

Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall



Software Classes



Custom Software Developmen Commercial development of non-standardised, fully individualis and non-reusable company-specifi software for a single customer.	nt CSD Standard Software D Sed, ic Commercial development standardised, partially ci and fully reusable doma software for many custor	evelopment STD Open So at of ustomisable, in-specific mers. Open So Non-com standardi and fully for many	ource Software Development OSS imercial development of ised, highly customisable, reusable generic software customers.
Class: Graphics & Media	Class: Business & Data	Class: Machinery & Network	Class: Development & Tools
target audience: consumers & enterprises	target audience: consumers & enterprises	target audience: consumers & enterprises	target audience: vendors & suppliers
Graphics Editing Application GEA	Office Productivity Application OPA	Technical Control System TC	S Software Development Kit SDK
Software for editing and rendering graphics in vector and bitmap format.	Software for productivity in the desktop-based office environment.	Software for controlling a physical machinery or technical system.	Software libraries and frameworks of reusable functionality for developing software.
Examples: Cinema4D, Maya, Blender, After Effects, Illustrator, Inkscape, Scribus, Photoshop, GIMP, etc. OSS	Examples: PowerPoint, Excel, Word, Visio, OmniGraffle, LibreOffice, Outlook, XMind, Firefox, Chrome, etc.	Examples: AquaTherm, AVM! CSU FritzBox Firmware, BirdDog STU Camera Firmware, etc. OSS	Examples: NDI SDK, HAPI, CSD. GraphQL4Q, Sequelize, JDK, STD Spring, Hibernate, etc. OSS
Graphics Animation Engine GAE	Business Information System BIS	Network Communication System NC	CS Software Development Tools SDT
Software for animating the 2D/3D virtual worlds of games and overlays of TV productions.	Software for driving business processes through interactive information management.	Software for protocol-based communication of data over a computer network.	Software tools for editing, linting, compiling, packaging, distributing, and installing software.
Examples: Unity, Unreal Engine, CryENGINE, Godot, HUDS, SPX-GC, Holographics, H2R Graphics, etc. OSS	Examples: Vote, CampS, Mission Control, IPW, KEZ-PSC, TimeSheet, SAP ERP, OpenProject, etc. OSS	Examples: Apache, NGINX, HAProxy, Mosquitto, RabbitMQ, Node-RED, KeyCloak, etc.	Examples: Visual Studio Code, Sublime Text, GCC, GNU Binutils, NPM, JDK, Docker, Helm, etc.
Audio/Video-Processing System AVS	Data Management System DMS	Operating System Kernel OS	Operating System Tools OST
Software for live-processing and post-production of audio/video based multimedia streams.	Software for protocol-based storing and retrieving of persistent data.	Software kernel for low-level operating a physical or virtual device and run programs on it.	Software tools for high-level operating a physical or virtual computing device.
Examples: vMix, OBS Studio, VLC, Lossless Cut, Handbrake, Adobe Premiere, FFmpeg, Nimble, etc. OSS	Examples: NextCloud, PostgreSQL, CockroachDB, Redis, InfluxDB, Neo4J, Tendermind, Gitea, Vault, etc.	Examples: Windows, macOS, iOS, Linux, FreeBSD, QNX, ChibiOS/RT, Kubernetes, Wildfly, etc.	Examples: Coreutils, Bash, CSD, Vim, TMux, FZF, cURL, STD RSYNC, OpenSSH, etc. OSS

There are three traditional approaches to Software Development: **Custom Software Development**, the commercial development of nonstandard, fully individualized, and non-reusable company-specific software for a single customer; **Standard Software Development**, the commercial development of a standardized, partially customizable, but fully reusable domain-specific software for many customers; and **Open-Source Software Development**, the noncommercial development of a standardized, highly customizable, and fully reusable technical software for many customers.

