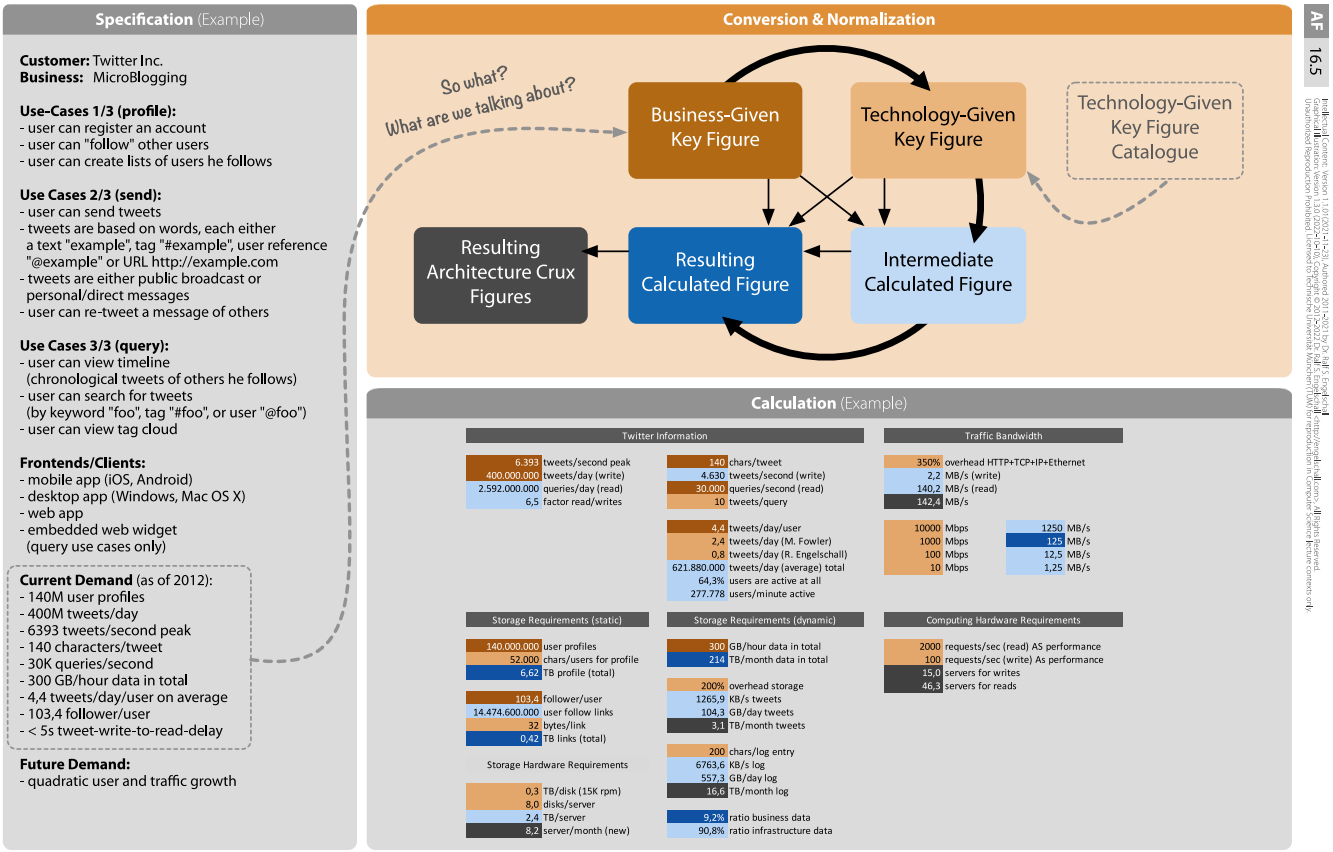


Software Engineering in Industrial Practice (SEIP)

Dr. Ralf S. Engelschall

Back of the Envelope Calculation



Intellectual Content: Version 1.1 (07/2021-12-24) Authored 2011-2021 by Dr. Ralf S. Engelschall (http://engelschall.com). All rights Reserved.
Graphical Illustration: Version 1.3.0 (2022-04-06) Copyright © 2011-2022 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.

Um ein besseres “Gefühl” für den Umfang und den Schwierigkeitsgrad einer zu entwickelnden Architektur zu bekommen, bietet es sich an, eine **Back of the Envelope Calculation** (“Überschlagsrechnung”) zu machen. Hierzu wird methodisch wie folgt vorgegangen: In einem Spreadsheet wird eine zweispaltige Tabelle angelegt, bei der die erste Spalte die Zahl und die zweite Spalte die Einheit aufnimmt.

