



Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

Business

Custom Software Development

CSD

Commercial development of non-standardised, **fully individualised**, and **non-reusable company-specific** software for a **single customer**.

Standard Software Development

STD

Commercial development of standardised, **partially customisable**, and **fully reusable domain-specific** software for **many customers**.

Open Source Software Development

OSS

Non-commercial development of standardised, **highly customisable**, and **fully reusable generic** software for **many customers**.

Class: Graphics & Media

target audience: consumers & enterprises

Graphics Editing Application

GEA

Software for editing and rendering graphics in vector and bitmap format.

Examples: Cinema4D, Maya, Blender, After Effects, Illustrator, Inkscape, Scribus, Photoshop, GIMP, etc.

Graphics Animation Engine

GAE

Software for animating the 2D/3D virtual worlds of games and overlays of TV productions.

Examples: Unity, Unreal Engine, CryENGINE, Godot, HUDS, SPX-GC, Holographics, H2R Graphics, etc.

Audio/Video-Processing System

AVS

Software for live-processing and post-production of audio/video based multimedia streams.

Examples: vMix, OBS Studio, VLC, Lossless Cut, Handbrake, Adobe Premiere, FFmpeg, Nimbale, etc.

Class: Business & Data

target audience: consumers & enterprises

Office Productivity Application

OPA

Software for productivity in the desktop-based office environment.

Examples: PowerPoint, Excel, Word, Visio, OmniGraffle, LibreOffice, Outlook, XMind, Firefox, Chrome, etc.

Business Information System

BIS

Software for driving business processes through interactive information management.

Examples: Veeva, CampS, Mission Control, IPW, KEZ-PSC, TimeSheet, SAP ERP, OpenProject, etc.

Data Management System

DMS

Software for protocol-based storing and retrieving of persistent data.

Examples: NextCloud, PostgreSQL, CockroachDB, Redis, InfluxDB, Neo4j, Tendermint, Gitea, Vault, etc.

Class: Machinery & Network

target audience: consumers & enterprises

Technical Control System

TCS

Software for controlling a physical machinery or technical system.

Examples: AquaTherm, AVM! FritzBox Firmware, BirdDog Camera Firmware, etc.

Network Communication System

NCS

Software for protocol-based communication of data over a computer network.

Examples: Apache, NGINX, HAProxy, Mosquitto, RabbitMQ, Node-RED, KeyCloak, etc.

Operating System Kernel

OSK

Software kernel for low-level operating a physical or virtual device and run programs on it.

Examples: Windows, macOS, iOS, Linux, FreeBSD, QNX, ChibiOS/RT, Kubernetes, Wildfly, etc.

Class: Development & Tools

target audience: vendors & suppliers

Software Development Kit

SDK

Software libraries and frameworks of reusable functionality for developing software.

Examples: NDI SDK, HAPI, GraphQL-IO, Sequelize, JDK, Spring, Hibernate, etc.

Software Development Tools

SDT

Software tools for editing, linting, compiling, packaging, distributing, and installing software.

Examples: Visual Studio Code, Sublime Text, GCC, GNU Binutils, NPM, JDK, Docker, Helm, etc.

Operating System Tools

OST

Es gibt drei traditionelle Ansätze der Software-Entwicklung: **Custom Software Development**, die kommerzielle Entwicklung von nicht-standardisierter, vollständig individualisierter und nicht wiederverwendbarer firmenspezifischer Software für einen einzelnen Kunden; **Standard Software Development**, die kommerzielle Entwicklung einer standardisierten, teilweise individualisierbaren, aber vollständig wiederverwendbaren fachlichen Software für viele Kunden; und **Open-Source-Software Development**, die nicht-kommerzielle Entwicklung einer standardisierter, hochgradig anpassbarer und vollständig wiederverwendbarer technischen Software für viele Kunden.

Es gibt vier Bereiche und 12 Klassen von Software. Im ersten Bereich gibt es drei Klassen von Software, die sich auf Grafik & Medien fokussieren: **Graphics Editing Application**, Software zum Bearbeiten und Rendern von Grafiken im Vektor- und Bitmap-Format; **Graphics Animation Engine**, Software für die Animation der virtuellen 2D/3D-Welten von Spielen und der Overlays von TV-Produktionen; und **Audio/Video-Processing Systems**, Software für die Live-Bearbeitung und Nachbearbeitung von Audio/Video-Multimedia-Streams.

