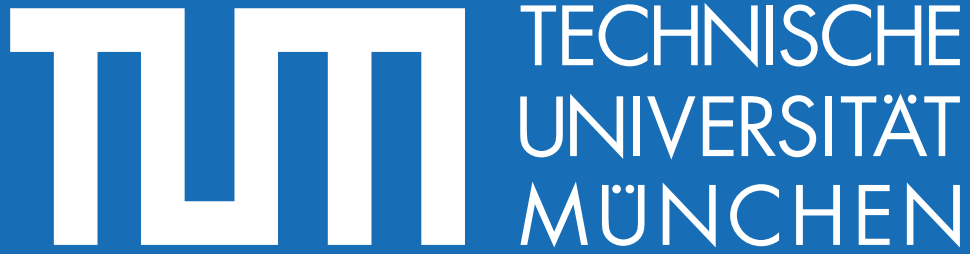


Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

ENGINEERING FUNDAMENTALS

Software Classes

TECHNISCHE UNIVERSITÄT MÜNCHEN

EF 01.1

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 a **class** of customers.

Open Source Software Development — OSS

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

~~CSD~~ 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.

~~CSD~~ 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, Outlook, XMind, Firefox, Chrome, etc.

~~CSD~~ 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.

~~CSD~~ 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.

~~CSD~~ ~~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.

~~CSD~~ 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.

~~CSD~~ ~~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.

~~CSD~~ ~~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.

~~CSD~~ ~~STD~~ OSS

audience & deliverable

ENGINEERING FUNDAMENTALS

Software Development Approaches

TECHNISCHE UNIVERSITÄT MÜNCHEN

EF 01.2

Development Approaches

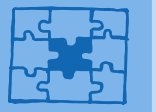
Software Prototyping *mocking* SP

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	X	100%	24-48	5-25
Software Engineering	>150	5-50	X	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	One-Man-Show	One-Man-Show	One-Man-Show	**Team Play**
Responsibility	Single	Single	Single	**Separated**
Model	Mental	Mental	Mental/**Documented**	**Documented**
Decisions	Implicit	Implicit	Implicit/**Explicit**	**Explicit**
Process	**Minimized**	**Partial**	**Partial**	Complete
Optimisation	**Time**	**Efficiency**	**Effectiveness**	**Economics**
Risks	Ignore	Ignore	Ignore	**Mitigate**
Stakeholders	Ignore	Ignore	Ignore	**Manage**
Mastering	**Time-Constraint**	Complexity	**Complication**	Complexity
Solutions	**Use Full**	**Use Full**	Use Partial	Use Partial
Standards	Use	Use	**Potentially Create**	Use
Efforts	**Configuration**	**Integration**	Programming	Programming
Target	**Demo**	Solution	**Product**	**Product**
Sustainability	**No**	**Partial**	Full	Full
Traceability	**No**	**No**	**Partial**	Full

goal & approach

ENGINEERING FUNDAMENTALS

Software Engineering

TECHNISCHE UNIVERSITÄT MÜNCHEN

EF 01.3

Engineering

Machinery

The collection of physical components of a mechanical apparatus using electrical power to perform a certain task.

specialised by

Hardware

The collection of physical components (machinery) of a computer system.

operated by

Software

The collection of instructions and data that tell a computer system how to work.

created by

Engineering

The systematic application of **scientific and technological knowledge, principles, approaches, practices, methods, and experiences**, to **design, build, operate, and maintain** complex production-grade solutions through a **traceable** process. *

transferred to

Software Engineering (transferred)

The systematic application of scientific and technological knowledge, principles, approaches, practices, methods, and experiences, to design, build, operate, and maintain complex production-grade **software** solutions through a traceable process.

simplified to

taking into account

Engineering Code of Conduct

Commit yourself to the **highest ethical and professional conduct**, accept responsibility in making decisions consistent with the **safety, health, and welfare of the public**, apply the **state of the art in science and technology** of your field, perform services only in areas of your **competence**, and comply with the principles of **sustainable development**. *

Wording

Engineering: derived from the *Latin* words **ingeniare** (to create, generate, contrive, devise) and **ingenium** (cleverness).