Nun werden im ersten Schritt die **Business-Given Key Figures**, also die fachlich vorgegebenen bzw. bekannten Zahlen, in die Tabelle als erste Zeilen eingetragen. Sie bekommen die erste Farbe zur Unterscheidung.

Weil einem diese Zahlen üblicherweise noch zu wenig sagen, werden in einem zweiten Schritt verschiedene **Technology-Given Key Figures** als Zeilen eingetragen. Diese werden eventuell aus bestehenden Katalogen entnommen oder stehen über die eigene Erfahrung zur Verfügung. Sie bekommen die zweite Farbe zur Unterscheidung und dienen vor allem als Vergleichsgrößen zu den **Business-Given Key Figures**.

Im dritten Schritt wird man nun üblicherweise sowohl die **Business-Given Key Figures** als auch die **Technology-Given Key Figures** einheitengetreu umrechnen und miteinander vergleichen. Die Zwischenergebnisse, genannt **Intermediate Calculated Figures**, sind Spreadsheet-Zellen mit Formeln, welche die dritte Farbe zur Unterscheidung bekommen.

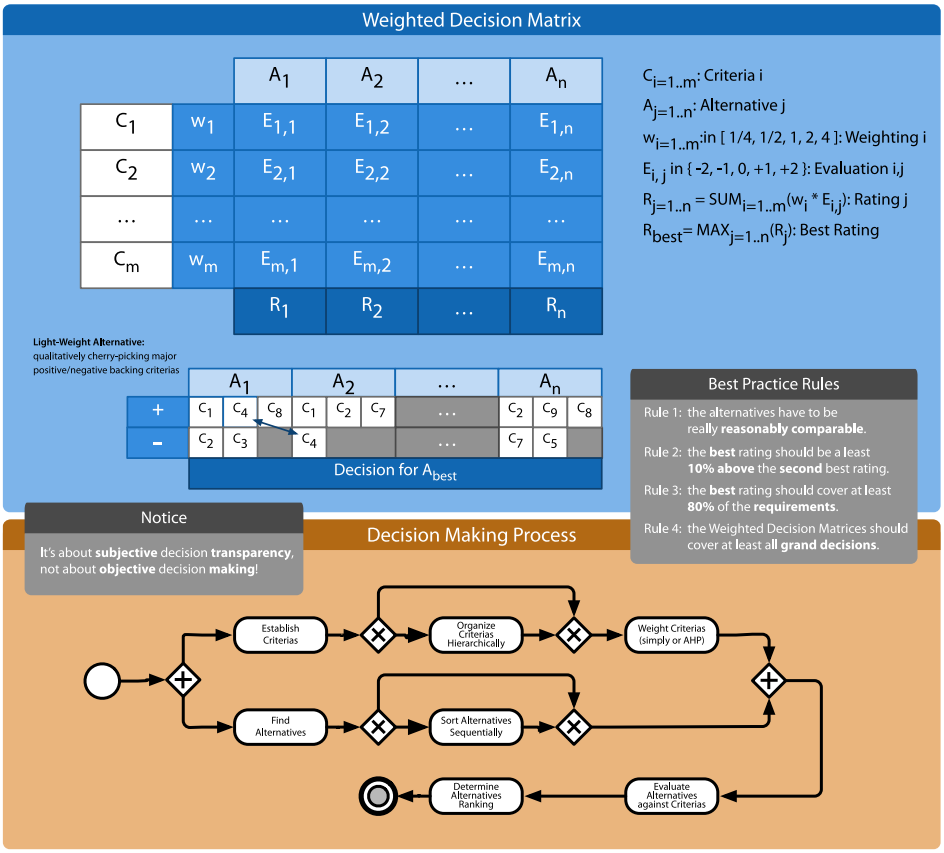
Immer wenn eine **Intermediate Calculated Figure** (ggf. auch schon eine **Business-Given Key Figure** oder eine **Technology-Given Key Figure**) einen entscheidenden Hinweis oder eine Erkenntnis liefert, wechselt man die Zeile auf die vierte Farbe. Sollte diese Erkenntnis potenziell eine wesentliche Relevanz für die spätere Architektur haben (und somit einen Knackpunkt darstellen), wechselt man die Zeile auf die fünfte Farbe, welche die **Resulting Architecture Crux Figure** zeigt.

Anschließend kann man optional die verschiedenen Zeilen zu logischen Gruppen im Spreadsheet bündeln, damit das Spreadsheet übersichtlicher wird.

Fragen

- Welche Methode kann man anwenden, um ein besseres “Gefühl” für den Umfang und den Schwierigkeitsgrad einer zu entwickelnden Architektur zu bekommen?

