TECHNISCHE
UNIVERSITÄT
MÜNCHEN

Software Engineering in Industrial Practice (SEIP)

Dr. Ralf S. Engelschall

ENGINEERING FUNDAMENTALS

Sustainability

TECHNISCHE UNIVERSITÄT MÜNCHEN

HI — Minimize HARDWARE Idleness

Minimize the idleness and maximize the utilization of existing hardware resources.

Rationale: Unused or under-utilized hardware are an unnecessary waste of already available resources.

Keywords: Virtualization, Utilization.

DE — Minimize DESIGN Excessiveness

Minimize the excessiveness and maximize the adequacy of solution designs.

Rationale: Non-adequate designs cause unnecessary complexity and waste resources.

Keywords: Reduced Libraries, Immutability.

HE — Minimize HUMAN Effort

Minimize the efforts of humans and maximize the efforts of machines in all production and operation processes.

Rationale: Delegating tasks to machines gives humans the possibility to concentrate on more important tasks.

Keywords: Computer, Robot, Automation.

SI — Minimize SOFTWARE Inefficiency

Minimize the inefficiency and maximize the efficiency of software applications and their development processes.

Rationale: Efficient software and development processes consume less resources.

Keywords: Caching, Monolith.

SE — Minimize SOLUTION Ephemerality

Minimize the ephemerality and maximize the life-span of any type of solutions.

Rationale: Short life-spans of solutions cause unnecessary short renewals and this way wastes resources.

Keywords: High Quality, Best Practice.

EC — Minimize ENERGY Consumption

Minimize the consumption and maximize the saving of energy in all production and operation processes.

Rationale: Electric energy still has to be partially generated from non-renewable resources.

Keywords: Eco Mode, Reduced CI/CD.

IA — Minimize INFORMATION Amount

Minimize the total amount of gathered, transmitted, stored and spreaded information.

Rationale: Reduced amount of information means less data transmission, less data storage, less GDPR issues, etc.

Keywords: Compression, No Big Data.

EE — Minimize ECOSYSTEM Exploitation

Minimize the exploitation and maximize the back-contribution in any type of ecosystems.

Rationale: The consumer and provider behaviour have to be in balance for every long-lasting ecosystem.

Keywords: Open Source Software.

CE — Minimize CARBON Emission

Minimize the carbon emission and hence the footprint during any type of production and operation processes.

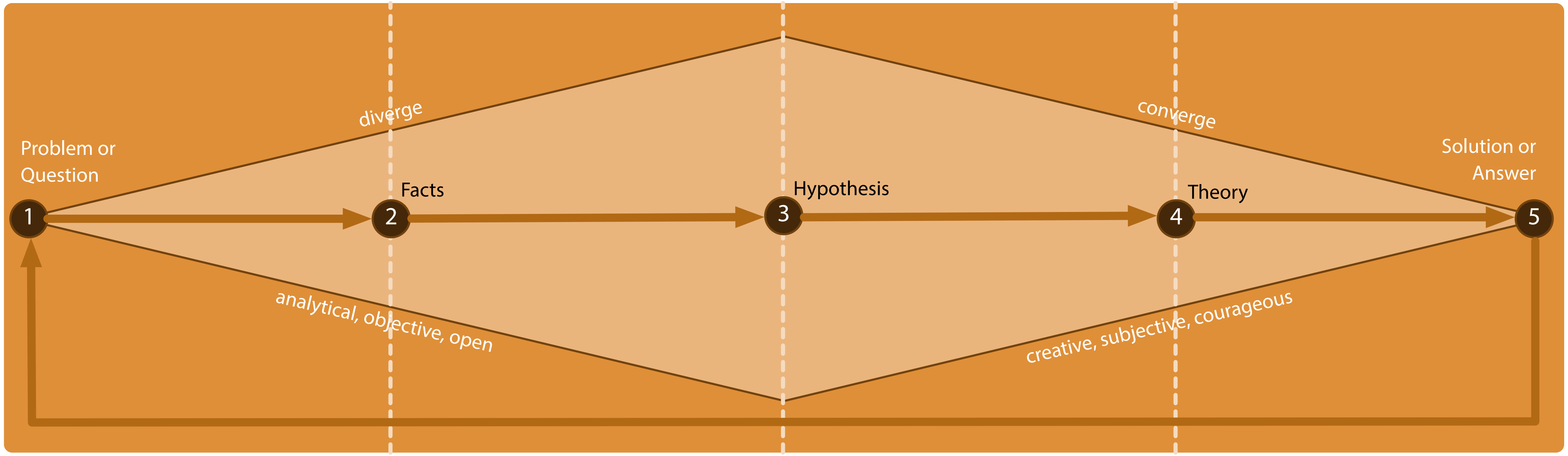
Rationale: Climate change and global warming is partially caused or at least accelerated by carbon emissions.