There are four areas and 12 classes of software. In the first area there are three classes of software focusing on Graphics & Media: **Graphics Editing Application**, Software for editing and rendering graphics in vector and bitmap format; **Graphics Animation Engine**, software for animating the 2D/3D virtual worlds of games and overlays of TV productions; and **Audio**/ **Video-Processing Systems**, software for liveprocessing and post-production of audio/video based multimedia streams.

In the second area there are three classes of software focusing on Business & Data: Office Productivity Application, software for productivity in the desktopbased office environment; Business Information System, software for driving business processes through information management; and Data Management System, software for storing and retrieving persistent data. In the third area there are three classes of software focusing on Machinery & Network: **Technical Control System**, software for controlling a physical machinery or technical system; **Network Communication System**, software for communicating data over a computer network; and **Operating System Kernel**, software kernel for operating a physical or virtual computing device.

In the forth area there are three classes of software focusing on Development & Tools: **Software Development Kit**, software libraries and frameworks of reusable functionality for developing software; **Software Development Tools**, software tools for editing, linting, compiling, packaging and installing software; and **Operating System Tools**, software tools for high-level operating a physical or virtual computing device.

Questions

Which two classes of software are primarily developed using the Custom Software Development approach?



So De mo mo to j

So

De sot co sol fea

So

De sof cle me cor

So

De soi risi in co

Software Development Approaches



Development Approaches	Development Approaches: Characteristics Comparison *										
tware Prototyping mocking SP elop an early sample or del of a software solution by king and cheating in order ust once test a concept, idea	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 still (mocking, interarting, crafting, teaming).	thort.Pe	HonDays	ns	9e Traceabilit	Target Te	chrology n. Production Solution	on Grade	ability on Caim	Souther Time	Nott's Cole (N)
mple: Customer Sales Demo	Software Prototyping	1-20	1-2		-	-	-	5%	0-3	0-3	* All figures are just rough orders of magnitude for indication and
tware Bricolage integrating SB	Software Bricolage	5-100	1-2		х	(x)	-	60%	3-24	1-10	illustration purposes.
elop a single instance of a ware solution by tinkering, bling and integrating partial	Software Craftsmanshi	ip 5-100	1-2		х	х	х	100%	24-48	5-25	Key Message All four approaches
itions in order to prove ibility or just provide a service.	Software Engineering	>150	5-50	x x	x	х	х	80%	>48	>25	are equally essential in practice. Which one(s) to choose, entirely depends
tware Craftsmanship crafting SC	So	Deve	lopme	nt Approa	aches:	Succ	ess Pa _{Softw}	attern are	S	So	ftware
elop a production-grade ware solution by professional, n but plain craftsmanship ns in order to solve a usually uplicated problem.	Performance Responsibility Model Ma	rototyping ne-Man-Shov ngle ental	totyping Bricolage 2-Man-Show One-Man-Show gle Single ntal Mental			Craftsmanship One-Man-Show Single Mental/Documented			En Tea Sej Do	gineering am Play parated cumented	
mple: Open Source Framework	Decisions Im Process Mi Optimisation Ti	nplicit inimized me	licit Implicit imized Partial e Efficiency			Implicit/Explicit Partial Effectiveness			Ex Co Eco	plicit mplete onomics	
tware Engineering teaming SE	Risks Igu Stakeholders Igu Mastering Ti	Ignore Ignore Time-Constraint Use Full Use Configuration		Ignore Ignore Complexity Use Partial Use Integration			Ignore Ignore Complication Use Partial Potentially Create Programming			Mi Ma Co	tigate inage mplexity
ware solution by a professional, hedged engineering approach rder to solve a usually place problem	Solutions Us Standards Us Efforts Co									Use Use Pro	Use Partial Use Programming
mple: Business Information System	TargetDeSustainabilityNoTraceabilityNo	emo o o		Solution Partial No			Produ Full Partia	ct I		Pro Ful Ful	oduct I I

One can distinguish four kinds of Software Development approaches.