Weighted Decision Matrix



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)

Intellectual Content: Version 1.1.1 (2022-05-20), Authored 2018-2022 by Dr. Ralf S. Engelschall, Graphical Illustration: Version 1.1.0 (2021-05-26), 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.

Um qualitative Entscheidungen transparent und nachvollziehbar (aber nicht zwingend objektiv) zu treffen und gleichzeitig die Entscheidungsfindung zu dokumentieren, kann man die Methode der **Weighted Decision Matrix** (gewichtete Entscheidungsmatrix) anwenden.

Voraussetzung ist, daß die zu treffende Entscheidung die Auswahl einer von vielen Alternativen ist. Diese trägt man in ein Spreadsheet als Spalten ein. Optional können die Alternativen in eine sinnvolle Reihenfolge gebracht werden.

Dann legt man verschiedene Kriterien fest, welche für die Entscheidung herangezogen werden sollen. Ziel ist es, mit möglichst wenig Kriterien die Alternativen möglichst gut zu unterscheiden. Jedes Kriterium bekommt ein Gewicht. Optional können die Kriterien hierarchisch gruppiert und die Gruppen in eine sinnvolle Reihenfolge gebracht werden.

Anschließend legt man für jedes Kriterium ein Gewicht fest, welches angibt, wie stark das Kriterium in die Entscheidungsfindung einfließt.

Nun werden alle Alternativen gegenüber allen Kriterien bewertet. Bei wenigen Kriterien und vielen Alternativen kann man pro Alternative alle Kriterien bewerten. Bei vielen Kriterien bietet es sich an, pro Kriterium alle Alternativen zu bewerten. Als Bewertungsskala bietet sich -2, -1, 0, +1, +2 an, um eine Mitte (0), positive/negative Bewertungen (+1, -1) und positive/negative Superlative (+2, -2) zu haben.

Zum Schluß wird für jede Alternative die Produktsumme aus der Kriterium-Gewichts-Spalte und der Alternativen-Bewertungs-Spalte berechnet. Die Entscheidung fällt dann für die Alternative mit dem Maximalbetrag in der Produktsumme.

