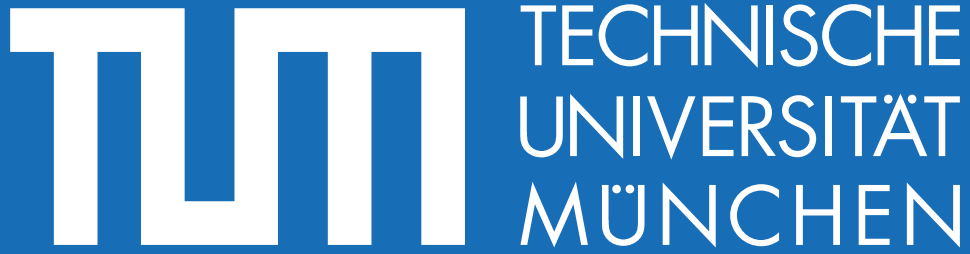


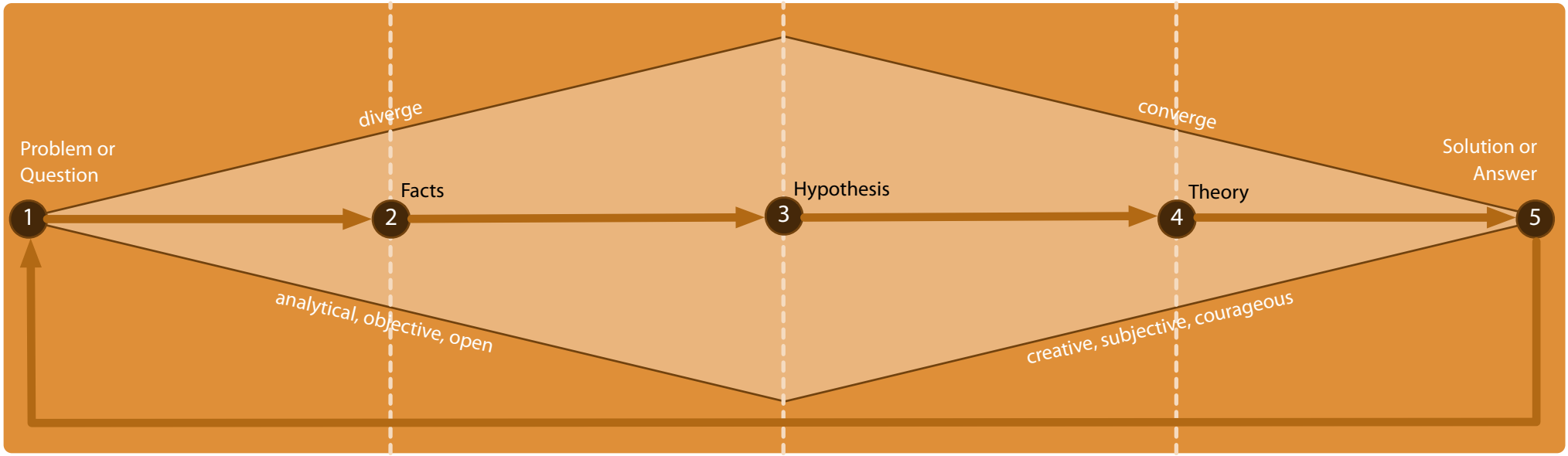
Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

ARCHITECTURE FUNDAMENTALS

Think Clearly

TECHNISCHE UNIVERSITÄT MÜNCHEN



INVESTIGATE & RESEARCH

1. Reflecting
(find facts via own knowledge/experience)

2. Searching
(find facts via body of knowledge)

3. Verification
(cross-check facts according to sources)

4. Tagging
(classify facts with tags)

STRUCTURE & SORT

1. Typing
(split/aggregate facts according to type)

2. Clustering
(hierarchically group facts by tags)

3. Relating
(link source to target facts)

4. Ordering
(order facts in each cluster)

REDUCE & COMPLEMENT

1. Substituting
(substitute/rename facts)

2. Extending
(add still missing facts)

3. Priorization
(priorize facts according to criterias)

4. Rejecting
(reject non-relevant/redundant facts)

INTEGRATE & PRESENT

1. Specialization
(specialize too general facts)

2. Generalization
(generalise too specific facts)

3. Integration
(aggregate/link facts)

4. Presentation
(convert facts into target form)

AF 16.1

Intellectual Content: Version 1.0.1 (2019-06-09), Authored 2010 by Dr. Ralf S. Engelschall based on work of Mark Lubkowitz.
Graphical Illustration: Version 1.0.1 (2019-06-09), Copyright © 2019 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

ARCHITECTURE FUNDAMENTALS

Problem Solving Heuristics

TECHNISCHE UNIVERSITÄT MÜNCHEN

Research (RE)

Crawling the problem domain's body of knowledge to find starting points.

Abstraction (AB)

Solving the problem in a model of the problem before applying it to the real problem to get a better understanding.

Lateral Thinking (LT)

Approaching the problem indirectly and creatively to find a not obvious solving lever.

Backward Search (BS)

Looking at the expected results and determine which operations could bring you to them.

Brainstorming (BR)

Suggesting larger number of solution ideas for further combination and development.