In **Software Prototyping**, one develops an early sample or model of a software solution by mocking and cheating in order to just once test a concept, idea or process.

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

In Software Craftsmanship, one develops a production-grade software solution by professional, clean but plain craftsmanship means in order to solve a usually complicated problem.

In Software Engineering, one develops a productiongrade software solution by a professional, risk-hedged engineering approach in order to solve a usually complex problem.

The four development approaches can be combined in practice: Prototyping and Bricolage can be earlier stages of Craftsmanship or Engineering. Craftsmanship can be part of Engineering. Each approach requires a special skill. All four approaches are equally essential in practice. Which one(s) to choose entirely depends on the particular requirements.

Questions

- 0 Which Software Development Approach should be choosen to realize a complex Business **Information System?**
- 0 Which Software Development Approach should be choosen to realize a complicated reusable library?

2021-2022 by Dr. Ralf S. Engelschall © 2021 Dr. Ralf S. Engelschall shttps



Software Engineering





Engineering is 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.

Software Engineering is the systematic application of engineering knowledge, principles, approaches, practices, methods, and experiences to the development of software.

For both Engineering and Software Engineering, the following **Code of Conduct** holds: 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

Questions

Is Software Engineering also suitable for the development of a non-complex software in a small team of two people?



Profession Characteristics



Professions usually have two of four characteristics: Being **creative** means involving the use of strong imagination and original ideas. Being **rational** means involving the use of analytical thinking and logical reasoning. Being **practical** means involving the actual doing and real application of something. Being **theoretical** means involving the manifested imagination or analytical study of something. One can distinguish five interesting professions: A **craftsman** acts in a creative and practical way, and solves problems and creates uniques. An **engineer** acts in a rational and practical way, and solves problems and create solutions. An **artist** acts in a creative and theoretical way, and creates beauty and uniques. A **scientist** acts in a rational and theoretical way, and analyses problems and creates knowledge. On the other hand, an **inventor** usually has to combine all characteristics.

TECHNISCHE

UNIVERSITÄT

Questions

When you're dealing with configuration and programming in Software Engineering, instead of an engineer you act more like a...?



Discipline Claim



Intellectual C Graphical II ur



Across the various Software Engineering Disciplines, there are some common properties they all claim from a conceptual point of view: farsighted, tenet-oriented, thoughtful, holistically, adequate, feasible, incremental, valueable, and sustainable.

Questions

What is the most difficult claim in a Discipline in today's practice?

Software Engineering Metamodel



Software Engineering can be understood through a meta-model based on four distinct but interlinked models.

ENGINEERING

FUNDAMENTALS

The **Craftsmanship Model** is the base and targets the WHEREBY & HOW. It spans from the **Professions** of individual persons, their corresponding **Know-Hows** and **Practices** to the underlying **Templates** and **Tools**.

The **Discipline Model** targets the HOW & WHAT. It segregates Software Engineering into **Disciplines**, which are grouped into **Areas** and which are motivated by the usual **Inclinations** of individual persons. Each Discipline is then described through input and output **Artifacts** and their **Aspects**.

The **Workflow Model** targets the WHAT & WHO. It describes a **Workflow** of **Cycles** which contain **Steps**. A **Flow** are the runs through those Steps over time.

TECHNISCHE

UNIVERSITÄT

The **Process Model** finally targets the WHO & WHEN. It maps **Activities** onto a project execution schedule, based on horizontal **Tracks** of **Roles** and vertical **Periods** of **Phases**.

Questions

How many Cycles are known in the Workflow Model of Software Engineering, in which persons with similar Inclinations act?



