



Software Engineering in Industrial Practice (SEIP)

Dr. Ralf S. Engelschall

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.



STD

OSS

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.

CSD

STD

OSS

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, Nimble, etc.



STD

OSS

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.



STD

OSS

Business Information System **BIS**

Software for driving business processes through interactive information management.



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

CSD

STD

OSS

Data Management System **DMS**

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



Examples: NextCloud, PostgreSQL, CockroachDB, Redis, InfluxDB, Neo4J, Tendermind, Gitea, Vault, etc.



STD

OSS

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.

CSD

STD

OSS

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.



STD

OSS

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.



STD

OSS

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.



STD

OSS

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.



STD

OSS

Operating System Tools **OST**

Software tools for high-level operating a physical or virtual computing device.



Examples: Coreutils, Bash, Vim, TMux, FZF, cURL, RSYNC, OpenSSH, etc.



STD

OSS

Development Approaches

Software Prototyping *mocking* **SP**

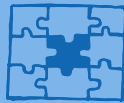
Develop an early sample or model of a software solution by mocking and cheating in order to just once test a concept, idea or process.



Example: Customer Sales Demo

Software Bricolage *integrating* **SB**

Develop a single instance of a software solution by tinkering, cobbling and integrating partial solutions in order to prove feasibility or just provide a service.



Example: Company-Internal SaaS

Software Craftsmanship *crafting* **SC**

Develop a production-grade software solution by professional, clean but plain craftsmanship means in order to solve a usually complicated problem.



Example: Open Source Framework

Software Engineering *teaming* **SE**

Develop a production-grade software solution by a professional, risk-hedged engineering approach in order to solve a usually complex problem.



Example: Business Information System

Continuum & Process

The four development approaches do *not* form a hierarchy, but can be combined in practice: **Prototyping** and **Bricolage** can be earlier stages of **Craftsmanship** or **Engineering**. **Craftsmanship** can be part of **Bricolage** or **Engineering**. Each approach requires a special skill (mocking, integrating, crafting, teaming).

Development Approaches: Characteristics Comparison *

	Effort: Person-Days	Effort: Persons	Process: Risk-Hedge	Process: Traceability	Solution: Target Technology	Solution: Production-Grade	Solution: Sustainability	Solution: Claim	Solution: Life-Time Months	Solution: Lines of Code (k)
Software Prototyping	1-20	1-2	-	-	-	-	5%	0-3	0-3	
Software Bricolage	5-100	1-2	-	-	X	(X)	60%	3-24	1-10	
Software Craftsmanship	5-100	1-2	-	-	X	X	100%	24-48	5-25	
Software Engineering	>150	5-50	X	X	X	X	80%	>48	>25	