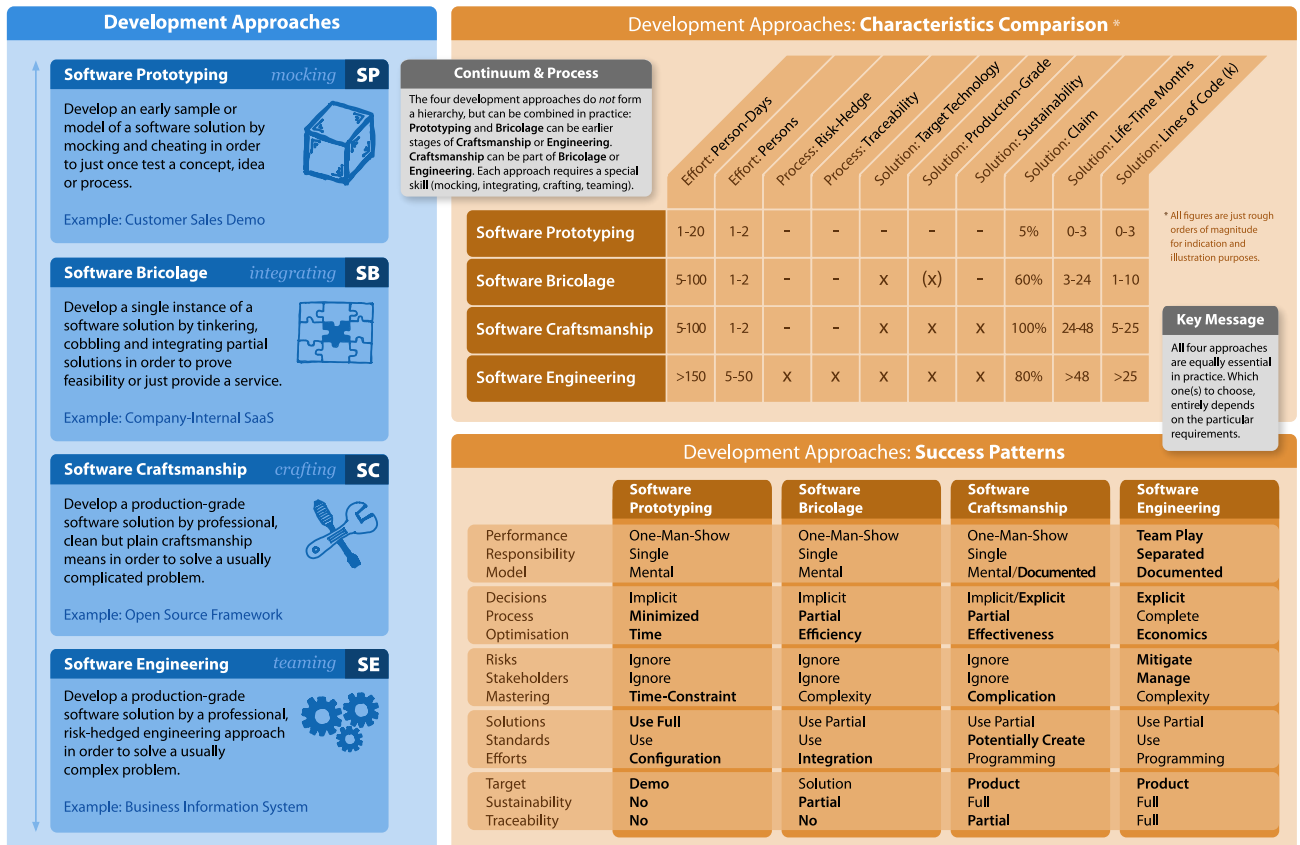
Im zweiten Bereich gibt es drei Klassen von Software, die sich auf Geschäft & Daten fokussieren: **Office Productivity Application**, eine Software für Produktivität in der Desktop-basierten Büroumgebung; **Business Information System**, eine Software zur Steuerung von Geschäftsprozessen mittels Informationsmanagement; und **Data Management System**, eine Software zum Speichern und Abrufen von persistenten Daten.