Keywords: Reduced CO2 Footprint.

ARCHITECTURE FUNDAMENTALS

Think Clearly

TECHNISCHE UNIVERSITÄT MÜNCHEN



INVESTIGATE & RESEARCH	STRUCTURE & SORT	REDUCE & COMPLEMENT	INTEGRATE & PRESENT
1. Reflecting (find facts via own knowledge/experience)	1. Typing (split/aggregate facts according to type)	1. Substituting (substitute/rename facts)	1. Specialization (specialize too general facts)
2. Searching (find facts via body of knowledge)	2. Clustering (hierarchically group facts by tags)	2. Extending (add still missing facts)	2. Generalization (generalise too specific facts)
3. Verification (cross-check facts according to sources)	3. Relating (link source to target facts)	3. Priorization (priorize facts according to criterias)	3. Integration (aggregate/link facts)
4. Tagging (classify facts with tags)	4. Ordering (order facts in each cluster)	4. Rejecting (reject non-relevant/redundant facts)	4. Presentation (convert facts into target form)

(primary step) (secondary step)

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.

5x WHY?

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

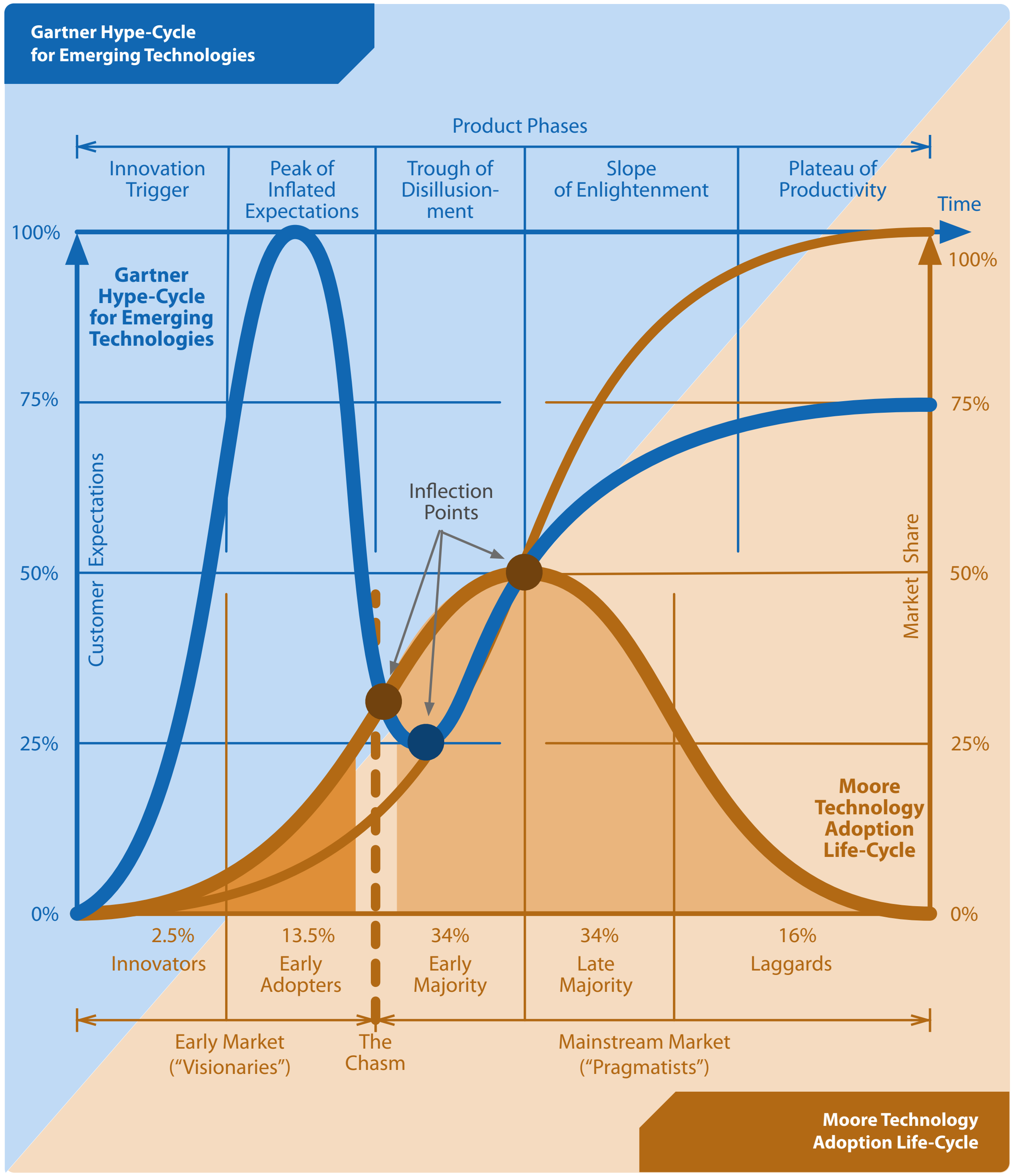
As a last resort, brute-force testing all potential solutions in case of a small enough total solution space.