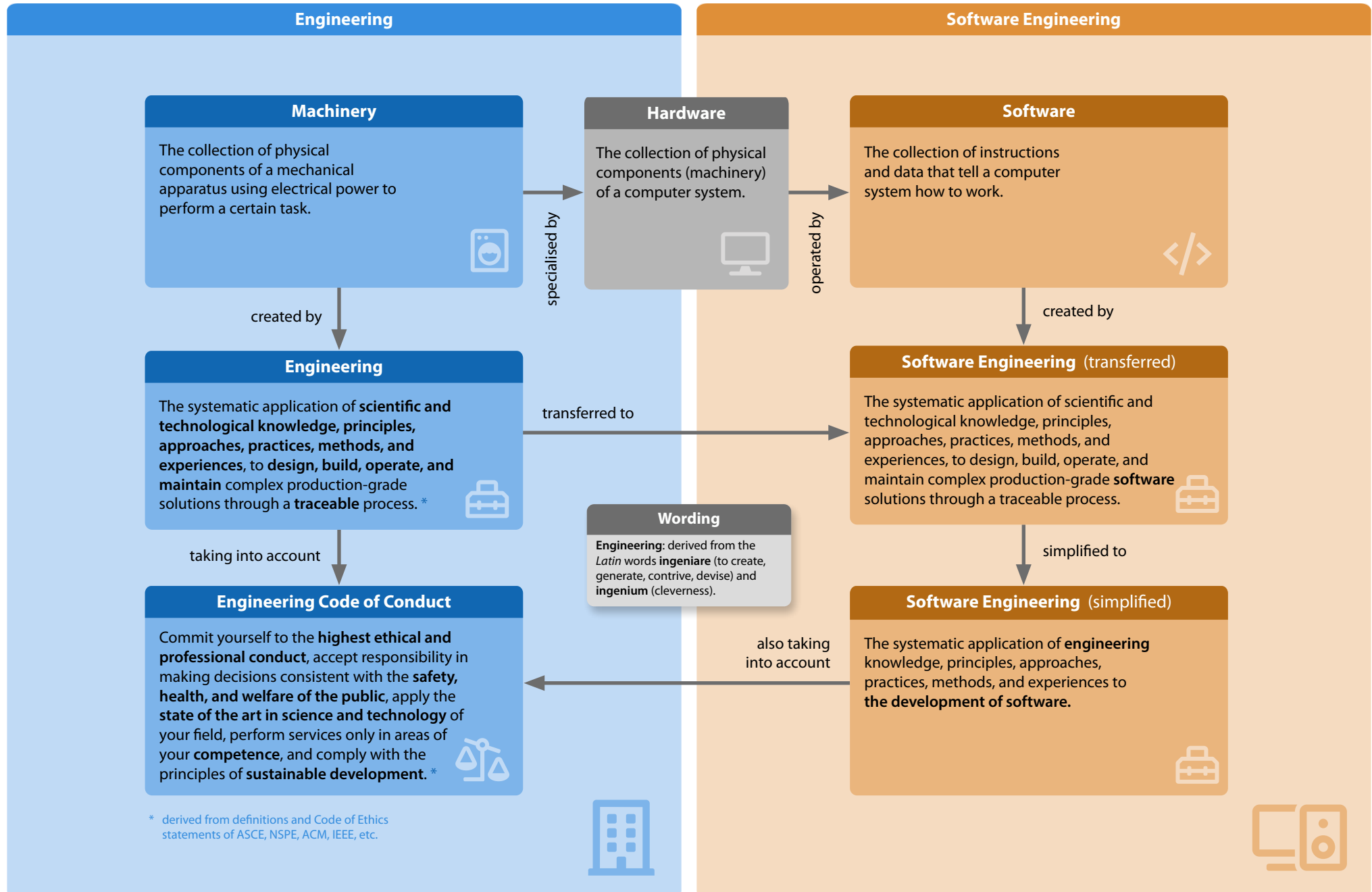
* All figures are just rough orders of magnitude for indication and illustration purposes.