Im dritten Bereich gibt es drei Klassen von Software, die sich auf Maschinen & Netzwerk fokussieren: **Technical Control System**, eine Software zur Steuerung einer physischen Maschine oder eines technischen Systems; **Network Communication System**, eine Software für die Kommunikation von Daten über ein Computer-Netzwerk; und **Operating System Kernel**, ein Software-Kern für den Betrieb eines physischen oder virtuellen Computers auf einer niedrigeren Ebene.

Im vierten Bereich gibt es drei Klassen von Software, die sich auf Entwicklung & Werkzeuge fokussieren: **Software Development Kit**, Software-Bibliotheken und -Frameworks mit wiederverwendbarer Funktionalität für die Entwicklung von Software; **Software Development Tools**, Software-Werkzeuge zum Bearbeiten, Analysieren, Übersetzen, Paketieren und Installieren von Software; und **Operating System Tools**, Software-Werkzeuge für den Betrieb eines physischen oder virtuellen Computers.

Fragen

- Welche beiden Klassen von Software werden vor allem mit Hilfe des Ansatzes **Custom Software Development** entwickelt?



Man kann vier Arten von Software-Entwicklungs-Ansätzen unterscheiden.

Bei **Software Prototyping** entwickelt man ein frühes Beispiel oder Modell einer Softwarelösung durch "Mocking" und "Cheating", um ein Konzept, eine Idee oder einen Prozess einmalig zu testen.

Bei **Software Bricolage** entwickelt man eine einzelne Instanz einer Softwarelösung durch Basteln, Zusammenschustern und Integrieren von Teillösungen, um eine Machbarkeit zu beweisen oder einfach nur eine Dienstleistung zu erbringen.

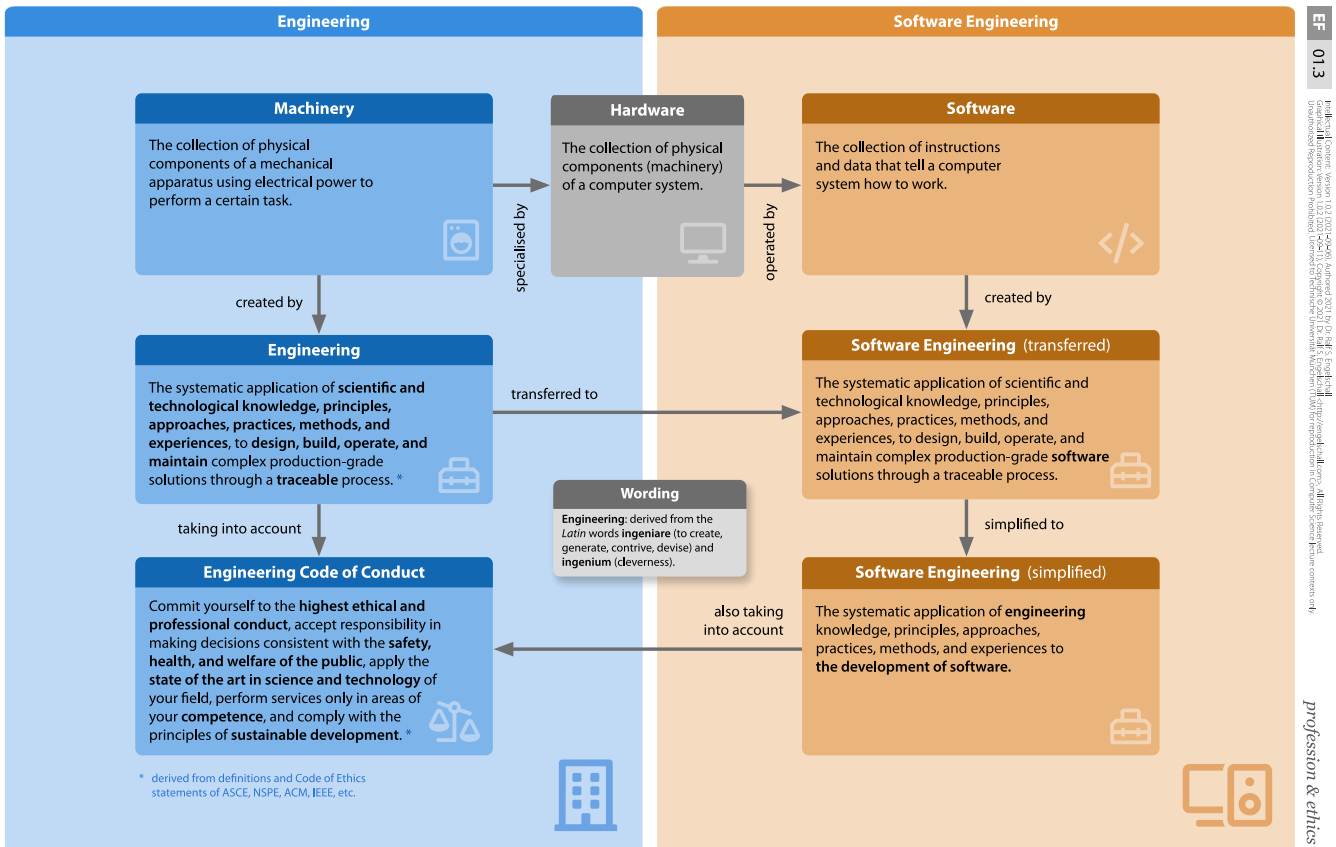
Bei **Software Craftsmanship** entwickelt man eine produktionsreife Softwarelösung mit professionellen, sauberen, aber reinen handwerklichen Mitteln, um ein meist kompliziertes Problem zu lösen.