also taking into account

Software Engineering (simplified)

The systematic application of **engineering** knowledge, principles, approaches, practices, methods, and experiences to **the development of software.**

* derived from definitions and Code of Ethics statements of ASCE, NSPE, ACM, IEEE, etc.

profession & ethics

ENGINEERING FUNDAMENTALS

TRUE Manifesto

true-manifesto.org

TECHNISCHE UNIVERSITÄT MÜNCHEN

targeted

adequate
suitable
focused

Statement: **We focus on adequate and suitable solutions and approaches.**

Rationale: **Both solutions and approaches have to be in a reasonable proportion to the problem.**

Implications:
- We avoid both over-engineered and cobbled-together solutions.
- We avoid "one-size-fits-all" approaches.
- We suitably adapt solutions, tools and methods.

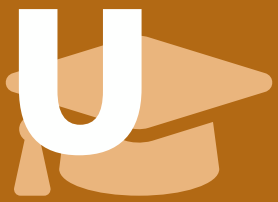
reasoned

considered
assessed
deliberate

Statement: **We think carefully and holistically in advance about our solutions and approaches.**

Rationale: **We always think large, even if we have to act small, because thinking in advance is more efficient and effective than correcting afterwards.**

Implications:
- We always develop the "big picture" first and add ancillary details as late as possible.
- We are opinionated and steadfast regarding our decisions and solutions.
- We know that conceptual modeling is key to understanding both problems and solutions.



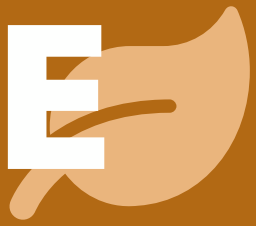
up-to-date

educated
experienced
insistent

Statement: **We develop high-quality solutions on the basis of up-to-date methods and technologies.**

Rationale: **We have to cope with the fact that the IT world is recurrently revolutionizing itself.**

Implications:
- We continuously educate ourselves.
- We continuously and critically challenge and assess emerging approaches and products.
- We are not satisfied with mediocre solutions.



evolutionary

sustainable
harmonic
contextual

Statement: **We develop sustainable solutions that optimally fit into their context.**

Rationale: **Nature teaches us that only evolutionary approaches and solutions have a good chance to survive in the long run.**

Implications:
- We actively learn from experiences of the past in order to improve the future.
- We avoid "quick hacks", as they are not long-term solutions, but just short-term means to get rid of problems.
- We assure that our solutions can be reasonably maintained in the long-term.

EF 01.4

entitlement & values

ENGINEERING FUNDAMENTALS

Software Engineering Metamodel

TECHNISCHE UNIVERSITÄT MÜNCHEN

EF 02.1

Craftsmanship Model 4

WHEREBY & HOW

Know how in a profession to operationally perform certain practices with the help of templates and tools.

Profession

A person's occupation, founded upon specialized educational training and formal qualification.

Know-How

Distinct overall knowledge on how to perform certain tasks.

Practice

Distinct steps to perform a certain task with the help of a certain know-how and template.

Template

The initial version of an artifact in order to speed up and/or support crafting the artifact.

Tool

Software application to create an artifact from scratch or from a template and further edit it.

Discipline Model 1

HOW & WHAT

Based on personal inclinations, understand how to act in area-grouped disciplines and create artifacts and their aspects.

Inclination

A person's natural tendency or urge to act or feel in a certain way.

Area

Logical group of strongly related disciplines.

Discipline

Distinct knowledge, responsibility and objectives within Software Engineering.

Artifact

Single representation of an internally used and/or externally delivered result.

Aspect

Smallest result unit of an artifact which can be created separately.

Workflow Model 2

WHAT & WHO

Understand the workflow of separate iterating cycles and work streams, comprised of one or more steps.

Workflow

The five distinct iteration cycles Product Management, Business, Development, Operations and Project Management.

Cycle

Logical iteration cycle of a sequence of strongly related and dependent workflow steps.

Step

Single step in a workflow cycle with a clear logical previous and next step in the logical work flow.