Generalization (GE)

Thinking about the problem more abstract to get rid of special cases.

Hypothesis Proof (HP)

Assuming a possible solution and trying to prove (or disprove) the assumption to find starting points.

Q.E.D.

Backtracking (BT)

Remembering path towards the solution and on failure tracking back and choosing a new path.

Analogy (AN)

Thinking in terms of similar problems for which solutions are known to get inspired.

Specialization (SP)

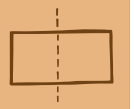
Solving a special case first to get an impression towards the full solution.

Root Cause (RC)

Asking "Why?" five times in sequence to explore the cause-and-effect relationships underlying the problem.

Divide & Conquer (DC)

Breaking down the large complex problem into smaller, easier solvable partial problems.



Reduction (RD)

Transform the problem into another one for which a solutions already exists to reduce solving efforts.

Variation (VA)

Changing the problem context or expressing the problem differently to find a not obvious solving lever.

Means End (ME)

Choosing an action from scratch just at each step to move closer and closer to the solution.

Trial & Error (TE)

As a last resort, brute-force testing all potential solutions in case of a small enough total solution space.

Definition: ***Heuristic*** — fallible experience-based technique or strategy for problem solving in case *Rule of Thumb Guessing*, *Intuitive Judgement*, *Common Sense* and *Stereotyping* are either not sufficient or not appropriate.

Intellectual Content: Version 1.0.3 (2019-10-31), Authored 2013-2019 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.0.4 (2021-02-21), Copyright © 2013-2021 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

ARCHITECTURE FUNDAMENTALS

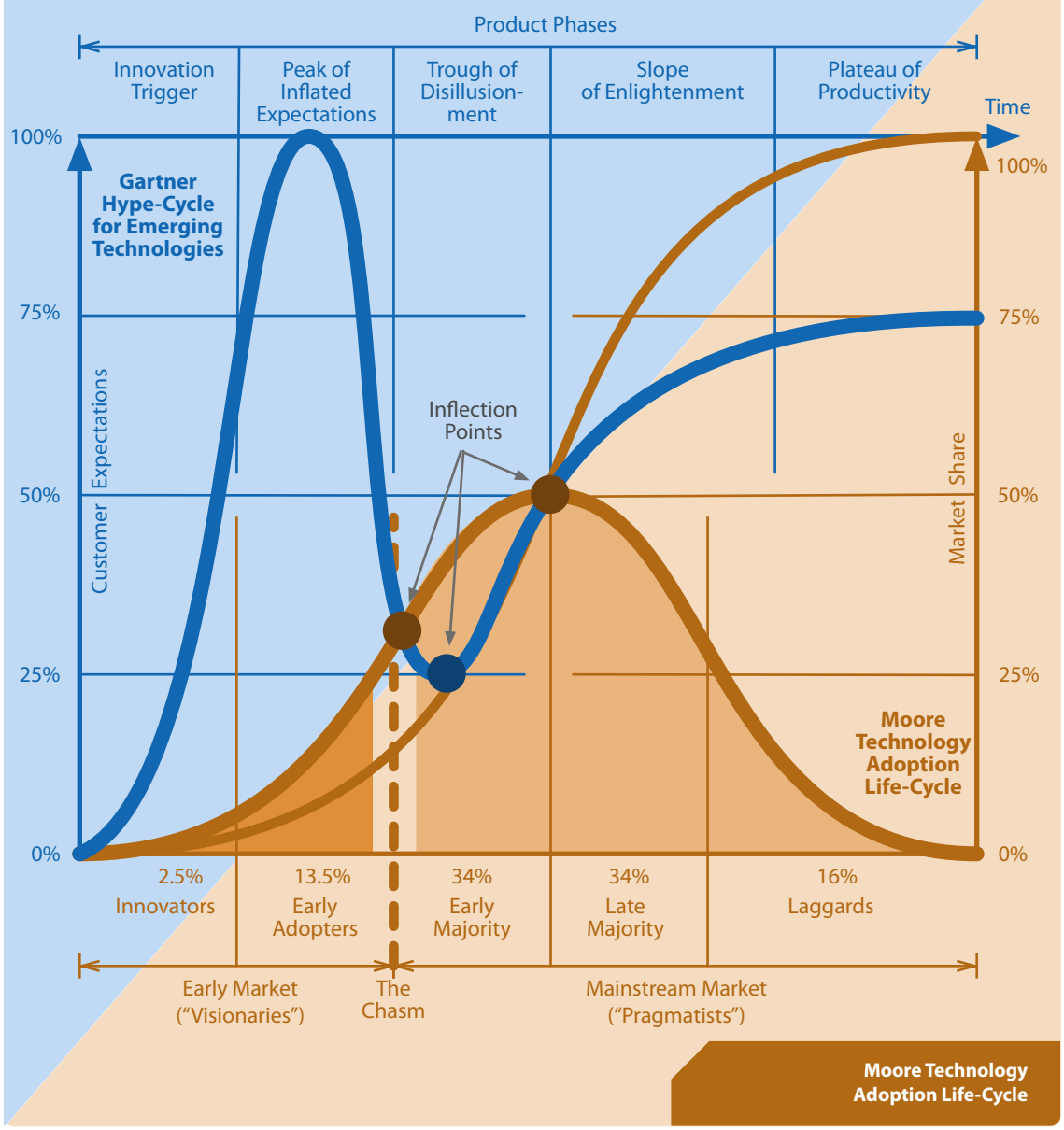
Technology Life-Cycles

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 16.3

Intellectual Content: Version 1.03 (2020-05-16) Authored 2018-2020 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.05 (2019-10-31) Copyright © 2018-2019 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