Bei **Software Engineering** entwickelt man eine produktionsreife Softwarelösung mit einem professionellen, risikoabsichernden Ingenieurs-Ansatz, um ein in der Regel komplexes Problem zu lösen.

Die vier Entwicklungsansätze können in der Praxis auch kombiniert werden: Prototyping und Bricolage können Vorstufen von Craftsmanship oder Engineering sein. Craftsmanship kann Teil des Engineering sein. Jeder Ansatz erfordert eine spezielle Fähigkeit. Alle vier Ansätze sind in der Praxis gleichermaßen wichtig. Welche(s) man jeweils wählt, hängt ganz von den konkreten Anforderungen ab.

Fragen

- Welchen Software-Entwicklungs-Ansatz sollte man wählen, um ein komplexes betriebliches Informationssystem zu realisieren?
- Welchen Software-Entwicklungs-Ansatz sollte man wählen, um eine komplizierte wiederverwendbare Bibliothek zu realisieren?



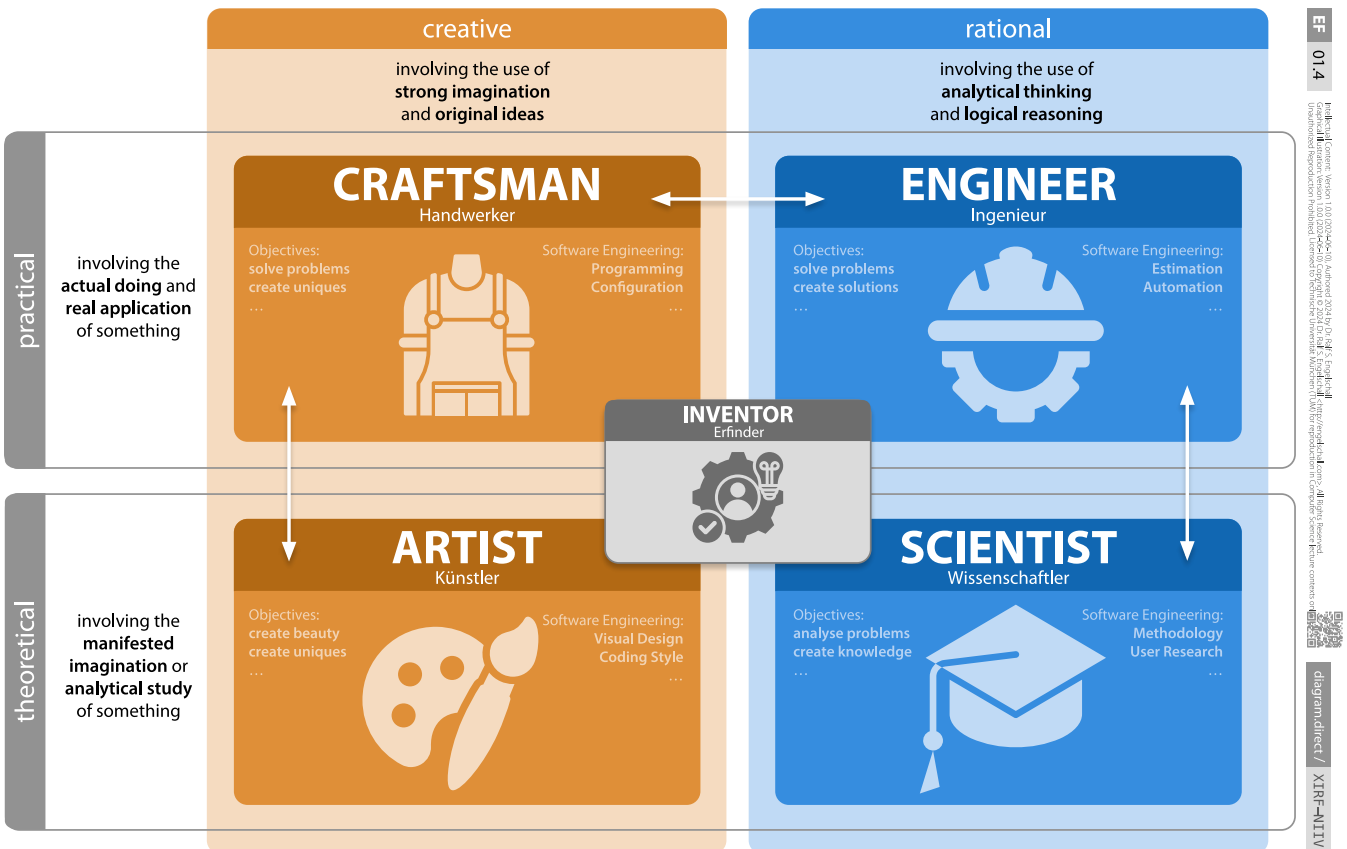
Ingenieurwesen (Engineering) ist die systematische Anwendung von wissenschaftlichen und technologischen Wissen, Prinzipien, Ansätze, Praktiken, Methoden und Erfahrungen, um komplexe, produktionsreife Lösungen durch einen nachvollziehbaren Prozess zu entwerfen, zu bauen, zu betreiben und zu warten.

