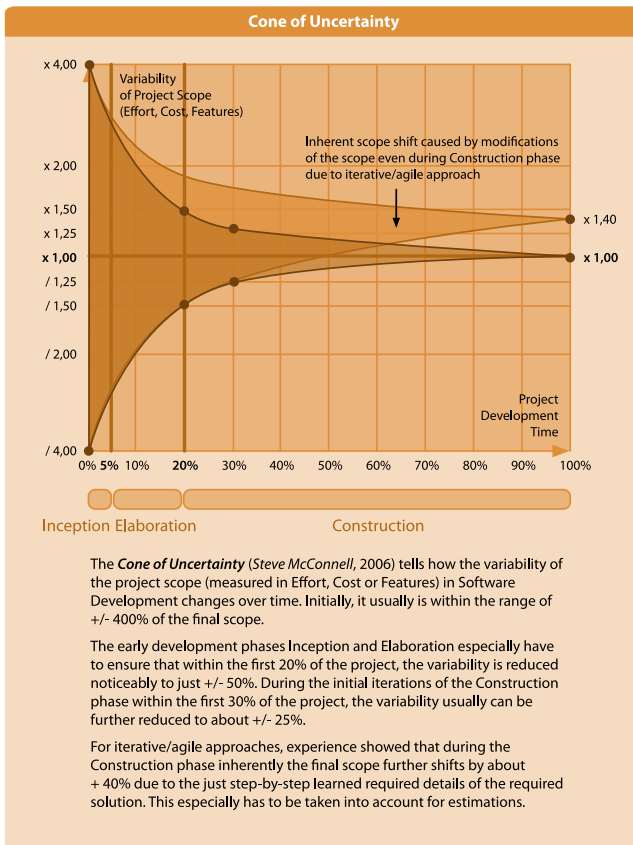
Softwareprodukte folgen einem **Lebenszyklus** von sieben zeitlichen, nicht gleich großen **Phasen**. Die einzelnen **Disziplinen** des Software-Engineering fokussieren ihre **Aufwände** auf diese Phasen und ihre Aufwände sind entweder von unten nach oben vom fachlichen Umfang abhängig, oder von oben 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?



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 ist 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.

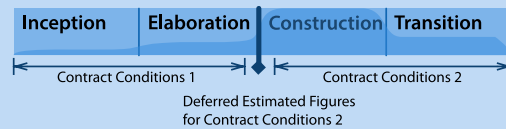
## Essential Elaboration Phase

### 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.

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.

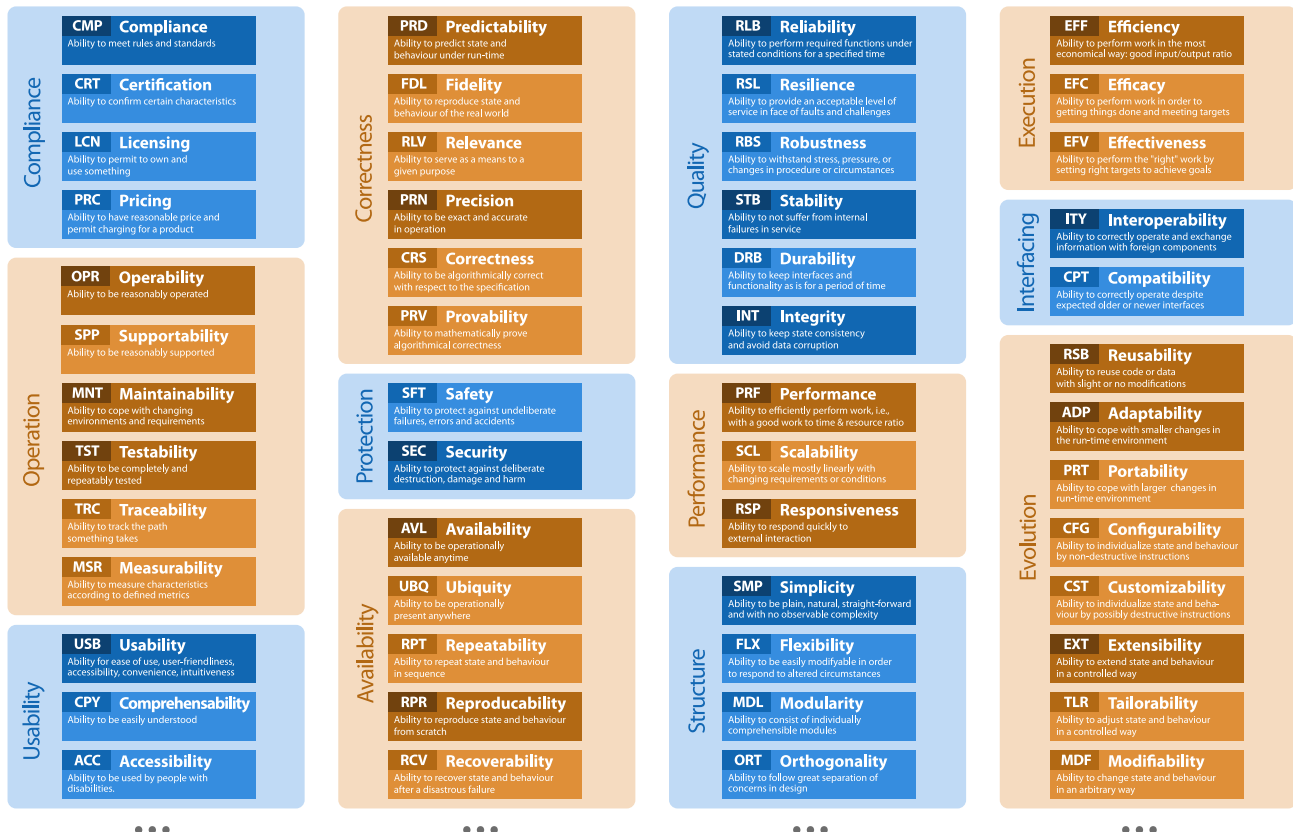
## Fragen

❓ Was wird insbesondere in der Projektphase "Elaboration" entwickelt?



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





**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**!