HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Visualisierung und Umsetzung von Web-API Design Patterns

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2017

Autoren: Sebnem Kaslack, Nicolas Dipner
Betreuer: Prof. Dr. Olaf Zimmermann

Abstract

Ein Autorenteam der HSR erarbeitet aktuell zusammen mit Kooperationspartnern die Patternsprache *Interface Representation Patterns (IRP)* für Design, Programmierung und Evolution messagebasierter Programmierschnittstellen. Ziel dieser Studienarbeit war es, IRP zu visualisieren sowie Known Uses der Patterns in öffentlichen Web-APIs zu finden sowie diese zu analysieren, zu verifizieren und die daraus gewonnenen Erkenntnisse zum Vorkommen von Pattern-Varianten zu dokumentieren und dem IRP-Autorenteam zur Verfügung zu stellen. Das Projektteam (i) untersuchte insgesamt sechs öffentlich zugängliche Web-APIs aus den Kategorien soziale Netze, Kalender, Bezahlung und Software Engineering Tools, (ii) dokumentierte die Rechercheergebnisse detailliert und (iii) synthetisierte daraus eine vergleichende Auswertung sowie einen Überblick. Der Hauptfokus der Arbeit lag im Ausarbeiten von Visualisierungen zur Illustration der Patterns. Das Projektteam entschied sich, ein Visualisierungskonzept als Grundlage für den Designprozess auszuarbeiten. Dieses diente als Leitfaden zur einheitlichen Gestaltung der Visualisierungen während der Entwurfs- und Umsetzungsphase. Die Visualisierungen wurden anhand von vordefinierten User Stories

und detaillierten nicht-funktionalen Anforderungen auf ihre Verständlichkeit und Verwendbarkeit geprüft und aufgrund des Feedbacks externer Experten iterativ und inkrementell verbessert. Mit Hilfe der Pattern-Illustrationen kann ein IRP-Benutzer eigene Grafiken nach Belieben zusammensetzen. Das Visualisierungskonzept kann künftig bei der Entwicklung von Visualisierungen in ähnlichen Projekten zur Unterstützung herbeigezogen werden.

Management Summary

Ausgangslage

Das Akronym API steht für «Application Programming Interface». Ein API ermöglicht den Austausch von Daten bzw. die Kommunikation zwischen zwei Systemen. Will eine Organisation das eigene Software-System anderen Applikationen zur Verfügung stellen, wird diese Anbindung über ein API realisiert. Mit Hilfe der Schnittstelle wird externen Entwicklern die Möglichkeit geboten, die zur Verfügung gestellten Daten und Funktionen dynamisch in die eigene Applikation zu integrieren.

APIs sind zentraler Bestandteil verteilter IT-Systeme. In der Literatur sind viele Best Practices zum Aufbau von APIs zu finden, es fehlt jedoch ein einheitliches, breit akzeptiertes Vokabular, wie man ihn aus anderen Software Engineering Bereichen wie beispielsweise Cloud Computing oder Enterprise Integration und Messaging kennt.

Mit dem Ziel eine einheitliche Sprache zur Gestaltung von APIs im Bereich der Webapplikationen zu entwickeln, arbeiten Pattern-Autoren zurzeit an der plattformübergreifenden Pattern-Sprache «Interface Representation Patterns» (IRP).

Ein Pattern entsteht aus erfolgreichen Engineering-Projekten und beschreibt einen generischen Lösungsvorschlag, welcher ein immer wiederauftretendes Problem adressiert und sich in der Anwendung bewährt hat. Patterns unterstützen Entwickler in der Gestaltung ihrer Softwarelösung, in dem sie die Sachlage der Problemdomäne aufgreifen, Lösungen benennen und beschreiben sowie Ursachen wie auch Konsequenzen der Verwendung dieser Lösungen anhand von Designkriterien beschreiben.

Ziel der Arbeit ist es, die IRPs mit geeigneten Elementen zu visualisieren sowie Vorkommnisse der Patterns in populären Web-APIs zu untersuchen.

Vorgehen

In einem ersten Schritt analysierte das Projektteam bereits schriftlich dokumentierte Patterns und deren Einsatz in bekannten Web-APIs. Zeitgleich mit den ersten Visualisierungsentwürfen wurde ein Visualisierungskonzept ausgearbeitet, welches als Grundlage für die Gestaltung der Illustrationen dient. In diesem Konzept sind unter anderem die Dimensionen der Icons und ihrer Komponenten wie auch Gestaltungsrichtlinien definiert. Es beschreibt ebenfalls, was die Illustrationen zum Ausdruck bringen sollen und wie mögliche Kompositionen der Icons aussehen können.

Die Visualisierungen entstanden durch den Einsatz des vektorbasierten Zeichnungstools «Adobe Illustrator». Während der Umsetzung der Visualisierungen achtete das Projektteam auf eine detaillierte Strukturierung der Illustrationen, um eine spätere Bearbeitung durch Dritte zu vereinfachen. Der Prozess von Pattern-Analyse, Dokumentation und Visualisierung wurde mehrmals durchlaufen, da pro Iteration neue Patterns aus dem «Pattern Backlog» des IRP-Autorenteams hinzukamen. Rückmeldungen zu den Visualisierungen wurden fortlaufend in-

tern und vom Auftraggeber bei externen Testpersonen eingeholt. Bestehende Visualisierungen wurden anschliessend aufgrund des internen und externen Feedbacks iterativ und inkrementell verbessert.