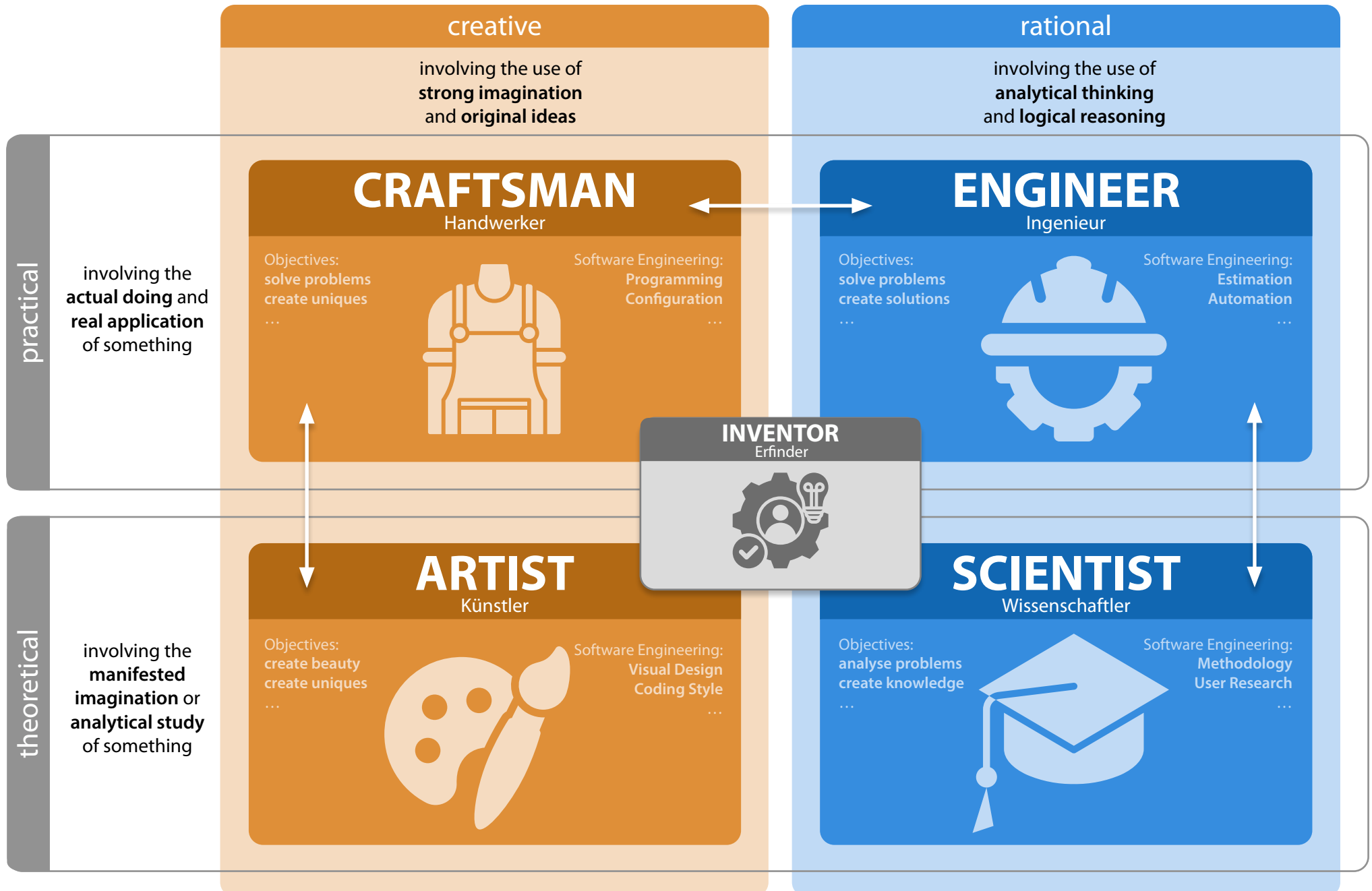
Key Message

All four approaches are equally essential in practice. Which one(s) to choose, entirely depends on the particular requirements.

Development Approaches: Success Patterns


	Software Prototyping	Software Bricolage	Software Craftsmanship	Software Engineering
Performance Responsibility Model	One-Man-Show Single Mental	One-Man-Show Single Mental	One-Man-Show Single Mental/ Documented	Team Play Separated Documented
Decisions Process Optimisation	Implicit Minimized Time	Implicit Partial Efficiency	Implicit/Explicit Partial Effectiveness	Explicit Complete Economics
Risks Stakeholders Mastering	Ignore Ignore Time-Constraint	Ignore Ignore Complexity	Ignore Ignore Complication	Mitigate Manage Complexity
Solutions Standards Efforts	Use Full Use Configuration	Use Partial Use Integration	Use Partial Potentially Create Programming	Use Partial Use Programming
Target Sustainability Traceability	Demo No No	Solution Partial No	Product Full Partial	Product Full Full







FA	FARSIGHTED weitblickend
<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>	

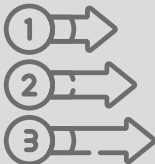
TE	TENET-ORIENTED grundsatzorientiert
<p>Orientate yourself on fixed tenets in your approach and solution finding.</p> <p>Orientiere dich an festen Grundsätzen in deinem Vorgehen und deiner Lösungsfindung.</p> <p>AR: Separation of Concern DV: Strict Coding-Style</p>	

TH	THOUGHTFUL wohlüberlegt
<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>	

HO	HOLISTICALLY ganzheitlich
<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>	