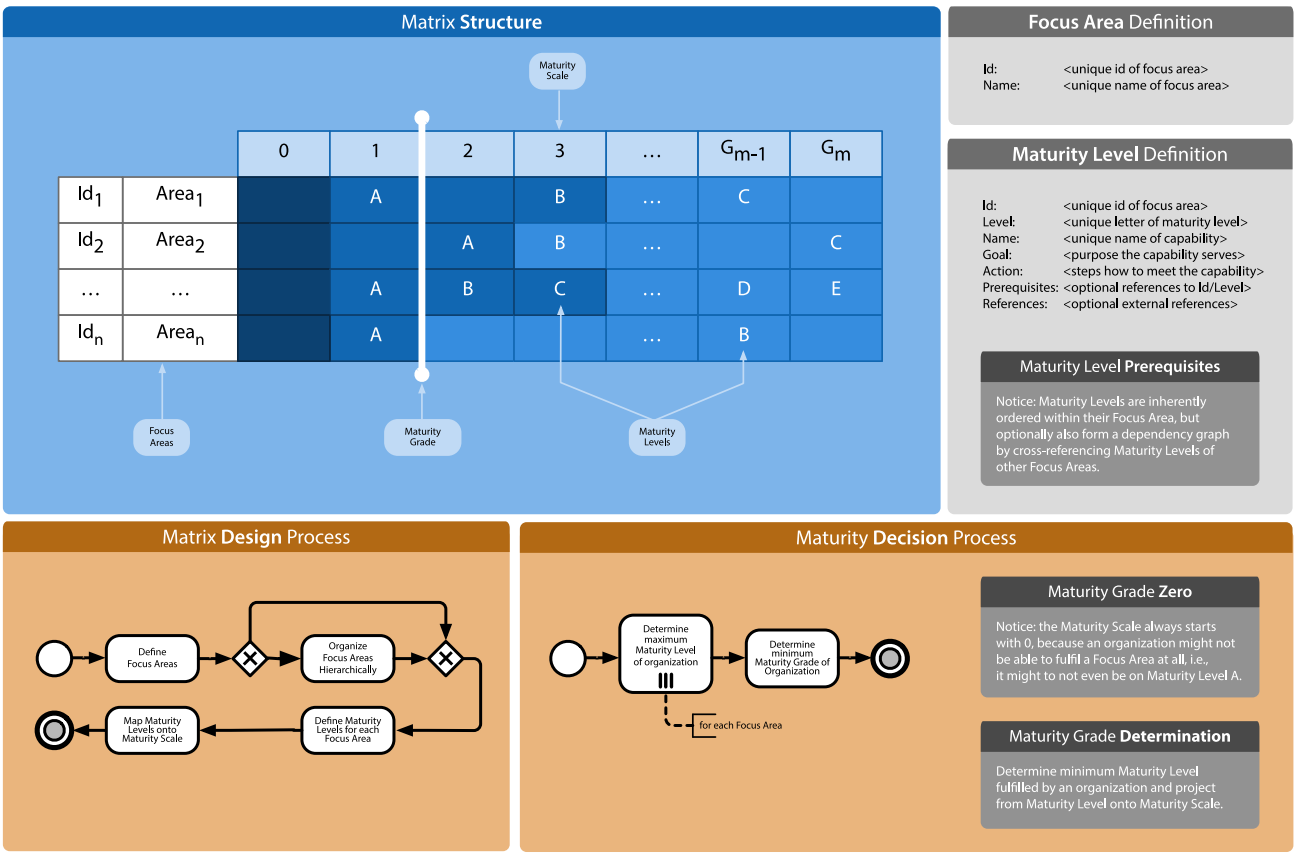
Fragen

? Wie kann man die qualitative Entscheidung für die Auswahl einer aus vielen Alternativen transparent und nachvollziehbar treffen und gleichzeitig dokumentieren?

ARCHITECTURE FUNDAMENTALS

Focus Area Maturity Model

TECHNISCHE UNIVERSITÄT MÜNCHEN



Das Focus Area Maturity Model (FAMM) ist eine Methode, um den Reifegrad einer Organisation bezüglich eines bestimmten Themenbereichs zu bestimmen.

Die Struktur eines FAMM ist eine Matrix aus horizontalen Focus Areas und ihrer Maturity Levels und möglichen vertikalen Maturity Grades auf einer Maturity Scale. Pro Focus Area kann es einen oder eine beliebige Anzahl an Maturity Levels geben und deren Positionierungen auf der Maturity Scale orientieren sich an der Wichtigkeit der Focus Areas und dem Bezug zwischen den Focus Areas und ihren Maturity Levels. Diese Matrix wird in einem ersten Schritt für einen Themenbereich entworfen und ist danach fest.

Für die Bestimmung des Reifegrads einer Organisation wird pro Focus Area überprüft, welchen maximalen Maturity Level die Organisation erfüllt. Der Maturity Grade der Organisation ergibt sich dann aus dem minimalen Maturity Level über alle Focus Areas und der Projektion dieses Maturity Levels auf die Maturity Scale.

Da die Maturity Levels immer oberhalb des Maturity Grade 0 in der Matrix positioniert werden sollten, hat eine Organisation im schlimmsten Fall den Maturity Grade 0, falls sie eine Focus Area gar nicht erfüllt.

Fragen

? Von wem kann man mit dem Focus Area Maturity Model (FAMM) den Reifegrad bestimmen?

Intellectual Content: Version 1.0.1 (2023-12-01), Authored 2023-2023 by Dr. Ralf S. Engelschall
Graphical Illustration: Version 1.0.0 (2023-05-21), 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 contents only.

Big Picture (8-D)

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.

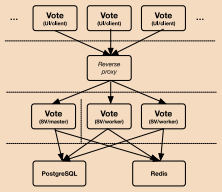
Format: 2xN Table

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.

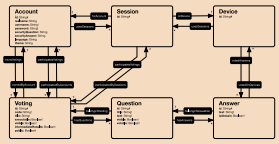
Format: 5xN Table

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 1.0.3 (2020-01-20), Authored 2019-2020 by Dr. Ralf S. Engelschall <http://engelschall.com>, All rights Reserved.
Graphical Illustration: Version 1.0.3 (2020-04-26), 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.

Um sehr früh ein zu erstellendes betriebliches Informationssystem zu verstehen, sollte ein sogenanntes **Big Picture** erstellt werden. Dieses kann man nach der Methode **8-D** von Dr. Ralf S. Engelschall entwickeln, welches aus 8 Dimensionen (4 fachlichen und 4 technischen) besteht.

Im ersten Schritt wird ein **Elevator Pitch** erstellt, der über **Name**, **Purpose**, **Actors** und **Devices** einen groben Überblick über die Lösung liefert. Im zweiten Schritt wird zusätzlich ein **Crux Flash** erstellt, der über **Functionality** und **Cruxes** grob die Funktionalität und deren wesentliche Knackpunkte aufzeigt. Beide Schritte bestehen üblicherweise aus nur sehr wenigen Prosa-Sätzen.

Im dritten Schritt wird eine **Customer Journey** skizziert, die über **Actor Roles** und **Use Cases** beschreibt, welche Anwendungsfälle die verschiedenen Rollen von Akteuren erleben. Diese besteht üblicherweise nur aus einer 2-spaltigen Tabelle.