Software Engineering ist die systematische Anwendung von ingenieurwissenschaftlichen Wissen, Prinzipien, Ansätzen, Praktiken, Methoden und Erfahrungen auf die Entwicklung von Software.

Sowohl für das Engineering als auch für das Software Engineering gilt der folgende **Verhaltenskodex** (Code of Conduct): Verpflichten Sie sich zu höchstem ethischen und professionellen Verhalten; übernehmen Sie Verantwortung Entscheidungen im Einklang mit der Sicherheit, der Gesundheit und dem Wohlergehen der Öffentlichkeit zu fällen; wenden Sie den Stand der Wissenschaft und Technik ihres Fachgebiets an; erbringen Sie Leistungen nur in Bereichen Ihrer eigenen Kompetenz; und beachten Sie die Prinzipien der nachhaltigen Entwicklung.

Fragen

- ❓ Ist Software Engineering auch für die Entwicklung einer nicht-komplexen Software in einem kleinen Team von zwei Personen sinnvoll?












Berufe haben in der Regel zwei von vier Merkmalen: **Kreativ** zu sein bedeutet, daß man eine starke Vorstellungskraft und originelle Ideen nutzt. **Rational** zu sein bedeutet, daß man analytisches Denken und logisches Schlussfolgern anwendet. **Praktisch** zu sein bedeutet, daß man etwas tatsächlich tut und anwendet. **Theoretisch** zu sein bedeutet, daß man eine Vorstellung einer Sache umsetzt oder eine Sache analytisch untersucht.