Ergebnisse

Diese Studienarbeit brachte eine Palette von 50 Visualisierungen der Pattern-Sprache hervor, die für die jeweiligen Pattern-Beschreibungen eingesetzt werden können. Der Auftraggeber kann anhand der ausgearbeiteten Grafiken und Komponenten selbst weitere Visualisierungen entwerfen. Geübte Anwender können sämtliche Komponenten der Visualisierungen mit einem vektorbasierten Zeichnungstool bearbeiten.

Die Illustrationen sind für die Verwendung am Whiteboard ausgelegt. Ausserdem stehen sie als einfache Bilder, für die Verwendung in Präsentationswerkzeugen wie Microsoft PowerPoint oder auf Webseiten zur Verfügung.

Die Rechercheerkenntnisse zu den entsprechenden Patterns wurden schriftlich festgehalten. Das erarbeitete Recherche-Dokument wird von den Pattern-Autoren genutzt, um einerseits «Known-Uses» in die Pattern-Sprache aufnehmen zu können sowie über die bisherige Ausarbeitung der Patterns zu reflektieren und weitere Schritte zu planen.

Die Visualisierungen im derzeitigen Stand können für die Pattern-Beschreibungen genutzt und bei Bedarf einfach angepasst werden. Die Illustrationen sollen Dritten ermöglichen, das Wesentliche eines API schnell zu erfassen und zu behalten.

Inhaltsverzeichnis

Teil 1

Hauptbericht

Kapitel 1

Einleitung

1.1 Projektübersicht

Das Projektteam setzt sich in dieser Studienarbeit mit den IRP [11], plattformunabhängigen Design- und Strukturierungsvorschlägen für API Technologien, auseinander. Basierend auf bereits erarbeiteten und dokumentierten Patterns, werden Grafiken ausgearbeitet, um die IRP visuell darzustellen. Durch Zusammensetzen der verschiedenen Grafiken soll es möglich sein, die Funktionsweise dieser Patternsprache zu illustrieren. Hierfür wurde ein Visualisierungskonzept (siehe Seite 15) mit Themen wie Designvorschriften, technische Aspekte, Kompositionen und Einflüsse ausgearbeitet. Sowohl das Projektteam als auch der Auftraggeber prüften die fertiggestellten Visualisierungen in verschiedenen Phasen auf deren Verständnis und Eignung durch Feedback von Testpersonen.

Nebst der Erstellung von Grafiken für die jeweiligen Patterns setzte sich das Projektteam detailliert mit dem Vorkommen der IRP in sechs öffentlichen Web-APIs (siehe Seite 53) aus verschiedenen Kategorien (Soziale Netze, Bezahlung, Tools, Kalender) auseinander. Während der Analyse wurden API-Aufrufe getestet und ein wesentliches Augenmerk auf *Request* und *Response* Nachrichten sowie den *Payload* gelegt, die während einer Kommunikation zwischen einem API Konsumenten und einem API Anbieter ausgetauscht werden. Durch ein Mapping der jeweiligen API-Funktion auf das Pattern wurde die Handhabung der Pattern-Entwürfe geprüft. Die Funktionsweise der API-Aufrufe und die gewonnenen Erkenntnisse wurden anschliessend als Feedback für die Pattern-Autoren festgehalten.

Diese Arbeit beinhaltet keine umfassende Beschreibung der IRP. Hierfür kann die von Zimmermann et al. erstellte Pattern-Übersicht [11] konsultiert werden.

1.2 Zielgruppe

Diese Arbeit richtet sich an Softwareentwickler und -architekten, die (i) sich über die Entstehung der Visualisierungen der Icon-Illustrationen informieren möchten, (ii) wissen wollen, welche Prozesse bei einem Entwurf einer Pattern-Visualisierung durchlaufen werden oder (iii) sich einen Überblick über die Web-API Rechercheergebnisse, die im Rahmen dieser Studienarbeit zusammengefasst wurden, verschaffen möchten.

1.3 Inhalt

Die nachfolgenden Kapitel beschäftigen sich einerseits mit Visualisierungen von Patterns, andererseits werden die Ergebnisse der Web-API Recherche in Bezug auf die IRP zusammengefasst.

Die verschiedenen Themengebiete sind in Kapitel gegliedert.