Gartner Hype-Cycle for Emerging Technologies



Gartner Hype-Cycle for Emerging Technologies

According to [1], provides "a graphic representation of the maturity and adoption of technologies and applications, and how they are potentially relevant to solving real business problems and exploiting new opportunities." It gives "a view of how a technology or application will evolve over time." The five product phases are:

"**Innovation Trigger:** A potential technology breakthrough kicks things off. Early proof-of-concept stories and media interest trigger significant publicity. Often no usable products exist and commercial viability is unproven.

Peak of Inflated Expectations: Early publicity produces a number of success stories — often accompanied by scores of failures. Some companies take action; many do not. The peek can be also considered a direct result of the *Dunning-Kruger Effect*, a "cognitive bias in which people mistakenly assess their cognitive ability as greater than it is" [2] and hence exaggerate in their expectations.

Trough of Disillusionment: Interest wanes as experiments and implementations fail to deliver. Producers of the technology shake out or fail. Investments continue only if the surviving providers improve their products to the satisfaction of early adopters.

Slope of Enlightenment: More instances of how the technology can benefit the enterprise start to crystallize and become more widely understood. Second- and third-generation products appear from technology providers. More enterprises fund pilots; conservative companies remain cautious.

Plateau of Productivity: Mainstream adoption starts to take off. Criteria for assessing provider viability are more clearly defined. The technology's broad market applicability and relevance are clearly paying off."

Moore Technology Adoption Life-Cycle

According to [3], describes "the adoption or acceptance of a new product or innovation, according to the demographic and psychological characteristics of defined adopter groups." The five distinct adopter groups are:

"**Innovators:** had larger" business, "were more educated, more prosperous and more risk-oriented.

Early Adopters: younger, more educated, tended to be community leaders, less prosperous.

Early Majority: more conservative but open to new ideas, active in community and influence to neighbours.

Late Majority: older, less educated, fairly conservative and less socially active.

Laggards: very conservative, had small" business "and capital, oldest and least educated."

According to [4], there is also a "chasm between the early adopters of the product (the technology enthusiasts and visionaries) and the early majority (the pragmatists)," because "visionaries and pragmatists have very different expectations." and technology is usually switched, at last at the Inflection Points.

Crossing The Chasm [4] is related to the *Innovator's Dilemma* [5], where "new entry next generation products" usually "find niches away from the incumbent customer set to build the new product."

[1] https://gtnr.it/36rBT4X
[2] https://bit.ly/2qZ4Lkx
[3] https://bit.ly/2N3fB1t
[4] https://bit.ly/2NuRNT7
[5] https://bit.ly/34lMkEW

ARCHITECTURE FUNDAMENTALS

Open Source Software

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 16.4

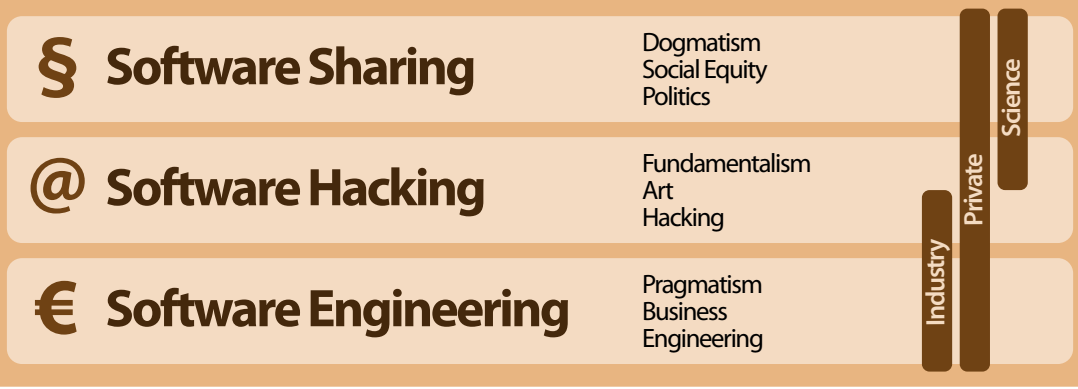
Intellectual Content: Version 1.0.2 (2023-04-22), Authored 2007-2023 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.0.2 (2022-10-09), Copyright © 2014-2022 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

Open Source Definition

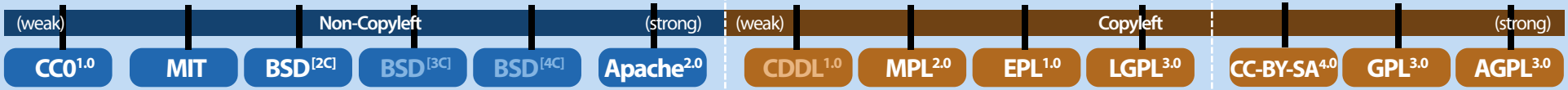
Distribution terms (license) of Open Source Software must be compliant with the following criterias:

- Free **Redistribution**
- (Original) **Source Code** (Availability)
- **Derived Works** (Allowance)
- **Integrity** of the Author's **Source Code**
- **No Discrimination** Against **Persons** or Groups
- **No Discrimination** Against **Fields** of Endeavor
- **Distribution** of (Non-Exclusive) **License**
- License Must **Not Be Specific to a Product**
- License Must **Not Restrict Other Software**
- License Must Be **Technology-Neutral**

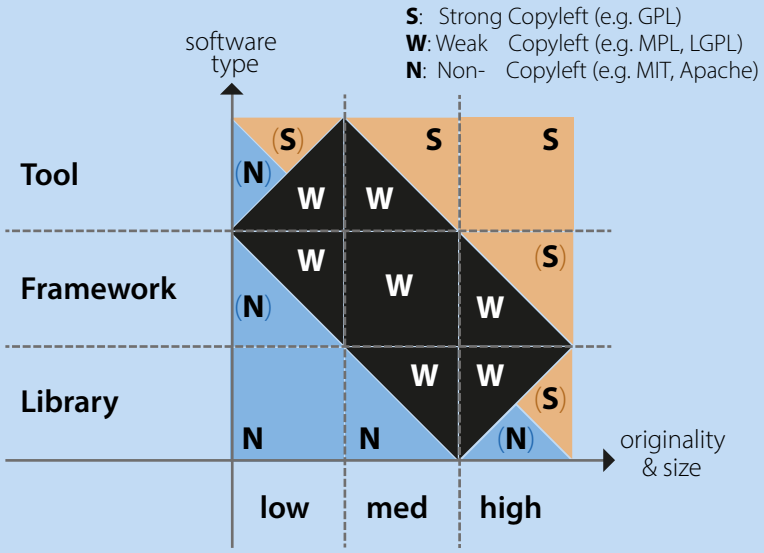
Open Source Personality Streams



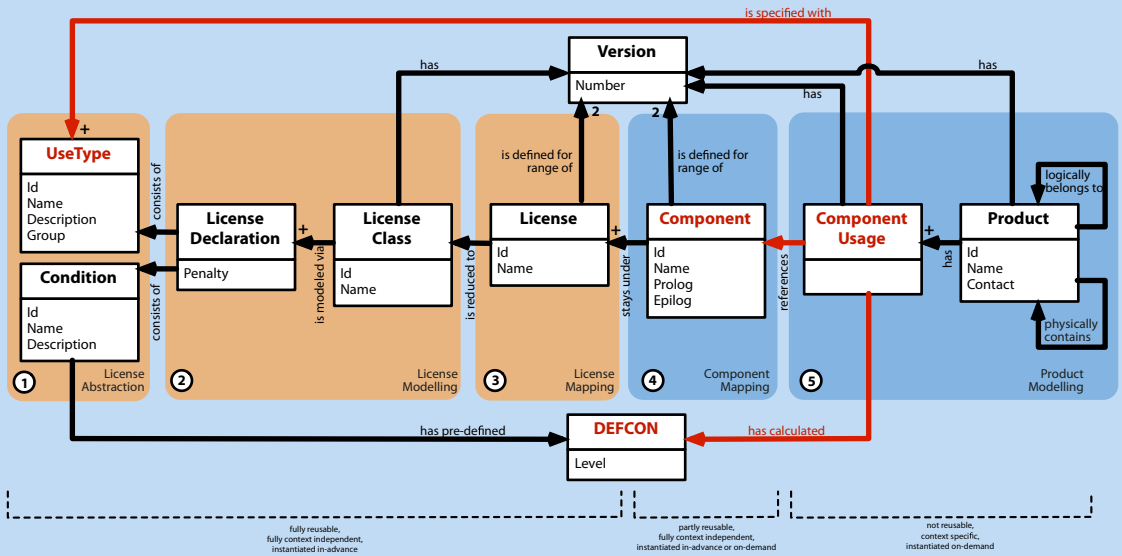
Most Popular Open Source Licenses



Choosing an Open Source License



License Compliance Checking Meta-Model



ARCHITECTURE FUNDAMENTALS

Back of the Envelope Calculation

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 16.5

Specification (Example)

Customer: Twitter Inc.
Business: MicroBlogging

Use-Cases 1/3 (profile):
- user can register an account
- user can "follow" other users
- user can create lists of users he follows

Use Cases 2/3 (send):
- user can send tweets
- tweets are based on words, each either a text "example", tag "#example", user reference "@example" or URL http://example.com
- tweets are either public broadcast or personal/direct messages
- user can re-tweet a message of others