Man kann fünf interessante Berufe unterscheiden: Ein **Handwerker** handelt auf kreative und praktische Weise und löst Probleme und schafft Unikate. Ein **Ingenieur** handelt rational und praktisch und löst Probleme und schafft Lösungen. Ein **Künstler** handelt kreativ und theoretisch und erschafft Schönheit und Unikate. Ein **Wissenschaftler** handelt rational und theoretisch und analysiert Probleme und schafft Wissen. Ein **Erfinder** hingegen muss normalerweise alle Eigenschaften in sich vereinen.

Fragen

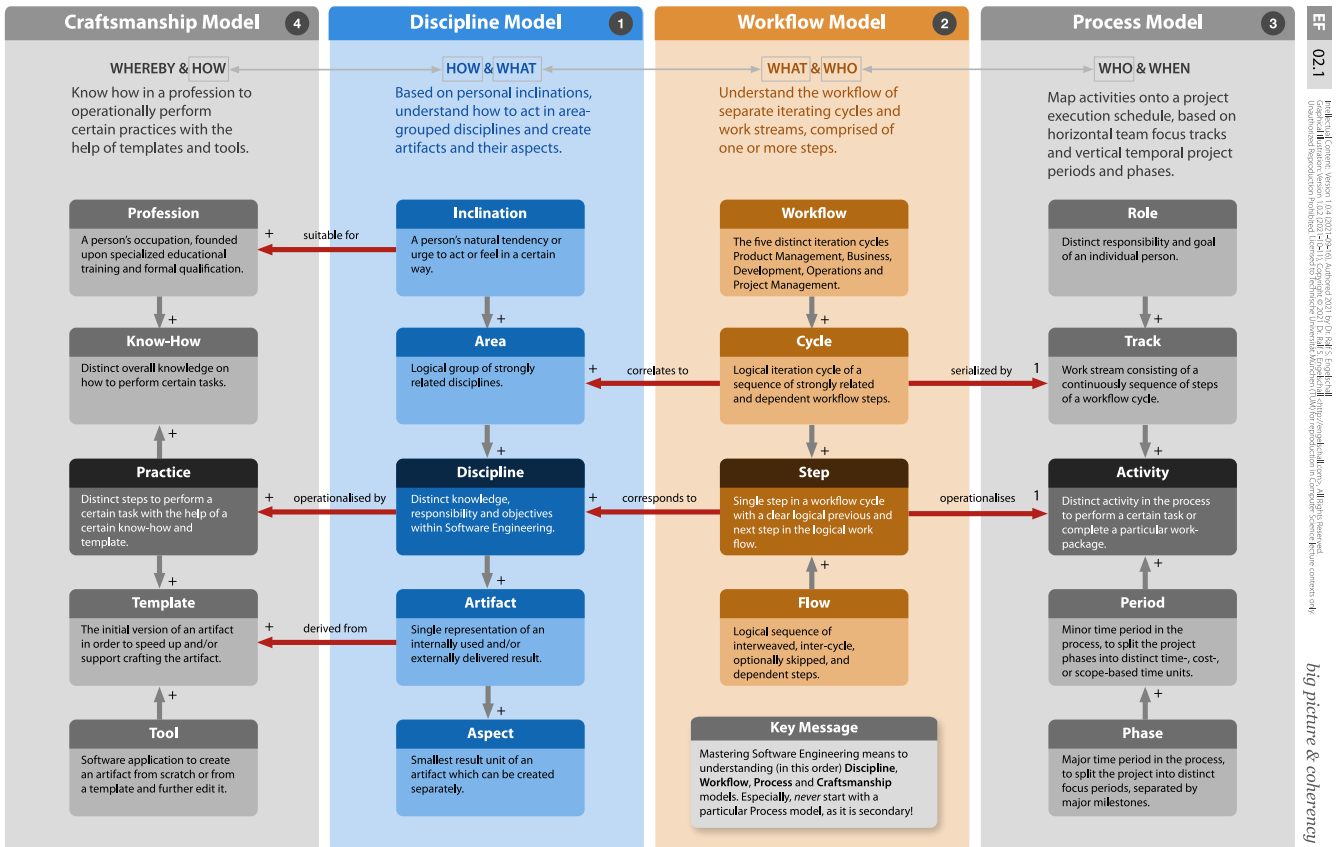
- Wenn man sich um Konfiguration und Programmierung im Software Engineering kümmert, agiert man statt als Ingenieur eher wie ein...?