- ☐ **Kapitel 2** umfasst wichtige Grundlagen mit Referenzangaben.
- ☐ **Kapitel 3** beginnt mit einem Überblick über die bestehenden IRP und zeigt die Zugehörigkeit der Patterns zu der jeweiligen Pattern-Kategorie.
- ☐ **Kapitel 4** beschreibt das Visualisierungskonzept, in welchem unter anderem (i) die Einflüsse für die Gestaltung der Grafiken, (ii) die Designrichtlinien und (iii) die technischen Aspekte näher erläutert werden.
- ☐ **Kapitel 5** beschreibt das allgemeine Vorgehen beim Entwurf von Visualisierungen und zeigt auf, welche Schritte durchlaufen wurden, bis eine Visualisierung den finalen Status erreicht hat.
- ☐ **Kapitel 6** zeigt die Umsetzung der Visualisierungen und liefert Kompositionsbeispiele.
- ☐ **Kapitel 7** beschreibt die Testarten, die für die Bewertung der Grafiken angewandt wurden und zeigt die Ergebnisse von Befragungen in tabellarischer Form auf.
- ☐ **Kapitel 8** stellt die Ergebnisse der Web-API Recherche vor, mit Bezug auf die IRP und deren Vorkommen in diesen APIs.
- ☐ **Kapitel 9** gibt einen Überblick und Hintergrundinformationen für Folgearbeiten.
- ☐ **Kapitel 10** beinhaltet einen Rückblick über die Studienarbeit.

Kapitel 2

Grundlagen

Dieses Kapitel beschreibt grundlegende Begriffe, welche in diesem Dokument verwendet werden. Es dient dem Zweck dem Leser ein gewisses Verständnis zu vermitteln und Unklarheiten im Vorfeld zu beseitigen.

2.1 Interface Representation Patterns (IRP)

Die IRP sind ein essentieller Bestandteil dieser Arbeit. Sie beschreiben plattformunabhängige Designvorschläge für verschiedene API Technologien wie RESTful http, Web Services (WSDL/SOAP) und WebSockets.

Zimmermann et al. zeigen auf, aus welchen Überlegungen diese Patterns entstanden sind. [11] Die Autoren erläutern, dass aktuell keine allgemein gültigen Designvorschläge beziehungsweise Patterns für das strukturiere Design von Request und Response Nachrichten in nachrichtenbasierten APIs vorhanden sind. Zimmermann et al. greifen diese Situation auf und arbeiten an einer Patternsprache als Lösungsvorschlag.

2.2 RESTful HTTP

REST steht für Representational State Transfer. In Kapitel 8 wird auf die Rechercheergebnisse der auf die IRP untersuchten Web-APIs eingegangen. Diese zeichnen sich durch eine RESTful HTTP-Schnittstelle aus. S. Allamaraju gibt Designvorschläge, wie solche Schnittstellen aufgebaut werden und eingesetzt werden können. [1] Die Umsetzung dieser Web Services basiert auf dem HTTP Protokoll. S. Allamaraju geht auf die HTTP Methoden ein (GET, HEAD, OPTIONS, PUT, DELETE) und erläutert deren Einsatz mit RESTful Web Services.

2.3 Application Programming Interface (API)

API ist eine Schnittstelle, die den Austausch von Daten beziehungsweise die Kommunikation zwischen zwei System ermöglicht. K. Spichale erläutert die verschiedenen API-Typen (Objektorientierte API, REST-API, Messaging-API und Dateibasierte API) [7] und definiert den Begriff API wie folgt: «Eine API kann man allgemein definieren als ein Programmteil, das von einem

Softwaresystem anderen zur Anbindung zur Verfügung gestellt wird.[1] Diese Definition betont korrekterweise den Integrationszweck einer API. Eine API beschreibt aber auch die möglichen Interaktionen, mit denen sie verwendet werden kann. Deswegen gehört zu einer API eine detaillierte Dokumentation der Schnittstellenfunktionen mit ihren Parametern.»

2.4 Pattern

Ein Pattern entsteht aus erfolgreichen Engineering-Projekten und beschreibt einen generischen Lösungsvorschlag, welcher ein immer wieder auftretendes Problem adressiert und sich in der Anwendung bewährt hat. Patterns unterstützen Entwickler in der Gestaltung ihrer Softwarelösung, indem sie die Sachlage der Problemdomäne aufgreifen, benennen und Ursachen wie auch Folgen des Einhaltens vorgeschlagener Designkriterien beschreiben.

Im Buch «Pattern-Oriented Software Architecture Volume 1» wird die Definition des Begriffs «Pattern» treffend beschrieben:

«Patterns help you build on the collective experience of skilled software engineers. They capture existing, well-proven experience in software development and help to promote good design practise. Every pattern deals with a specific, recurring problem in the design or implementation of a software system. Patterns can be used to construct software architectures with specific properties.» [3]

[1] https://de.wikipedia.org/wiki/Programmierschnittstelle

Kapitel 3

Pattern-Übersicht

Dieses Kapitel zeigt auf, wie die IRP strukturiert sind. Diese Einführung soll den Einstieg in das Kapitel 4 «Visualisierungskonzept» vereinfachen.

Abbildung 3.1 gibt einen Überblick über die aktuell bestehenden Pattern-Kategorien.