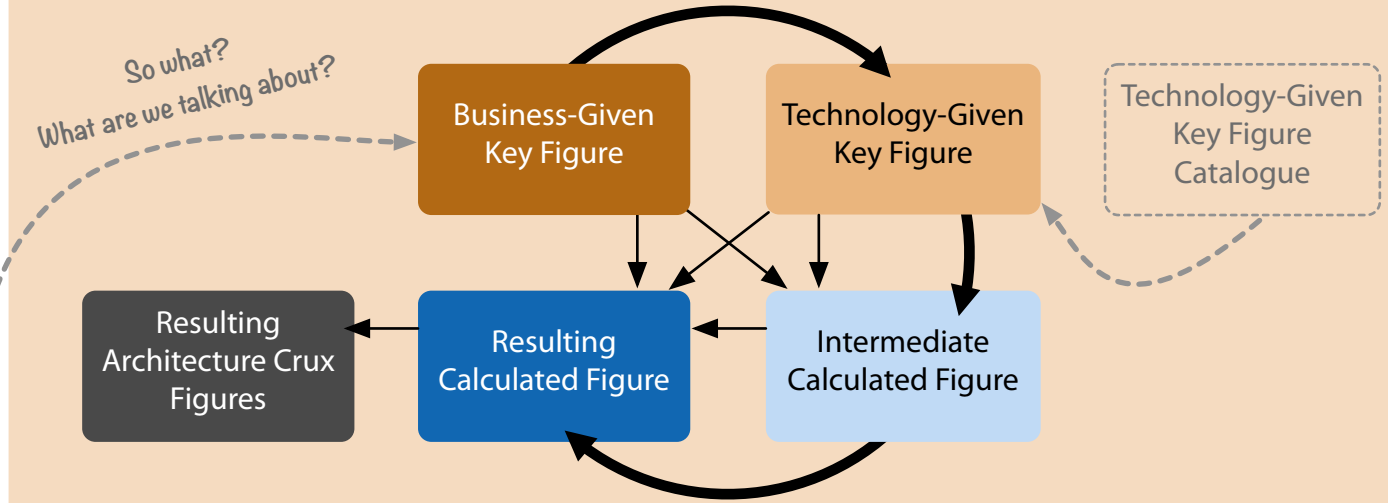
Use Cases 3/3 (query):
- user can view timeline (chronological tweets of others he follows)
- user can search for tweets (by keyword "foo", tag "#foo", or user "@foo")
- user can view tag cloud

Frontends/Clients:
- mobile app (iOS, Android)
- desktop app (Windows, Mac OS X)
- web app
- embedded web widget (query use cases only)

Current Demand (as of 2012):
- 140M user profiles
- 400M tweets/day
- 6393 tweets/second peak
- 140 characters/tweet
- 30K queries/second
- 300 GB/hour data in total
- 4,4 tweets/day/user on average
- 103,4 follower/user
- < 5s tweet-write-to-read-delay

Future Demand:
- quadratic user and traffic growth

Conversion & Normalization



Calculation (Example)

Twitter Information

6.393	tweets/second peak
400.000.000	tweets/day (write)
2.592.000.000	queries/day (read)
6,5	factor read/writes

140	chars/tweet
4.630	tweets/second (write)
30.000	queries/second (read)
10	tweets/query
4,4	tweets/day/user
2,4	tweets/day (M. Fowler)
0,8	tweets/day (R. Engelschall)
621.880.000	tweets/day (average) total
64,3%	users are active at all
277.778	users/minute active

Traffic Bandwidth

350%	overhead HTTP+TCP+IP+Ethernet
2,2	MB/s (write)
140,2	MB/s (read)
142,4	MB/s

10000	Mbps
1000	Mbps
100	Mbps
10	Mbps

1250	MB/s
125	MB/s
12,5	MB/s
1,25	MB/s

Storage Requirements (static)

140.000.000	user profiles
52.000	chars/users for profile
6,62	TB profile (total)
103,4	follower/user
14.474.600.000	user follow links
32	bytes/link
0,42	TB links (total)

Storage Hardware Requirements

0,3	TB/disk (15K rpm)
8,0	disks/server
2,4	TB/server
8,2	server/month (new)

Storage Requirements (dynamic)

300	GB/hour data in total
214	TB/month data in total
200%	overhead storage
1265,9	KB/s tweets
104,3	GB/day tweets
3,1	TB/month tweets
200	chars/log entry
6763,6	KB/s log
557,3	GB/day log
16,6	TB/month log
9,2%	ratio business data
90,8%	ratio infrastructure data

Computing Hardware Requirements

2000	requests/sec (read) AS performance
100	requests/sec (write) As performance
15,0	servers for writes
46,3	servers for reads

Intellectual Content: Version 1.1.0 (2021-11-23) Authored 2011-2021 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.3.0 (2023-10-10) Copyright © 2012-2023 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

ARCHITECTURE FUNDAMENTALS

Weighted Decision Matrix

TECHNISCHE UNIVERSITÄT MÜNCHEN

Weighted Decision Matrix

		A_1	A_2	…	A_n
C_1	w_1	$E_{1,1}$	$E_{1,2}$	…	$E_{1,n}$
C_2	w_2	$E_{2,1}$	$E_{2,2}$	…	$E_{2,n}$
…	…	…	…	…	…
C_m	w_m	$E_{m,1}$	$E_{m,2}$	…	$E_{m,n}$
		R_1	R_2	…	R_n

$C_{i=1..m}$: Criteria i

$A_{j=1..n}$: Alternative j

$w_{i=1..m}$:in [1/4, 1/2, 1, 2, 4]: Weighting i

$E_{i,j}$ in { -2, -1, 0, +1, +2 }: Evaluation i,j

$R_{j=1..n} = SUM_{i=1..m}(w_i * E_{i,j})$: Rating j

$R_{best} = MAX_{j=1..n}(R_j)$: Best Rating

Light-Weight Alternative:
qualitatively cherry-picking major positive/negative backing criterias

	A_1			A_2			…	A_n		
+	C_1	C_4	C_8	C_1	C_2	C_7	…	C_2	C_9	C_8
-	C_2	C_3		C_4			…	C_7	C_5	
	Decision for A_{best}									

Best Practice Rules

Rule 1: the alternatives have to be really **reasonably comparable**.