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.

ARCHITECTURE FUNDAMENTALS

Technology Life-Cycles

TECHNISCHE UNIVERSITÄT MÜNCHEN



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/34IMkEW

ARCHITECTURE FUNDAMENTALS

Open Source Software

TECHNISCHE UNIVERSITÄT MÜNCHEN

Open Source Definition

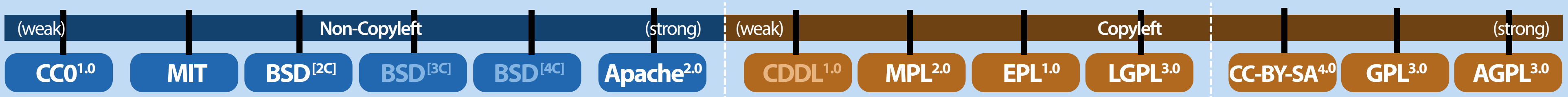
Distribution terms (license) of Open Source Software must be compliant with the following criterias:

- 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

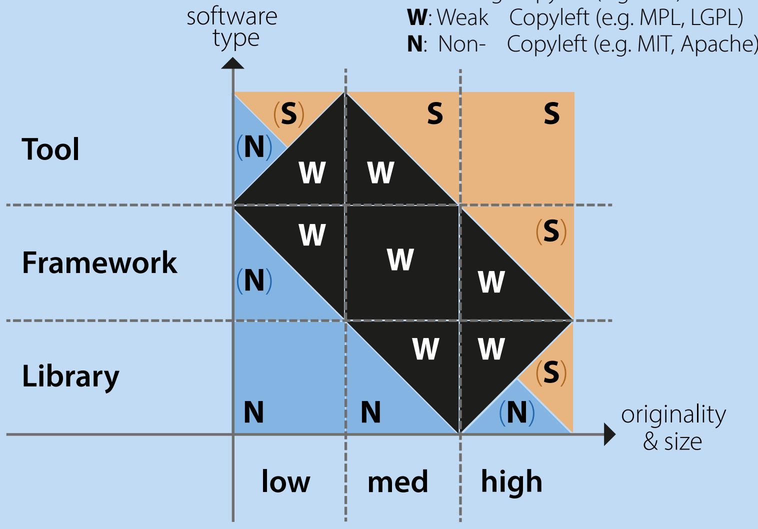
§	**Software Sharing**	Dogmatism Social Equity Politics	Science
@	**Software Hacking**	Fundamentalism Art Hacking	Private
€	**Software Engineering**	Pragmatism Business Engineering	Industry