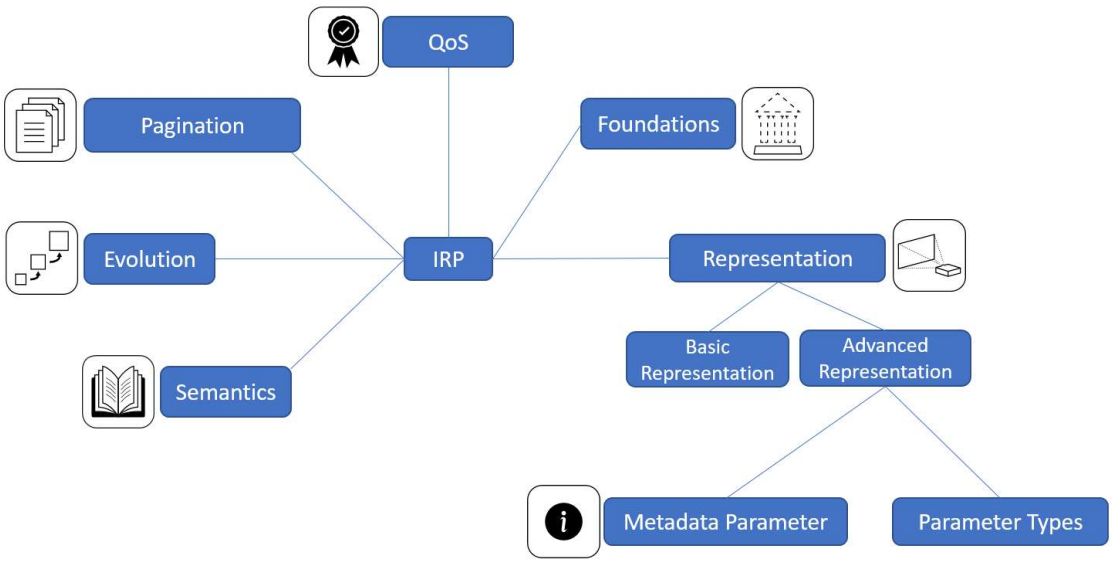
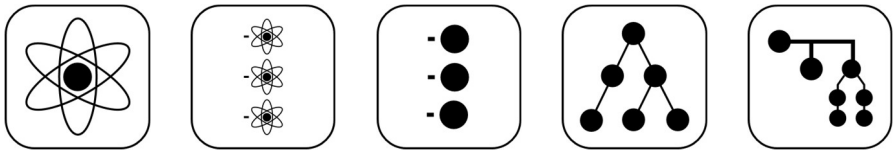


Abbildung 3.1: Pattern-Kategorien Übersicht

Das Projektteam hat in der Studienarbeit verschiedene Patterns aus diesen Kategorien analysiert und in einem weiteren Schritt für diese Patterns eine Visualisierung erarbeitet. Nebst den Visualisierungen wurden diese Patterns auch in öffentlichen Web APIs (Facebook Graph API, Google Calendar API, Stripe API, YouTube Data API v3, GitHub REST API v3, Twitter REST API) auf deren Vorkommen recherchiert und die Ergebnisse daraus dokumentiert (siehe Kapitel 8 «Rechercheergebnisse»).

Auf den nachfolgenden zwei Seiten werden die einzelnen Patterns unter der jeweiligen Pattern-Kategorie aufgeführt. Diese Gliederung hat sich im Verlaufe des Projektverlaufes ergeben und kann sich zu einem späteren Zeitpunkt ändern.

Basic Representation Patterns



Atomic Parameter | Atomic Parameter List | Parameter List | Parameter Tree | Parameter Comb

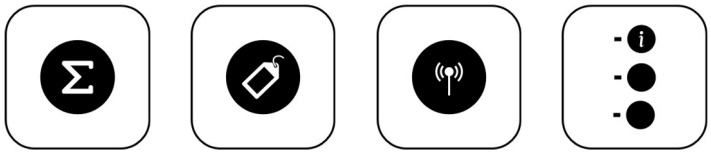
Pagination Patterns

Cursor / Token Based Pagination | Page Based Pagination | Time Based Pagination

Advanced Representation Patterns

Wish List | Wish Object | Metadata Parameter

Metadata Parameter

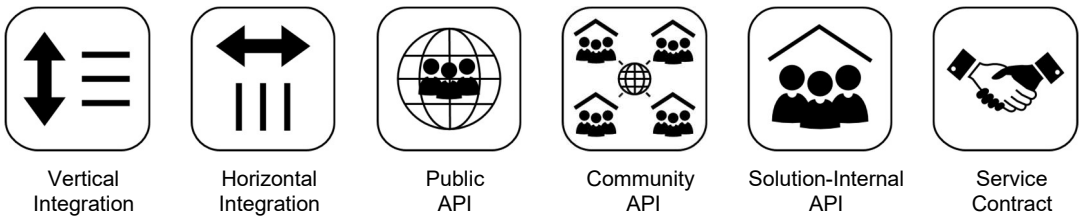


Aggregated Metadata Parameter | Provenance Metadata Parameter | Control Metadata Parameter | Annotated Parameter List

Parameter Types

ID Parameter | Entity Parameter | Link Parameter

Foundation Patterns

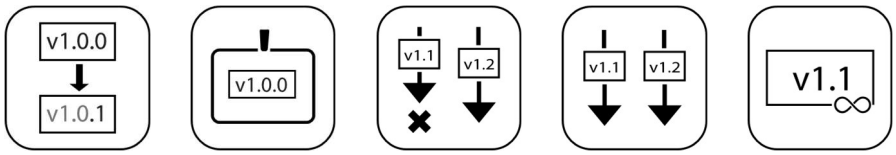


Vertical Integration | Horizontal Integration | Public API | Community API | Solution-Internal API | Service Contract