<p>FA FARSIGHTED weitblickend</p> <p>Be farsighted in your solution finding.</p> <p>Sei weitblickend in deiner Lösungsfindung.</p> <p>AR: Scalable Hub'n'Spoke DV: Plugin SPI</p> 	<p>TE TENET-ORIENTED grundsatzorientiert</p> <p>Orientate yourself on fixed tenets in your approach and solution finding.</p> <p>Orientiere dich an festen Grundsätzen in deinem Vorgehen und deiner Lösungsfindung.</p> <p>AR: Separation of Concern DV: Strict Coding-Style</p> 	<p>TH THOUGHTFUL wohlüberlegt</p> <p>Act thoughtful in your approach and solution finding.</p> <p>Agiere wohlüberlegt in deinem Vorgehen und deiner Lösungsfindung.</p> <p>AR: Modularization DV: Algorithmical Control Structure</p> 
<p>HO HOLISTICALLY ganzheitlich</p> <p>Think holistically and in the long-term when finding your solutions.</p> <p>Denke ganzheitlich und langfristig in deiner Lösungsfindung.</p> <p>AR: Walking Skeleton Design DV: Consistent Error Handling</p> 	<p>AD ADEQUATE angemessen</p> <p>Ensure that your approach and solutions are adequate to the boundary conditions.</p> <p>Sorge dafür, daß dein Vorgehen und deine Lösungen angemessen zu den Rahmenbedingungen sind.</p> <p>AR: No Cloud-Native Complexity DV: No Over-Engineered Abstractions</p> 	<p>FE FEASIBLE machbar</p> <p>Ensure that your approach and solutions can be realised at reasonable costs.</p> <p>Sorge dafür, daß dein Vorgehen und deine Lösungen mit vernünftigen Kosten realisiert werden können.</p> <p>AR: Existing Framework Functionality DV: Realistic Programming Model</p> 
<p>IN INCREMENTAL inkrementell</p> <p>Apply the depth of your discipline incrementally.</p> <p>Wende die Tiefe deiner Disziplin inkrementell an.</p> <p>AR: Identified Solution Cruxes DV: Minimum Viable Product</p> 	<p>VA VALUEABLE wertvoll</p> <p>Provide clearly recognizable added values with your approach and solutions.</p> <p>Liefere klar ersichtliche Mehrwerte mit deinem Vorgehen und deinen Lösungen.</p> <p>AR: Technology Stack Design DV: User-Story-Driven Functionality</p> 	<p>SU SUSTAINABLE nachhaltig</p> <p>Create sustainable solutions that are well integrated into their environment.</p> <p>Erschaffe nachhaltige Lösungen, die gut in ihre Umgebung integriert sind.</p> <p>AR: Interoperable Interfaces DV: Maintainable Code</p> 

Über die verschiedenen **Disziplinen** des Software-Engineering hinweg, gibt es einige gemeinsame Eigenschaften, die sie alle vom konzeptionellen Standpunkt aus beanspruchen: **weitblickend, grundsatzorientiert, wohlüberlegt, ganzheitlich, angemessen, machbar, inkrementell, wertvoll und nachhaltig.**

Fragen

- ❓ Was ist der schwierigste Anspruch in einer Disziplin in der heutigen Praxis?



Software Engineering kann durch ein Meta-Modell verstanden werden, das auf vier verschiedenen, aber miteinander verknüpften, Modellen basiert.