Most Popular Open Source Licenses

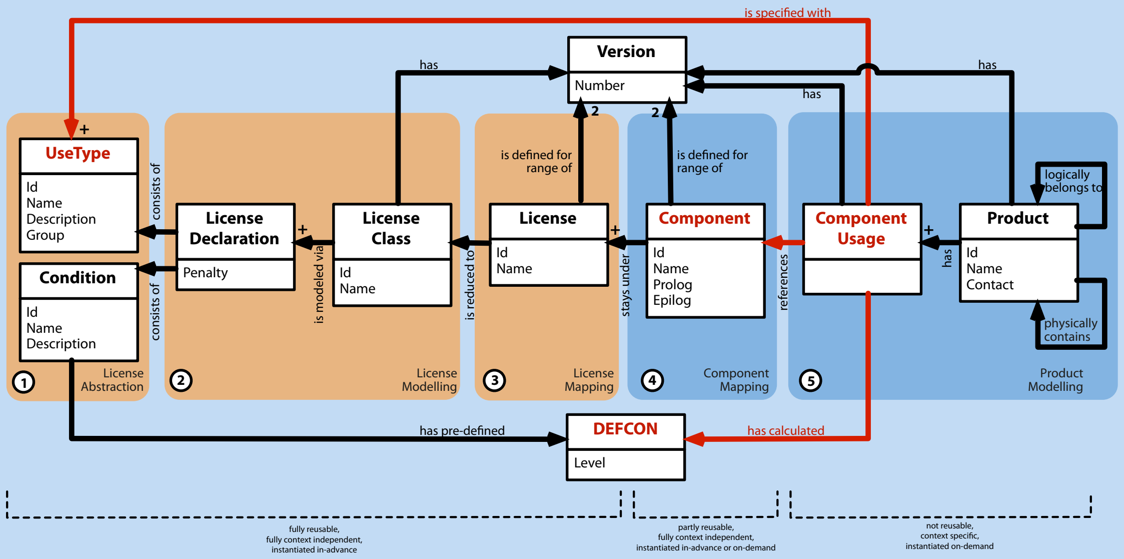


Choosing an Open Source License

S: Strong Copyleft (e.g. GPL)
W: Weak Copyleft (e.g. MPL, LGPL)
N: Non- Copyleft (e.g. MIT, Apache)



License Compliance Checking Meta-Model



ARCHITECTURE FUNDAMENTALS

Back of the Envelope Calculation

TECHNISCHE UNIVERSITÄT MÜNCHEN

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

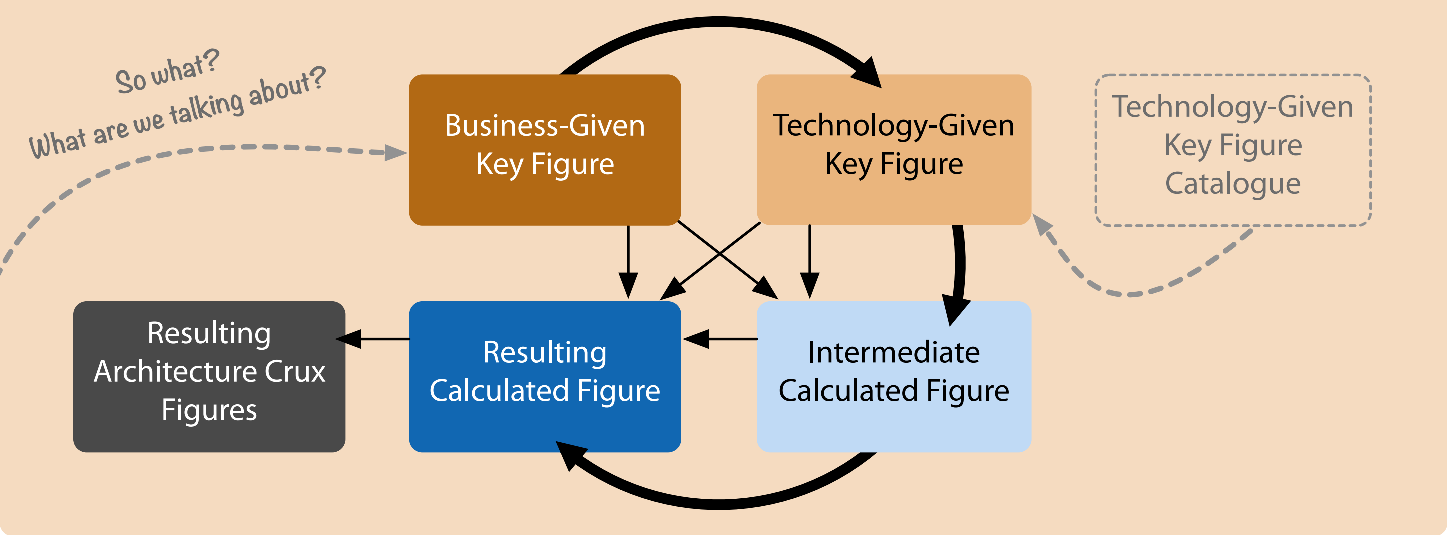
Use Cases 3/3 (query):
- user can view timeline (chronological tweets of others he follows)
- user can search for tweets (by keyword "foo", tag "#foo", or user "@foo")
- user can view tag cloud