QoS Patterns

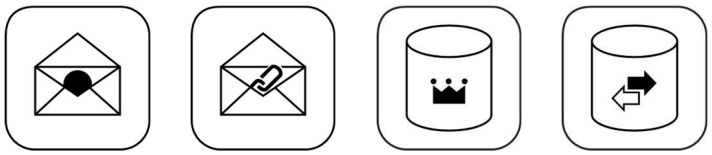
API Key | Rate Limit | Error Reporting | SLA-SLO | Request Bundle

Evolution Patterns



Semantic Versioning | Version Identifier | Aggressive Deprecation | Two in Production | Lifetime Guarantee

Semantic Patterns



Embedded Reference Data | Linked Reference Data | Master Data | Transaction Data

Kapitel 4

Visualisierungskonzept

4.1 Einleitung

Das Visualisierungskonzept dient als Grundlage für den Designprozess. In diesem Kapitel werden die Rahmenbedingungen für die Visualisierungen festgelegt. Das Konzept beschreibt sowohl die Erwartungen des Auftraggebers an das Endprodukt, wie auch praktische Richtlinien zur Umsetzung und Überlegungen im Zusammenhang mit dem Entwurfsprozess.

Dieses Dokument entstand in einem iterativen Prozess. Das Projektteam erfasste Anforderungen und leitete daraus Richtlinien für die Entwurfsphase ab. Die Richtlinien wurden während des Visualisierungsprozesses fortlaufend geprüft und allenfalls angepasst, wenn dies als sinnvoll erachtet wurde und nicht gegen die Anforderungen verstiess. Die finale Struktur erhielt das Konzept während des Erstellens der Schlussdokumentation.

4.1.1 Ausgangslage und Motivation

Während zu Beginn des Projektes unklar war, in welcher Form die Ergebnisse der Arbeit dokumentiert werden, stand für den Auftraggeber wie auch für das Projektteam fest, dass ein Leitfaden nötig sein würde, um aussagekräftige Visualisierungen zu gestalten. Die Wichtigkeit von visuellen Repräsentationen im Software-Engineering Bereich wird unter anderem im Artikelvon Moody [5] beschrieben. Dieses Paper befasst sich mit den Zielen und Prinzipien guter visueller Gestaltungen und bildet somit eine solide Grundlage für Visualisierungen im technischen Bereich. Der Wunsch nach begründeten Designentscheiden führte schlussendlich zu diesem Visualisierungskonzept. Sämtliches Gedankengut, welches Einfluss auf die Visualisierungen nimmt, soll aus Praxisgründen festgehalten werden.

4.1.2 Ziel und Zweck

Das Konzept soll dem Leser transparent aufzeigen, auf welcher Basis die Visualisierungen der IRP entstanden sind. Durch das Einhalten der im Dokument festgelegten Richtlinien ist eine konsistente Darstellung der verschiedenen Visualisierungskomponenten gewährleistet. Die Patternsprache mit ihren Illustrationen ist nach aktuellem Stand nicht abgeschlossen, folglich

ist die Dokumentation der gemachten Überlegungen im Rahmen dieses Projektes eine wertvolle Ressource für weiterführende Arbeiten.

4.2 User Stories

Die folgenden drei User Stories beschreiben Kernaufgaben der Anwender, welche durch die Verwendung der zu erstellenden Visualisierungen umgesetzt werden sollen. Die User Stories eignen sich für Testprozeduren, in welchen die Visualisierungsentwürfe auf ihre Verwendbarkeit geprüft werden sollen.

- Als **API-Designer** möchte ich meine API-Entwürfe und Lösungen mithilfe der IRP-Visualisierungen plattformunabhängig darstellen. Die Visualisierungen sollen mich, als Werkzeug zur Modellierung vor, während und nach dem effektiven Designprozess, unterstützen, um den API-Konsumenten eine in sich konsistente, robuste Schnittstelle bereitzustellen.
- Als **Applikationsentwickler** möchte ich die Funktionsweise eines APIs, ohne Implementierungsdetails, durch die IRP und den dazugehörenden Visualisierungen schnell verstehen können, um das API mit anderen Anbietern zu vergleichen und den persönlichen Nutzen der Funktionalitäten zu erkennen.
- Als **übergeordneter IT-Architekt**, der verschiedene APIs untersucht, muss ich ohne plattformspezifisches Wissen die essentiellen Eigenschaften eines APIs auf Anhieb verstehen können, um diese potentiellen Anwendern zur Verfügung zu stellen und in der Planung zukünftiger Projekte zu berücksichtigen.

Der folgende Abschnitt hat einen starken Einfluss auf die Gestaltung der Icons und beschreibt die Anforderungen des Umfelds an die Visualisierungen. Er steckt somit den Rahmen ab, in welchem sich das Projektteam während dem Designprozess bewegt.