Das Handwerks-Modell (**Craftsmanship Model**) ist die Basis und zielt auf das WOMIT & WIE. Es spannt den Bogen von den Berufen (**Profession**) der einzelnen Personen, deren entsprechenden **Know-Hows** und Praktiken (**Practice**) bis hin zu den zugrunde liegenden Vorlagen (**Templates**) und Werkzeugen (**Tools**).

Das Disziplinen-Modell (**Discipline Model**) zielt auf das WIE & WAS. Es trennt Software Engineering in Disziplinen (**Disciplines**), die in Bereiche (**Area**) gruppiert sind und die durch die üblichen Neigungen (**Inclination**) von einzelnen Personen motiviert sind. Jede Disziplin wird dann durch Eingabe- und Ausgabe-Artefakte (**Artifact**) und deren Aspekte (**Aspect**) beschrieben.

Das Arbeitsablauf-Modell (**Workflow Model**) zielt auf das WAS & WER. Es beschreibt einen Arbeitsablauf (**Workflow**) aus Zyklen (**Cycle**), die Schritte (**Step**) enthalten. Ein Fluss (**Flow**) ist der Durchlauf durch diese Schritte über die Zeit.

Das Prozess-Modell (**Process Model**) schließlich zielt auf das WER & WANN. Es bildet Aktivitäten (**Activity**) auf einen Projektablaufplan ab, basierend auf horizontalen Bahnen (**Track**) von Rollen (**Roles**) und vertikalen Perioden (**Period**) von Phasen (**Phase**).

Fragen

- Wieviele Zyklen kennt man im Workflow Model des Software Engineering, in denen jeweils die Personen mit ähnlichen Neigungen agieren?



Software Engineering lässt sich durch 20 verschiedene Disziplinen (**Discipline**) (operationalisiert durch Input- und Output-Artefakte und deren Aspekte), die logisch in 10 verschiedene Bereiche (**Area**) gruppiert sind, und die wiederum logisch in 5 verschiedene Neigungen (**Inclination**) von einzelnen Personen gruppiert sind, verstehen.

Personen mit einer starken fachlichen und geschäftlichen (**domain-specific** and **business-oriented**) Neigung agieren in den Bereichen Analyse (**Analysis**) und Erfahrung (**Experience**) und in den entsprechenden Disziplinen Software-Anforderungen (**Software Requirements**), Fachmodellierung (**Domain Modeling**), Benutzererfahrung (**User Experience**) und Benutzungsschnittstelle (**User Interface**).

Personen mit einer starken konstruktiven und technologischen (**constructive** and **technological**) Neigung agieren in den Bereichen Architektur (**Architecture**) und Entwicklung (**Development**) und in den entsprechenden Disziplinen Software-Architektur (**Software Architecture**), System-Architektur (**System Architecture**), Software-Entwicklung (**Software Development**) und Software-Überarbeitung (**Software Refactoring**).

Personen mit einer starken infrastrukturellen und technologischen (**infrastructural** and **technological**) Neigung agieren in den Bereichen Konfiguration (**Configuration**) und Auslieferung (**Delivery**) und in den entsprechenden Disziplinen Software-Versionierung (**Software Versioning**), Software-Montage (**Software Assembly**), Software-Bereitstellung (**Software Deployment**) und Software-Betrieb (**Software Operations**).

Personen mit einer starken analytischen und fachlichen (**analytical** and **domain-specific**) Neigung agieren in den Bereichen Analytik (**Analytics**) und Verständnis (**Comprehension**) und in den entsprechenden Disziplinen Software-Überprüfung (**Software Reviewing**), Software-Test (**Software Testing**), Benutzungsdokumentation (**User Documentation**) und Benutzerschulung (**User Training**).

Personen mit einer starken personellen und prozessorientierten (**people-oriented** and **process-oriented**) Neigung agieren in den Bereichen Führung (**Management**) und Anpassung (**Adjustment**) und in den entsprechenden Disziplinen Projektmanagement (**Project Management**), Projektauditorium (**Project Auditing**), Projektcoaching (**Project Coaching**) und Veränderungsmanagement (**Change Management**).

Fragen

- ❓ Welche Disziplinen im Software Engineering werden als die beiden **Königdisziplinen** angesehen?