Im vierten Schritt wird ein **Dialog Storyboard** illustriert, welches über **Dialogs**, **Interactions** und **Control Flow** aufzeigt, welche Benutzeroberfläche die Anwendung bietet. Dieses besteht üblicherweise aus einem Graphen von **Wireframes** (absichtlich sehr groben und unscharfen Skizzen der Dialoge).

Im fünften Schritt werden die **Quality Requirements** aufgelistet, welche über **Qualities** und **Expectations** zeigen, welche Erwartungshaltung bezüglich nicht-funktionaler Qualitätseigenschaften existiert. Diese werden üblicherweise als 2-spaltige Tabelle ausgeführt.

Im sechsten Schritt wird die **System Architecture** illustriert, welche über **Actors**, **Systems**, **Tiers/Areas** und **Programs** zeigt, welche hauptsächlichen Komponenten auf der Ebene von System Architecture existieren werden. Diese besteht üblicherweise aus einem "Boxes'n'Lines"-Diagramm.

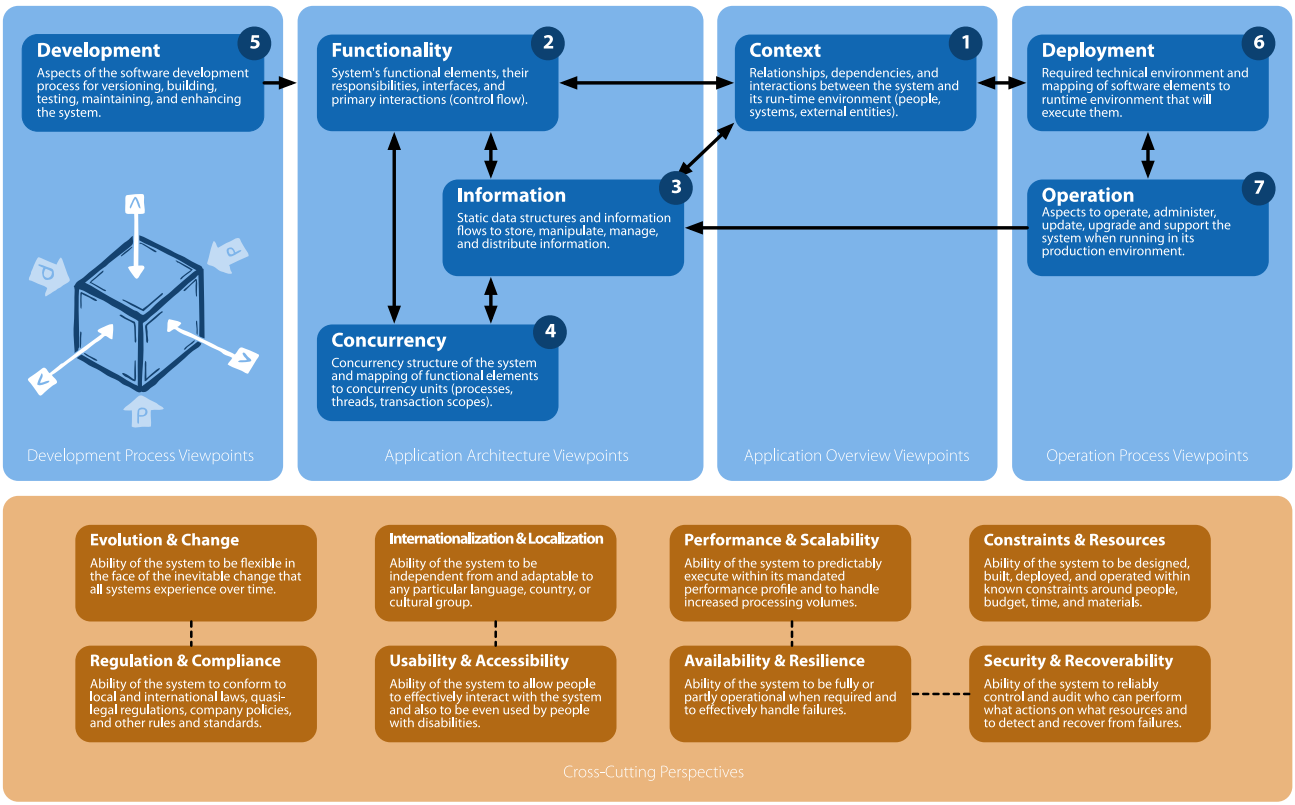
Im siebten Schritt wird das **Data Model** modelliert, welches über **Entities** und **Relationships** zeigt, welche hauptsächlichen fachlichen Klassen und Beziehungen die Daten der Anwendung besitzen. Dieses wird üblicherweise als "UML Class Diagram" repräsentiert.