4.2.1 Benutzercharakteristik

Die Visualisierungen ergänzen die Entwicklung der IRP und richten sich somit nach derselben Zielgruppe, den Lesern des IRP-Buches. Die Buchautoren gehen von einer weltweiten Leserschaft mit Interesse im Bereich von plattformunabhängigen Webarchitekturlösungen aus. Zur Leserschaft gehören entsprechend Integration-Architects, Service-Designer wie auch Webentwickler. Des Weiteren sollen sowohl Frontend-Entwickler (API Konsumenten) als auch Backend-to-Backend Integrationsspezialisten von dem erarbeiteten Wissen profitieren. Die sekundäre Zielgruppe besteht aus API Product Owners, Cloud Offering Providers/Operators und API Reviewers. [11]

4.2.2 Qualitätsmerkmale

Die nachfolgenden Qualitätsmerkmale wurden gemäss den kritischen Erfolgsfaktoren aus der Aufgabenstellung[2] in Zusammenarbeit mit dem Auftraggeber in der ersten Woche der Studienarbeit ausgearbeitet.

Visuelle Konsistenz	Die Visualisierungen haben einen klar ersichtlichen einheitlichen Stil. Die verwendeten Elemente der Visualisierungen folgen demselben Grundprinzip.
Abstraktionsgrad	Die Visualisierungen haben denselben Abstraktionsgrad, d. h. Beispiele aus anderen Themengebieten zur Verdeutlichung sind sinnvoll und auf einem vergleichbaren Level gewählt. In den Visualisierungen treten keine grossen Sprünge zwischen Abstraktionsstufen oder willkürliche Wechsel zwischen Themen auf.
Unterscheidbarkeit	Unterschiede zwischen den verschiedenen Patterns müssen klar erkenntlich sein. Ähnliche Darstellungen sind nur dann zugelassen, wenn die Unterscheidung effektiv gerechtfertigt ist.
Wiedererkennungswert	Die Visualisierungen haben einen potentiellen Wiedererkennungswert, damit Betrachter die Darstellungen in Zukunft implizit mit den IRP in Verbindung setzen und die Patterns somit besser memorieren können. Dies gilt für sämtliche Verwendungsbereiche.
Farbkonzept	Für das Design wird ein blauer Grundton gewünscht. Dieser dient einerseits dem Wiedererkennungswert, andererseits zur Abgrenzung von anderen Patternsprachen. Diese Anforderung wird vom Auftraggeber als optional bewertet. Des Weiteren müssen die Visualisierungen auch in Schwarz/Weiss funktionieren, ohne dass der Informationsgehalt abnimmt. Farbenblinde und Personen mit anderen Sehein-schränkungen sollen durch die Farbwahl nicht benachteiligt werden.
Multi-Channel Verwendung	Die Visualisierungen werden auf verschiedenen Kanälen (z. B. Buch, Beamer oder Whiteboard) eingesetzt und sollen entsprechend darauf ausgerichtet sein. Für die Verwendung an einem Whiteboard bedeutet dies, dass die einzelnen Icons so einfach wie möglich nachgezeichnet werden können. Für die Verwendung in digitalen Medien/ Projektionen müssen die Darstellungen in unterschiedlichen Grössen gut aussehen

[2] Siehe «Aufgabenstellung» zu Beginn der Abschlussdokumentation

	und einfach zu handhaben sein. Die Icons und Abläufe sollten nicht zu dicht gepackt sein, um die Lesbarkeit auch auf kleinen Flächen zu erhalten.
International verwendbar	Zielgruppe sind Leser der ganzen Welt, entsprechend sollen global verständliche Sujets verwendet werden (Negativbeispiel: der Schwingsport ist über die Schweizer Grenze hinaus aussageschwach).
Political Correctness	Die Illustrationen beleidigen in keiner Weise eine bestimmte Gruppe von Menschen.
Skalierbarkeit	Dieser Punkt fliesst in die Verwendung auf verschiedenen Kanälen mit ein. Die Darstellungen sollen gut skaliert werden können, ohne in der Optik oder Aussagekraft an Qualität zu verlieren. Die Verwendung von Vektorgrafiken bietet sich entsprechend an.
Master / Detail View	Ein Pattern verfügt über eine übergeordnete Darstellung, welche lediglich das Pattern repräsentiert und einer detaillierten Visualisierung der entsprechenden internen Abläufe. Die beiden Darstellungen können je nach Verwendungszweck unterschiedlich eingesetzt werden.
Zweidimensionale Darstellung	Dreidimensionale Darstellungen sind ungeeignet für eine Patternsprache, da sie schlecht nachzeichenbar und ohne zusätzlichen Nutzen sind.
Erweiterbarkeit	Eine Patternsprache ist nicht abschliessend, die Visualisierungen sollen eine einfache Erweiterung unterstützen. Neue Patterns lassen sich leicht integrieren.
Pattern-Kategorien	Aus der Visualisierung eines Patterns soll die Zuordnung zu der jeweiligen Pattern-Kategorie ersichtlich sein.
Lizenzen	Es sollen lediglich geeignete OpenSource-Lizenzen verwendet werden (wenn möglich Creative Commons, Apache 2 oder Eclipse), um den Einsatz der Sprache zu erleichtern. Die Verwendung von BSD und MIT wird je nach Umstand geduldet. Zu vermeiden sind die Lizenzmodelle GPL und LGPL, da diese oftmals ein No-Go für Firmen sind. Die Visualisierungen sollen allgemein möglichst Lizenz-unabhängig sein.