Rule 2: the **best** rating should be a least **10% above** the **second** best rating.

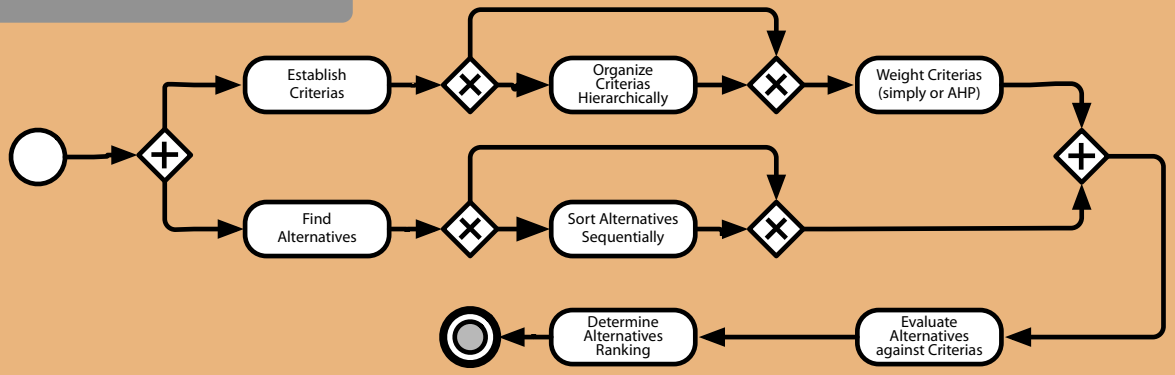
Rule 3: the **best** rating should cover at least **80%** of the **requirements**.

Rule 4: the Weighted Decision Matrices should cover at least all **grand decisions**.

Notice

It's about **subjective** decision **transparency**, not about **objective** decision **making**!

Decision Making Process



Standard Criteria Catalogs

Software Selection:
Suitable Functionality
Available Usage Examples
Reasonable Documentation
Reasonable Support
Permissive License
Long-Term Release Track Record
Current Market Momentum

Software Selection (Open Source):
+ Clean Source Code
+ Clean Build Process
+ Open Source License

Software Selection (Library):
+ Non-Invasive Programming Model
+ Orthogonal Application Programming Interface
+ Minimum/No Dependencies
+ Non-Copyleft Open Source License

Software Selection (Framework):
+ Orthogonal Application Programming Interface
+ Adequate Dependencies
+ Non-Overlapping Scope
+ Non-Copyleft Open Source License

Software Selection (Tool):
+ Clean Deployment Procedure
+ Pleasant Command-Line Interface

Software Selection (Application):
+ Clean Deployment Procedure
+ Pleasant Graphical User Interface

Software Architecture Evaluation:
Meets Functional Requirements
Meets Non-Functional Requirements
Adequate Technology Overhead
Single Dependency Direction
Distance to State of the Art ("modern")
Distance to Most Simple Approach ("adequate")
Distance to Mainstream Approach ("mainstream")
Documented Architecture Decisions ("rationales")
Documented Architecture Views
Documented Architecture Perspectives (NFR)

AF 16.6

Intellectual Content: Version 1.11 (2022-05-20), Authored 2018-2022 by Dr. Ralf S. Engelschall <http://engelschall.com>. All Rights Reserved.
Graphical Illustration: Version 1.10 (2021-09-26), Copyright © 2018-2021 Dr. Ralf S. Engelschall <http://engelschall.com>. All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