Im achten und letzten Schritt wird ein **Sizing Sketch** erstellt, welcher über **Aspects**, **Amounts**, **Sizes**, **Total Sizes** und **Units** aufzeigt, welche Größenordnungen (im "worst case") bei den Daten und den Systemkomponenten zu erwarten sind. Dieser besteht üblicherweise aus einer 5-spaltigen Tabelle.

Fragen

- Was sollen die acht Dimensionen des **8-D** Modells leisten?

Viewpoints & Perspectives



Freely available content. Version 1.0.4 (2012-12-15). Authored 2011-12-10 by Dr. Gernot Starke, based on Nick Rozanski, Eoin Woods: "Software Systems Architecture", 2nd ed. 2011. ISBN 978-0-321-71833-4
Graphical illustration, Version 1.0.8 (2012-04-26). Copyright © 2011-2012 Dr. Gernot Starke, http://arc42.de. All rights reserved.
Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

Um eine **Architecture Description** (im deutschen Kontext üblicherweise **IT-Konzept** genannt) für eine Anwendung zu erstellen, dokumentiert man methodisch über **Viewpoints** (Sichten) und **Perspectives** (Perspektiven).

Erstere sind 2-dimensionale Diagramme mit Erklärung eines einzigen ganz bestimmten Sachverhalts. Letztere sind Erklärungen eines übergreifenden Sachverhalts, welche alle von **Nicht-Funktionalen Anforderungen** abstammen.

Wie viele **Viewpoints** und **Perspectives** tatsächlich dokumentiert werden müssen, hängt vom **Concern** der **Stakeholder** ab! Ein Standard-Satz an **Viewpoints** und **Perspectives** ist der folgende.

Bei den **Viewpoints** bietet es sich an, z.B. 7 konkrete **Views** zu dokumentieren: **Context**, **Functionality**, **Information**, **Concurrency**, **Development**, **Deployment** und **Operation**. Die beiden Viewpoints **Functionality** und **Information** sind die beiden wichtigsten, um die Architektur einer Anwendung zu dokumentieren. Sie sollten immer dokumentiert und somit niemals weggelassen werden!

Bei den **Perspectives** bietet es sich an, z.B. 8x2 konkrete **Aspects** zu dokumentieren, welche auf üblichen **Non-Functional Requirements** basieren: **Evolution & Change**, **Regulation & Compliance**, **Internationalization & Localization**, **Usability & Accessibility**, **Performance & Scalability**, **Availability & Resilience**, **Constraints & Resources** und **Security & Recoverability**. Man muss bei den **Perspectives** in der Praxis aber generell alle im Projekt relevanten **Non-Functional Requirements** berücksichtigen und darf sich nicht nur auf diese üblichen beschränken.

Fragen

- Welche zwei **Viewpoints** in der **Architecture Description** einer Anwendung sollten immer dokumentiert werden?

Security Basics

Information Protection Requirements

Confidentiality	**Vertraulichkeit** (germ.)	
Information and systems are available to authorized users only.	Informationen und Systeme stehen nur autorisierten Benutzern zur Verfügung.	
Integrity	**Integrität** (germ.)	
Correctness of information and functionality of systems is given.	Die Korrektheit der Informationen und die Funktionsweise von Systemen ist gegeben.	
Availability	**Verfügbarkeit** (germ.)	
Information and systems are available to authorized users without hindrance.	Informationen und Systeme stehen autorisierten Benutzen ungehindert zur Verfügung.	

Information Protection Assessment

	Confidentiality			Integrity & Availability
	Military:	Gov. (germ.):	Industry:	Recoverable within…
very high	top secret	streng geheim	(secret)	4 hours
high	secret	geheim	(strictly confidential)	1 day
normal	confidential	vertraulich	confidential	1 week
low	(restricted)	nur für den Dienstgebrauch	internal	2 week
none	(public)	(öffentlich)	public	N.A.

Privacy by Design Aspects

TRY Processing Transparency	**DIF Differential Privacy**
Full transparency of all person-related data processing actions and the rationales for the processing.	Introduced randomness into the results of queries on underlying confidential or person-related data to ensure ambiguity in the deduced underlying data.
CST Processing Consent	**STR Secure Storage**
Explicit consent from the data owner for the data processing, including the right to contradict the consent at any later time.	Secure storage of all person-related information to prevent data loss or exposure while the information is at rest.
MIN Data Minimization	**TRS Secure Transfer**
Minimization of the gathered, processed and stored person-related data.	Secure transfer of all person-related information to prevent data loss or exposure while the information is transferred.
PSN Data Pseudonymization	**ERA Right to Erasure**
Substitute person-related information with artificial identifiers (pseudonyms) while still keeping information for backtracking to the original information.	Ensure the data owners, at any time, have the right and operational means to lookup and optionally erase all their person-related information.
ANO Data Anonymization	**PRT Right to Portability**
Substitute person-related information with artificial identifiers without keeping any information for backtracking to the original information.	Ensure the data owners, at any time, have the right and operational means to lookup and optionally export all their person-related information.
AGG Data Aggregation	**NOT Breach Notification**
Always report person-related statistical information only as aggregated numbers which each does not cover less than about 10 involved persons.	Transparently provide notifications in case of any protection breach of confidential or person-related information.