4.3 Einflüsse

Das Projektteam wird von verschiedenen Einflüssen geprägt, welche sich auf die Gestaltung der Visualisierungen auswirken. In diesem Abschnitt werden verschiedene Aspekte unterschiedlicher Vorbilder beschrieben, welche das Projektteam beeinflussen und an welchen es sich bewusst orientiert. Es ist zu betonen, dass sämtliche Visualisierungen von Grund auf eigenständig entworfen werden. Das Bewusstmachen der Einflüsse hat unter anderem den Zweck, die Gemeinsamkeiten hervorzuheben, um unbewusstes Kopieren oder Übernehmen von bestehenden Visualisierungen zu vermeiden. Es soll darauf aufmerksam gemacht werden, dass Einflüsse nicht zwingend gemieden werden sollen bzw. nicht zu vermeiden sind. Das Projektteam strebt einen korrekten Umgang mit bestehendem Bilder- und Gedankengut an. Wie die genannten Einflüsse das Endprodukt beeinflussen, wird in einem weiteren Schritt bei den Designprinzipien und Guidelines ersichtlich.

4.3.1 Enterprise Integration Patterns

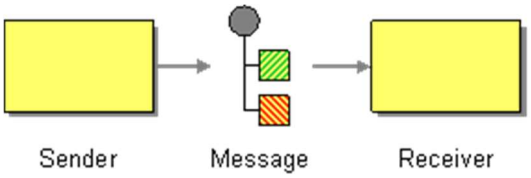


Abbildung 4.1 - EIP Message Visualisierung

Bekannt seit der Veröffentlichung des gleichnamigen Buches [4] im Jahr 2004, werden die Enterprise Integration Patterns [13] (nachfolgend EIP genannt) verwendet, um technologieunabhängige Designunterstützung für Integrationslösungen wie beispielsweise verteilte Softwaresysteme zu bieten. Kennzeichnend für diese Messaging-Patterns ist die Verwendung von simplen geometrischen Formen wie auch diverser Farben zur Unterscheidung von Elementen, wie auch in der Abbildung 4.1 zu sehen ist. Die EIP arbeitet stark mit Annotationen in Textform und setzt bei den Visualisierungen auf das Verständnis des Anwenders, was die Semantik angeht. So muss der Betrachter der Patterns mit der EIP-Darstellungen vertraut sein, um den abstrakten Formen wie Punkten und Rechtecke ohne Beschreibung ihre Bedeutung zuordnen zu können.

4.3.2 Cloud Computing Patterns

Abbildung 4.2 - CCP Elastic Platform Icon

Die weitverbreiteten Visualisierungen der Cloud Computing Patterns [14] (nachfolgend CCP genannt) üben einen starken Einfluss auf die Entwickler der Bildsprache für IRP aus, da die Patterns bereits in Projekten zum Einsatz kamen und somit bekannt sind. Ihren Wiedererkennungswert verdanken die Icons einerseits dem abgerundeten Rahmen, welcher an Icons wie man sie beispielsweise von iOS kennt erinnert, andererseits den dominanten Konturen und sehr vereinfachten Darstellungen. Ein weiteres Merkmal und gleichzeitig auch ein Vorteil der CCP ist ihre Gestaltung in Schwarz/Weiss, somit sind sie auch für Personen mit eingeschränkten Sehfähigkeiten gut zu nutzen. Wie in Abbildung 4.2 zu sehen ist, gibt die Verwendung von verschiedenen Graustufen den Icons einen eigenen Charakter.

Die CCP bestehen unter anderem aus Visualisierungselementen, welche in der Informatik bereits stark verankerte Bedeutungen haben wie beispielsweise den aufgestellten Zylinder, welcher sich über die letzten 30 Jahre zur Standartvisualisierung für diverse Speichermöglichkeiten entwickelt hat.

4.3.3 Cisco Icons

Abbildung 4.3 - ATM Router Icon

Die Cisco Icons für Netzwerktopologien [15] dürften vor allem Netzwerkspezialisten bekannt sein. Obwohl dieser Bereich weniger mit den IRP verwandt ist als die EIP oder CCP, möchten wir diese Art von Icon nichtdestotrotz in den Einflüssen aufführen, da sie das Modellieren und visuelle Dokumentieren von Implementationen in der Informatik entscheidend mitgeprägt haben.

Wie auch bei der Abbildung 4.3 zu sehen ist, bestehen die Visualisierungen meist aus dreidimensionalen Darstellungen. Den wahrscheinlich grössten Anteil zur Wiedererkennung leistet jedoch die Farbe. Das Blau wird implizit mit den Cisco-Icons assoziiert, unserer Meinung nach ein gutes Beispiel für den gezielten Einsatz von Farben.