Software Engineering Disciplines



ANALYSIS AN	ARCHITECTURE 🕁 AR	CONFIGURATION CF	ANALYTICS AC	MANAGEMENT MG
Software Requirements REQ	Software Architecture 🕁 SWA	Software Versioning VER	Software Reviewing REV	Product Management PRD
Identify Needs: 3 We understand which outcomes of the solution are most valuable to users. 0 Requirements Engineer / Business Analyst 0	Design Software: We design an orthogonal, well- balanced and well- considered solution. Software Architect	Version Artifacts: 1 We place every artifact 1 of the solution under 1 strict version control. 1 Configuration Manager 1	Review Code: We regularly and semantically peer-review the source code of the solution. Software Tester	Push Product: We continuously push the development and release of the solution to the users. Product Manager/ Product Owner
Domain Modeling DOM	System Architecture SYA	Software Assembly ASM	Software Testing TST	Project Management PRJ
Determine Solution: We model and specify the solution through involved functional and non-functional aspects. Business Analyst / Business Architect	Design Systems: We ensure that the solution fits optimally into its environment. 2 System Architect / Enterprise Architect 5	Assemble Artifacts: We build and package the solution through an automated and repeatable mechanism. Build Manager/ Build Engineer	Test Solution:2We adequately test the functional and non- functional aspects of the solution.0Software Tester	Steer Process: 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	DEVELOPMENT DV	DELIVERY DL	COMPREHENSION CP	ADJUSTMENT AD
User Experience UXP	Software Development DEV	Software Deployment DPL	Usage Documentation DOC	Project Coaching COA
Optimize Workflows: We align the solution to the perspective of the target audience. User Experience Expert	Implement Code: Implement Code: We develop the solution outside-in, from coarse to fine aspects. Implement Code: Software Engineer / Software Developer	Deploy Artifacts: We ship and deploy the solution through an automated and repeatable mechanism. System Engineer	Document Solution: We adequately document the usage and operation of the solution. 2 Technical Writer	Support Members: We ensure that project members use state-of- the-art methodology, technology, and tools. Project Coach / Methodology Master
User Interface UID	Software Refactoring REF	System Operations OPS	User Training TRN	Change Management CHG
Design User Interfaces: We design a useful, intuitive, and beautiful user interface for the solution. User Interface Designer / Graphics Designer	Refactor Code: We regularly and holistically refactor the solution to ensure long- term quality. Software Developer	Operate Solution: We ensure that our infra- structures and the solution can be operated in a resilient and secure way. System Administrator / System Operator	Train Users: 4 We adequately train the users and operators of the solution. B Product Expert C	Involve Stakeholders: We ensure that all stakeholders of the solution are suitably involved. Change Manager

WB white-box view (details before whole) BB black-box view (whole before details)

individual persons.

Software Engineering can be understood through 20 distinct **Disciplines** (operationalized through input and output artifacts and their aspects), which are logically grouped into 10 distinct **Areas**, and which in turn are logically grouped into 5 distinct **Inclinations** of

Persons with a strong **domain-specific** and **businessoriented** Inclination act in the Areas **Analysis** and **Experience** and the corresponding Disciplines **Software Requirements**, **Domain Modeling**, **User Experience** and **User Interface**.

Persons with a strong **constructive** and **technological** Inclination act in the Areas **Architecture** and **Development** and the corresponding Disciplines **Software Architecture**, **System Architecture**, **Software Development** and **Software Refactoring**. Persons with a strong **infrastructural** and **technological** Inclination act in the Areas **Configuration** and **Delivery** and the corresponding Disciplines **Software Versioning**, **Software Assembly**,

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

Persons with a strong **analytical** and **domain-specific** Inclination act in the Areas **Analytics** and **Comprehension** and the corresponding Disciplines **Software Reviewing**, **Software Testing**, **Usage Documentation** and **User Training**.

Software Deployment and Software Operations.

Persons with a strong **people-oriented** and **process-oriented** Inclination act in the Areas **Management** and **Adjustment** and the corresponding Disciplines **Project Management**, **Project Auditing**, **Project Coaching** and **Change Management**.

Questions

Which Disciplines of Software Engineering are considered the **King Disciplines**?