Intellectual Content, Version 1.0.0 (2018-10-31), Authored 2018 by Dr. Ralf S. Engelschall, Graphical Illustration, Version 1.0.0 (2018-10-31), Copyright © 2018 Dr. Ralf S. Engelschall <http://engelschall.com> – All Rights Reserved. Unauthorized Reproduction Prohibited. Licenced to Technische Universität München (TUM) for reproduction in Computer Science lecture contents only.

Beim Schutzbedarf unterscheidet man drei Anforderungen (**Information Protection Requirements**): Vertraulichkeit (**Confidentiality**) sorgt dafür, daß Informationen und Systeme nur autorisierten Benutzern zur Verfügung stehen; Integrität (**Integrity**) sorgt dafür, daß die Korrektheit der Informationen und die Funktionsweise von Systemen gegeben ist; und Verfügbarkeit (**Availability**) sorgt dafür, daß Informationen und Systeme autorisierten Benutzern ungehindert zur Verfügung stehen.

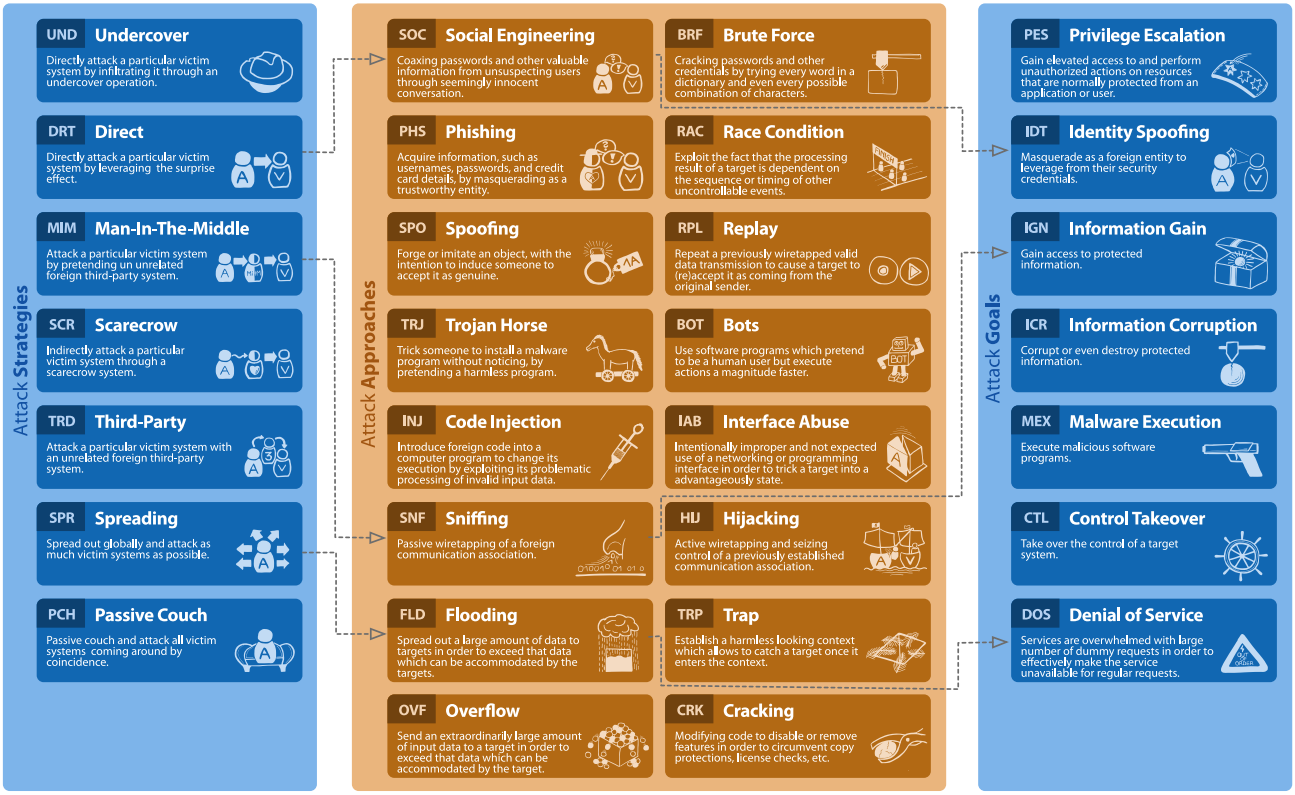
Bei einer Schutzbedarfsfeststellung (**Information Protection Assessment**) werden die Informationen und die Systeme bezüglich dieser drei Anforderungen in verschiedene Strengen klassifiziert und attributiert (**very high**, **high**, **normal**, **low**, **none**).