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



So what?
What are we talking about?
Business-Given Key Figure
Technology-Given Key Figure
Technology-Given Key Figure Catalogue
Resulting Architecture Crux Figures
Resulting Calculated Figure
Intermediate Calculated Figure

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	1250	MB/s
1000	Mbps	125	MB/s
100	Mbps	12,5	MB/s
10	Mbps	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

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

$R_{j=1..n} = \mathrm{SUM}_{i=1..m}(w_i * E_{i,j})$: Rating j

$R_{best} = \mathrm{MAX}_{j=1..n}(R_j)$: 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.

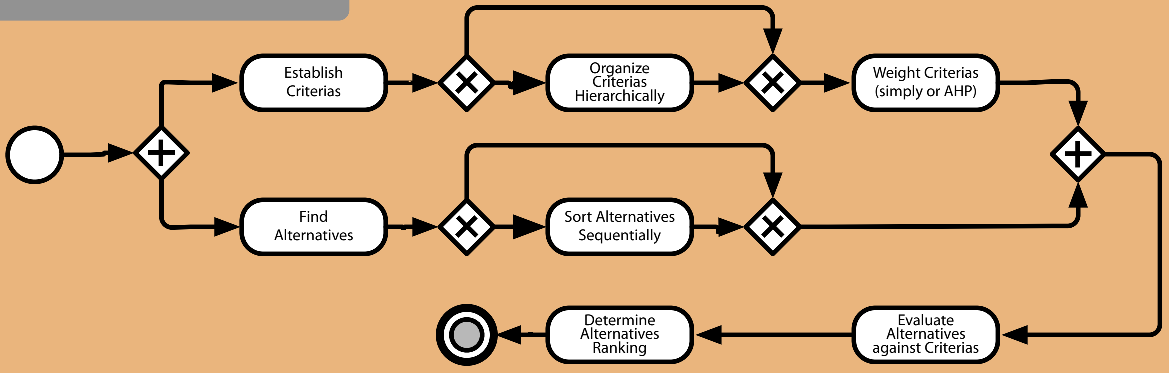
Rule 3: the **best** rating should cover at least **80%** of the **requirements**.

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


Establish Criterias
Organize Criterias Hierarchically
Weight Criterias (simply or AHP)
Find Alternatives
Sort Alternatives Sequentially
Evaluate Alternatives against Criterias
Determine Alternatives Ranking