Flow

Logical sequence of interweaved, inter-cycle, optionally skipped, and dependent steps.

Key Message

Mastering Software Engineering means to understanding (in this order) **Discipline**, **Workflow**, **Process** and **Craftsmanship** models. Especially, *never* start with a particular Process model, as it is secondary!

Process Model 3

WHO & WHEN

Map activities onto a project execution schedule, based on horizontal team focus tracks and vertical temporal project periods and phases.

Role

Distinct responsibility and goal of an individual person.

Track

Work stream consisting of a continuously sequence of steps of a workflow cycle.

Activity

Distinct activity in the process to perform a certain task or complete a particular work-package.

Period

Minor time period in the process, to split the project phases into distinct time-, cost-, or scope-based time units.

Phase

Major time period in the process, to split the project into distinct focus periods, separated by major milestones.

suitable for

correlates to

serialized by 1

operationalised by

corresponds to

operationalises 1

derived from

big picture & coherency

ENGINEERING FUNDAMENTALS

Software Engineering Disciplines

TECHNISCHE UNIVERSITÄT MÜNCHEN

ANALYSIS AN

Software Requirements REQ

Identify Needs: 3
We understand which outcomes of the solution are most valuable to users.

Requirements Engineer / Business Analyst

Domain Modeling DOM

Determine Solution: 2
We model and specify the solution through involved functional and non-functional aspects.

Business Analyst / Business Architect

ARCHITECTURE AR

Software Architecture SWA

Design Software: 1
We design an orthogonal, well-balanced and well-considered solution.

Software Architect

System Architecture SYA

Design Systems: 2
We ensure that the solution fits optimally into its environment.

System Architect / Enterprise Architect

CONFIGURATION CF

Software Versioning VER

Version Artifacts: 1
We place every artifact of the solution under strict version control.

Configuration Manager

Software Assembly ASM

Assemble Artifacts: 1
We build and package the solution through an automated and repeatable mechanism.

Build Manager / Build Engineer

ANALYTICS AC

Software Reviewing REV

Review Code: 4
We regularly and semantically peer-review the source code of the solution.

Software Tester

Software Testing TST

Test Solution: 2
We adequately test the functional and non-functional aspects of the solution.

Software Tester

MANAGEMENT MG

Product Management PRD

Push Product: 3
We continuously push the development and release of the solution to the users.

Product Manager / Product Owner

Project Management PRJ

Steer Process: 3
We rigorously balance time, cost and scope to react on changes and reach the goals.

Project Manager

business-oriented & domain-specific

constructive & technological

infrastructural & technological

analytical & domain-specific

people-oriented & process-oriented

EXPERIENCE EX

User Experience UXP

Optimize Workflows: 3
We align the solution to the perspective of the target audience.

User Experience Expert

User Interface UID

Design User Interfaces: 2
We design a useful, intuitive, and beautiful user interface for the solution.

User Interface Designer / Graphics Designer

DEVELOPMENT DV

Software Development DEV

Implement Code: 1
We develop the solution outside-in, from coarse to fine aspects.

Software Engineer / Software Developer

Software Refactoring REF

Refactor Code: 4
We regularly and holistically refactor the solution to ensure long-term quality.

Software Engineer / Software Developer

DELIVERY DL

Software Deployment DPL

Deploy Artifacts: 1
We ship and deploy the solution through an automated and repeatable mechanism.

System Engineer / System Developer

System Operations OPS

Operate Solution: 4
We ensure that our infrastructures and the solution can be operated in a resilient and secure way.

System Administrator / System Operator

COMPREHENSION CP

Usage Documentation DOC

Document Solution: 2
We adequately document the usage and operation of the solution.

Technical Writer

User Training TRN

Train Users: 4
We adequately train the users and operators of the solution.

Technical Writer / Product Trainer

ADJUSTMENT AD

Project Coaching COA

Support Members: 4
We ensure that project members use state-of-the-art methodology, technology, and tools.

Project Coach / Methodology Master

Change Management CHG

Involve Stakeholders: 3
We ensure that all stakeholders of the solution are suitably involved.

Change Manager

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

EF 02.2

inclination & knowledge