Bei **Privacy by Design** sollen verschiedene sicherheitsrelevante Aspekte direkt bei der Konstruktion (**Design**) der Lösung berücksichtigt werden. Diese sind teilweise sogar durch Gesetze gefordert (z.B. Transparenz in der Verarbeitung, Daten-Sparsamkeit, Recht auf Vergessen und Meldepflicht bei Verletzungen durch die DSGVO).

Fragen

- Welche drei Anforderungen unterscheidet man beim Schutzbedarf von Informationen und Systemen?

Security Attacks



Attack = Strategy x Approach x Goal

Den Raum von Sicherheitsangriffen (**Security Attacks**) kann man über drei Dimensionen aufspannen: welche Strategie (**Strategy**) der Angreifer wählt, welchen Ansatz (**Approach**) der Angreifer wählt und welches Ziel (**Goal**) der Angreifer verfolgt. Jeder Angriff ist eine konkrete Kombination dieser drei Dimensionen.

Für einen **Identity Spoofing** Angriff wählt ein Angreifer beispielsweise als Strategie einen direkten Angriff (**Direct**) von Personen, bei dem über den Ansatz des **Social Engineering** versucht wird, an persönliche Informationen zu gelangen.

Für einen **Information Gain** Angriff wählt ein Angreifer beispielsweise eine **Man-In-The-Middle** Strategie, bei der er sich in eine fremde Kommunikation einklinkt und dort über den passiven Ansatz des **Sniffing** Informationen abgreift.

Für einen **Denial of Service** Angriff wählt ein Angreifer beispielsweise die **Spreading** Strategie, bei der er über den **Flooding** Ansatz versucht ein Zielsystem mit Anfragen zu überlasten.

Fragen

? Über welche drei Dimensionen kann man Sicherheitsangriffe (**Security Attacks**) verstehen?

Intellectual Content: Version 1.0.2 (2014-05-01), Authored 2013-2014 by Dr. Ralf S. Engelschall <http://engelschall.com>. All Rights Reserved.
Graphical Illustration: Version 1.2.1 (2020-06-25), Copyright © 2013-2020 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.

Security Defenses





Intellectual Content: Version 1.0.2 (2012-10-17). Authored 2012 by Dr. Ralf S. Engelschall. Graphical Illustration: Version 1.0.4 (2020-04-23). Copyright © 2012–2020 Dr. Ralf S. Engelschall. http://creativecommons.org/licenses/by-nc-nd/2.0/ All rights reserved. Unauthorized Reproduction Prohibited. Licensed to Technische Universität München (TUM) for reproduction in Computer Science lecture contexts only.

Es gibt zahlreiche Sicherheitsverteidigungen (**Security Defenses**). Diese zerfallen in der Praxis allerdings in zwei Bereiche: solche, die auf der Ebene der Netzwerkprotokolle (**Network Protocols**) ansetzen, und solche, die auf der Ebene der Anwendung (**Software Application**) ansetzen.

Auf der Ebene der Netzwerkprotokolle (**Network Protocols**) sind klassische Verteidigungsmaßnahmen der Einsatz eines **Secure Network Protocol** wie z.B. des sehr weit verbreiteten **Transport Layer Security (TLS)**, der Einsatz eines **Protocol Proxy** wie z.B. NGINX oder PGProxy, die **Peer Identification**, **Peer Authentication** und **Digital Signatures** über X.509-Zertifikate und das **Peer Whitelisting** über eine Access Control List (ACL).

Auf der Ebene der Anwendung (**Software Application**) sind klassische Verteidigungsmaßnahmen die Vorschrift einer **Strong Password Policy** mit z.B. einer Mindestlängenangabe, der Einsatz einer **Role-Based Authorization** wie z.B. Linux RBAC und das **Execution Sandboxing** mit z.B. virtuellen Maschinen wie JVM oder Containern wie Docker.

Fragen

- Nennen Sie ein sehr weit verbreitetes **Secure Network Protocol**, welches auf der Ebene der Netzwerkprotokolle einer der üblichen **Security Defenses** ist!