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

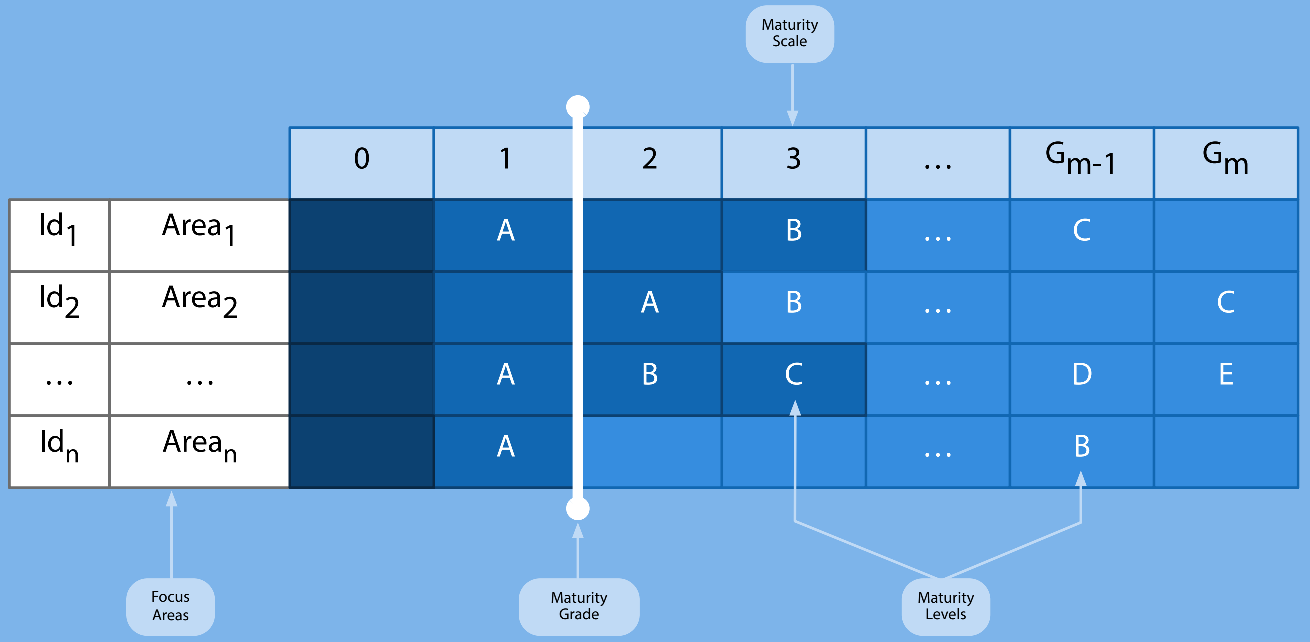
Software Architecture Evaluation:
Meets Functional Requirements
Meets Non-Functional Requirements
Adequate Technology Overhead
Single Dependency Direction
Distance to State of the Art ("modern")
Distance to Most Simple Approach ("adequate")
Distance to Mainstream Approach ("mainstream")
Documented Architecture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

ARCHITECTURE FUNDAMENTALS

Focus Area Maturity Model

TECHNISCHE UNIVERSITÄT MÜNCHEN

Matrix **Structure**



Focus Area Definition

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

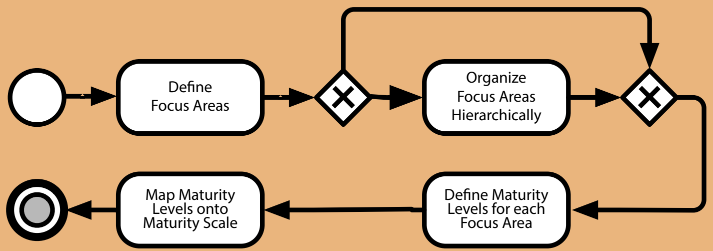
Maturity Level Definition

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>

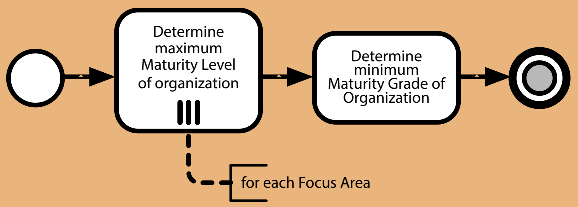
Maturity Level **Prerequisites**

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


Define Focus Areas
Organize Focus Areas Hierarchically
Define Maturity Levels for each Focus Area
Map Maturity Levels onto Maturity Scale

Maturity **Decision** Process


Determine maximum Maturity Level of organization
for each Focus Area
Determine minimum Maturity Grade of Organization

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.