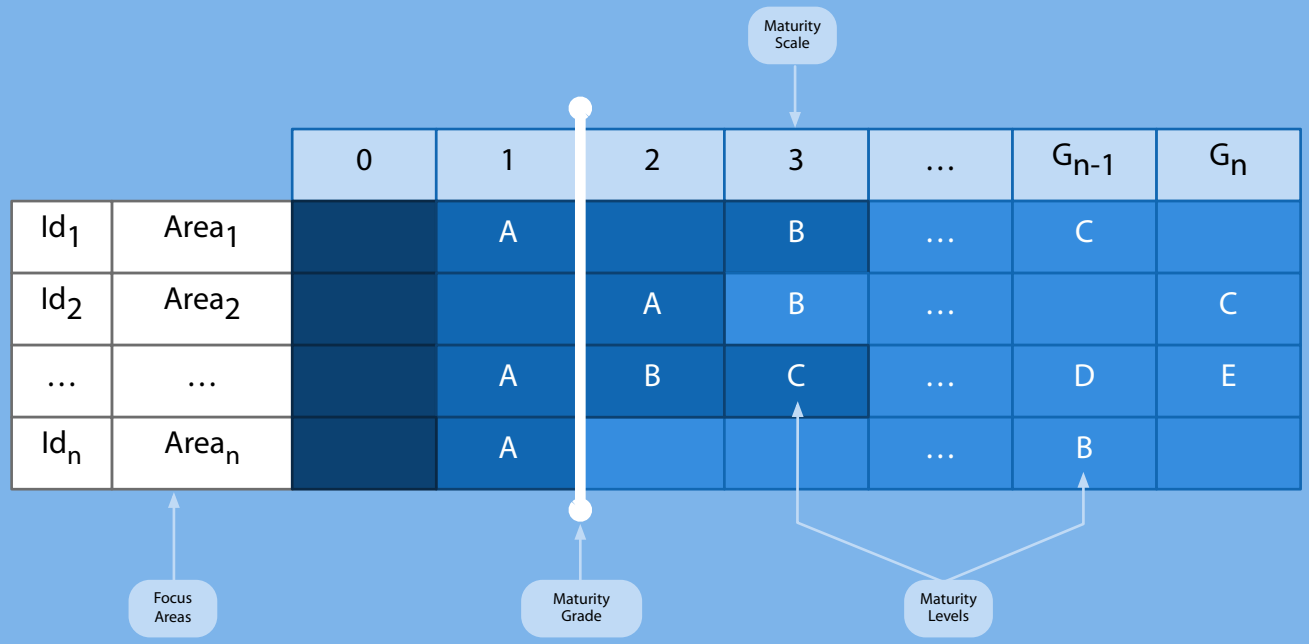
ARCHITECTURE FUNDAMENTALS

Focus Area Maturity Model

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 16.7

Matrix **Structure**



Focus Area Definition

Id: <unique id of focus area>
Name: <unique name of focus area>

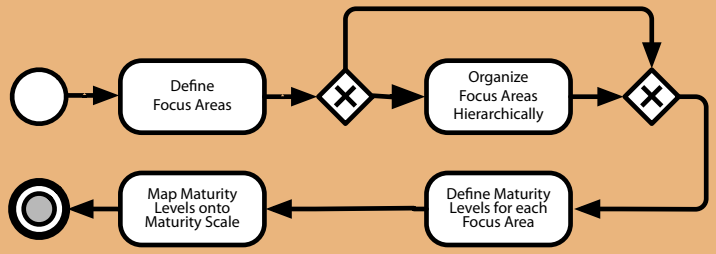
Maturity Level Definition

Id: <unique id of focus area>
Level: <unique letter of maturity level>
Name: <unique name of capability>
Goal: <purpose the capability serves>
Action: <steps how to meet the capability>
Prerequisites: <optional references to Id/Level>
References: <optional external references>

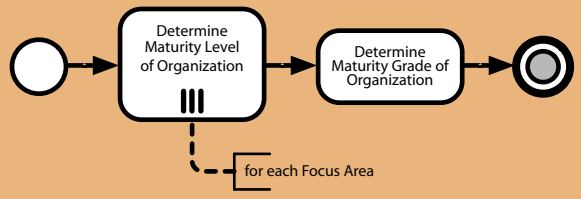
Maturity Level **Prerequisites**

Notice: Maturity Levels are inherently ordered within their Focus Area, but optionally also form a dependency graph by cross-referencing Maturity Levels of other Focus Areas.

Matrix **Design** Process



Maturity **Decision** Process



Maturity Grade **Zero**

Notice: the Maturity Scale always starts with 0, because an organization might not be able to fulfil a Focus Area at all, i.e., it might to not even be on Maturity Level A.

Maturity Grade **Determination**

Determine minimum Maturity Level fulfilled by an organization and project from Maturity Level onto Maturity Scale.

Intellectual Content: Version 1.0.0 (2022-05-21), Authored 2022 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.0.0 (2022-05-21), Copyright © 2022 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

ARCHITECTURE FUNDAMENTALS

Big Picture (8-D)

TECHNISCHE UNIVERSITÄT MÜNCHEN

AF 17.1

Vote is a portable mobile-first designed application for easily performing anonymous online votings within a small group of people to figure out their opinions or moods.

Votings are created in advance, executed at a certain time, conducted by the users, and then finally reported.

1 Elevator Pitch (RE D)

Name, Purpose, Motivation, Actors, Devices.

Rationale: Roughly describe the purpose and primary motivation.

Format: Prose Abstract

2 Crux Flash (RE D)

Functionality, Cruxes.

Rationale: Roughly describe the functionality and the cruxes.

Format: Prose Abstract

Votings can be quickly accessed by QR-code or URL and are based on one or more questions and corresponding multiple-choice-based answers.

Votings are interactively conducted, and answers are received and reported either asynchronously in batches (offline voting) or even synchronously in real-time (online voting).

Actor Role	Use-Case
User	Register Account
	Recover Account
	Configure Account
	Login Account
	Logout Account
Author	Create Voting
	Grant Voting Access
Supervisor	Execute Voting
	Enable Question
	Display Result
Voter	Vote Question
	Display Result

3 Customer Journey (UX D)

Actor Roles, Use-Cases.

Rationale: Sketch the customer journey through major use-cases.