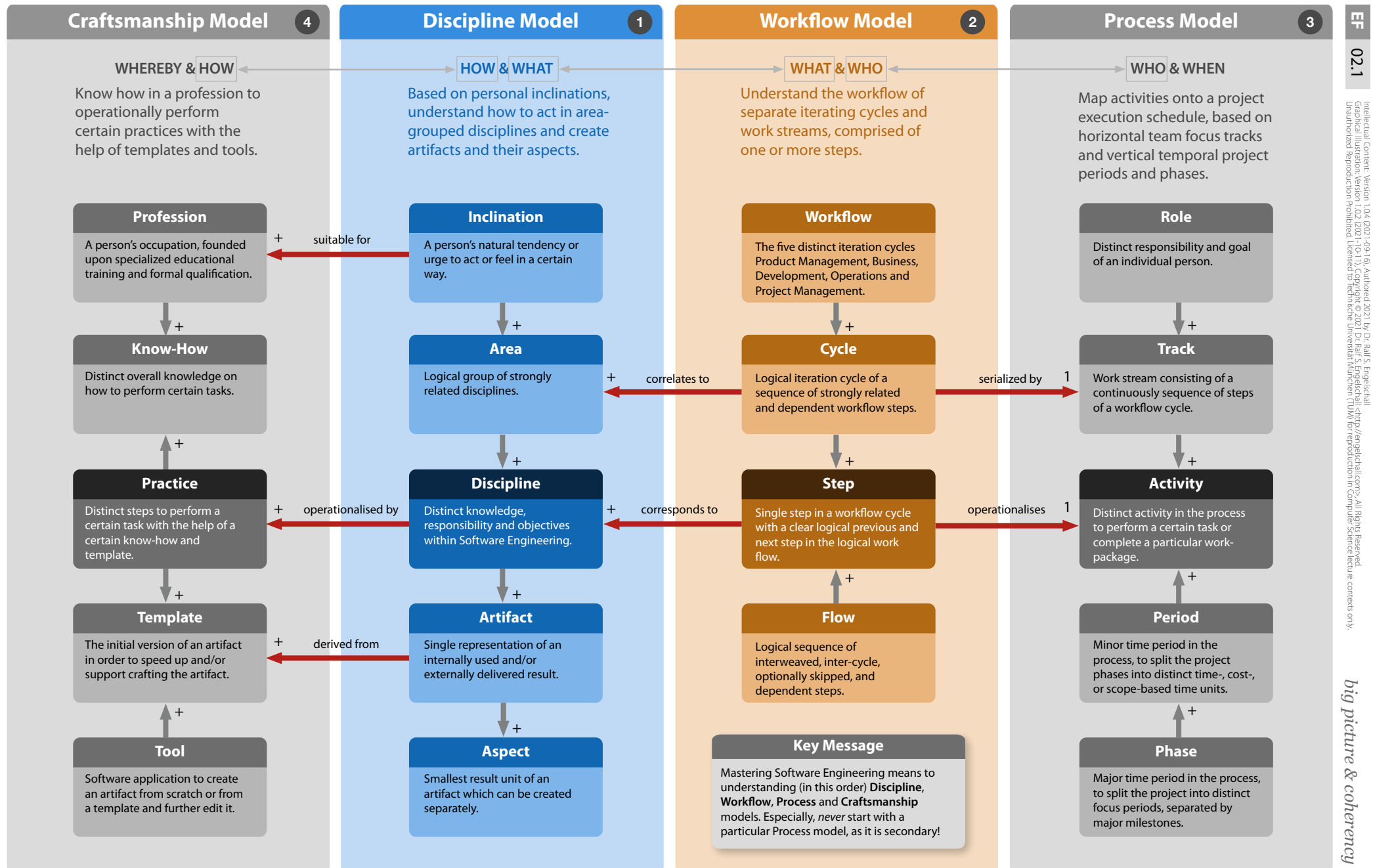
AD	ADEQUATE angemessen
<p>Ensure that your approach and solutions are adequate to the boundary conditions.</p> <p>Sorge dafür, daß dein Vorgehen und deine Lösungen angemessen zu den Rahmenbedingungen sind.</p> <p>AR: No Cloud-Native Complexity DV: No Over-Engineered Abstractions</p>	

FE	FEASIBLE machbar
<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>	

IN	INCREMENTAL inkrementell
<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>	

VA	VALUEABLE wertvoll
<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>	

SU	SUSTAINABLE nachhaltig
<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>	





WB white-box view (details before whole)

BB black-box view (whole before details)

X scalability layer (from 4/most to 1/least dispensable)