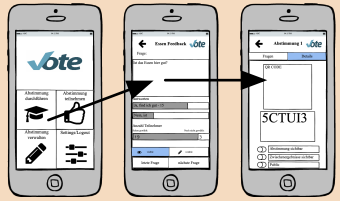
Format: 2xN Table or UML UC Diag.

4 Dialog Storyboard (UX D)

Dialogs, Interaction, Control Flow.

Rationale: Illustrate the major user interface dialogs (or dialog types).

Format: Wireframe Graph Diagram



Quality	Expectation
Cross-Platform Client	yes
Non-Cleartext Password Storage	yes
Minimum Concurrent Voters (people)	50
Maximum Display Result Latency (sec)	1

5 Quality Requirements (RE T)

Qualities, Expectations.

Rationale: List requirements on the major non-functional qualities.

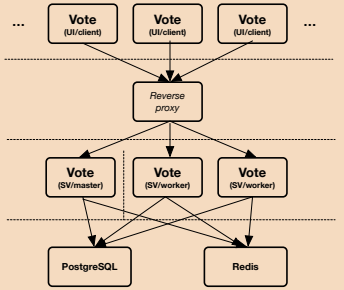
Format: 2xN Table

6 System Architecture (SY T)

Actors, Systems, Zones, Programs.

Rationale: Illustrate the major system architecture components.

Format: Boxes'n'Lines Diagram



Aspect	Amount	Size	Total Size	Unit
Account Data	10.000	256	2.560.000	B
Voting Data	10.000	1.024	10.240.000	B
Server RAM Usage	200	20	4.000	MB

8 Sizing Sketch (SW T)

Aspects, Amounts, Sizes, Total Sizes, Units.

Rationale: Sketch the sizing of major entities and system parts.

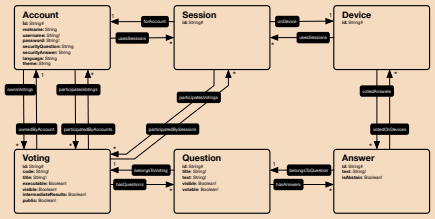
Format: 5xN Table

7 Data Model (SW T)

Entities, Relationships.

Rationale: Model major data entities and their relationships.

Format: UML Class Diagram



(Example) (Method) (Example)

1 Creation Step · RE Requirements Engineering · UX User Experience · SY Systems Architecture · SW Software Architecture · D Domain Scope · T Technology Scope

Intellectual Content: Version 1.03 (2020-07-20), Authored 2019-2020 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.03 (2020-07-20), Copyright © 2019-2020 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

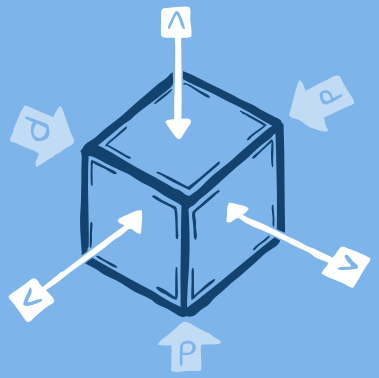
ARCHITECTURE FUNDAMENTALS

Viewpoints & Perspectives

TECHNISCHE UNIVERSITÄT MÜNCHEN

Development
5

Aspects of the software development process for versioning, building, testing, maintaining, and enhancing the system.



Development Process Viewpoints

Functionality
2

System's functional elements, their responsibilities, interfaces, and primary interactions (control flow).

Information
3

Static data structures and information flows to store, manipulate, manage, and distribute information.

Concurrency
4

Concurrency structure of the system and mapping of functional elements to concurrency units (processes, threads, transaction scopes).

Application Architecture Viewpoints

Context
1

Relationships, dependencies, and interactions between the system and its run-time environment (people, systems, external entities).

Application Overview Viewpoints

Deployment
6

Required technical environment and mapping of software elements to runtime environment that will execute them.

Operation
7

Aspects to operate, administer, update, upgrade and support the system when running in its production environment.

Operation Process Viewpoints

Evolution & Change

Ability of the system to be flexible in the face of the inevitable change that all systems experience over time.

Regulation & Compliance

Ability of the system to conform to local and international laws, quasi-legal regulations, company policies, and other rules and standards.

Internationalization & Localization

Ability of the system to be independent from and adaptable to any particular language, country, or cultural group.

Usability & Accessibility

Ability of the system to allow people to effectively interact with the system and also to be even used by people with disabilities.

Performance & Scalability

Ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes.

Availability & Resilience

Ability of the system to be fully or partly operational when required and to effectively handle failures.

Constraints & Resources

Ability of the system to be designed, built, deployed, and operated within known constraints around people, budget, time, and materials.

Security & Recoverability

Ability of the system to reliably control and audit who can perform what actions on what resources and to detect and recover from failures.

Cross-Cutting Perspectives

AF 17.2

Intellectual Content: Version 1.04 (2019-09-15) Authored 2011-2019 by Dr. Ralf S. Engelschall based on Nick Rozanski, Eoin Woods: "Software Systems Architecture", 2nd ed. 2011, ISBN 978-0-321-71833-4
Graphical Illustration: Version 1.08 (2018-09-15) Copyright © 2011-2018 Dr. Ralf S. Engelschall <http://engelschall.com> All Rights Reserved
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.