4.3.4 Unified Modeling Language

UML ist im Bereich der Softwareentwicklung ein vertrauter Begriff. Die einfache Bildsprache mit Pfeilen und Rechtecken kann von jedem Softwareentwickler angewendet und verstanden werden. Die extrem starke Vereinheitlichung erleichtert die bereichsübergreifende Kollaboration. An UML angelehnte Modellierungen geniessen die Vorteile der weitverbreiteten Sprache.

4.3.5 Bauhaus

Als Einfluss der Bauhaus-Bewegung verstehen wir die starke Verknüpfung von Gestaltung und Zweck. Typisch für den im 20. Jahrhundert aufgekommenen Trend in den Feldern der Architektur, Design und Kunst sind die schlichten Gestaltungformen, welche auf Funktionalität ausgerichtet sind. [12] Die Vermischung von Kunst und Handwerk als neue Denkweise führte zu einem Paradigma-Wechsel, in welchem schnörkelloses Design detaillierten Verzierungen vorgezogen wurde.

4.3.6 Icon-Design

Technische Visualisierungen sind stark geprägt von Illustrationen, welche durch ihre weitverbreitete Verwendung allgemein akzeptiert sind und den Betrachtern bereits die angestrebte Absicht vermitteln. Jeder Smartphone- oder Internetbenutzer wurde bereits mit solchen Icons konfrontiert. So dürfte der eben genannten Gruppe beispielsweise klar sein, dass sich hinter der Abbildung einer Mülltonne eine Löschfunktion befindet. Zu den konkreten Einflüssen zählen Icons des Material Designs [16], des Bootstrap-Frameworks [17] oder diejenigen der Font Awesome Library [18], welche allesamt Ähnlichkeiten in der Gestaltung aufweisen und dasselbe Ziel verfolgen: Dem Benutzer mit beschränkten Bildinformationen möglichst viel zu vermitteln. Die Icons sind stark auf Usability ausgelegt und für die Skalierung optimiert, dies lässt nur wenig Spielraum für Details. Des Weiteren wurden die Icons alle in Schwarz/Weiss ohne Graustufen entworfen.

4.3.7 Webtrends

Da die IRP moderne Probleme der Webentwicklung adressieren, sollten die Visualisierungen entsprechend aktuelle Webtrends wiederspiegeln. Ein starker Einfluss hierbei ist die «mobile Welt», Smartphones und Tablets sind in der heutigen Zeit nicht mehr wegzudenken. Der Softwarevertrieb wird vermehrt durch die Verteilung von Apps kanalisiert. Um mehr Anwender zu erreichen, werden Lösungen für sämtliche Plattformen entwickelt. Ob bewusst oder unbewusst, prägt diese Entwicklung den weborientierten Anwender und übt Einfluss auf das Qualitätsbewusstsein aus.

In der Designbranche hat sich das Flat Design durchgesetzt, dies lässt sich unter anderem besonders gut an der Entwicklung verschiedener Betriebssysteme und ihrer Benutzeroberflächen beobachten. So kamen führende Hersteller wie Google, Microsoft oder Apple weg von dreidimensionalen, detailreichen Abbildungen und verwenden immer mehr flache, abstrahierte Darstellungen in ihren Softwarelösungen.

4.4 Visualisierungskomponenten

Bevor man sich die Frage stellt, wie die Visualisierungen zu gestalten sind, sollte man sich damit auseinandersetzen, was effektiv visualisiert werden soll.

Illustrationen der IRP lassen sich in Komponenten mit unterschiedlichen Zwecken hinsichtlich der Darstellung unterteilen. Dieser Abschnitt beschreibt die Absichten der verschiedenen Komponenten auf abstrakter Stufe. Konkret umgesetzte Komponenten sind im Kapitel 7«Endprodukt» dokumentiert.

4.4.1 Basis der Visualisierungen

In diesem Projekt werden die Pattern-Entwürfe der IRP-Autoren visualisiert. Grundlage für die Darstellungen bilden die von den Autoren zur Verfügung gestellten Textfragmente. Sämtliche Abbildungen dienen dazu, die Semantik der Textgrundlage zu wiederspiegeln.

4.4.2 Icons

Jedes Pattern wird von einem Icon repräsentiert. Das Icon dient zur besseren Wahrnehmung und kann in Kompositionen wie auch in Übersichten dargestellt werden. Der Inhalt des Icons lässt einerseits auf den Namen des Patterns schliessen, um die Patterns besser zu memorieren, andererseits lässt er bereits auf die Funktionsweise des Patterns schliessen. Entsprechend sollte sich die Essenz des Patterns in seinem Icon wiederspiegeln. Das Konzept der Icons wird bereits bei den Cloud Computing Patterns eingesetzt, wodurch diese Umsetzung implizit einen entsprechenden Einfluss auf die Gestaltung der Interface Representation Patterns hat.

Nicht nur die einzelnen Patterns, auch die übergreifenden Pattern-Kategorien werden durch ein Icon dargestellt.

4.4.3 Kommunikationskomponenten

Während sich mit den Icons statische Kompositionen wie beispielsweise ein Action Plan [9] eines APIs bereits gut visualisieren lässt, werden für die Darstellung der Kommunikation zweier Systeme weitere Komponenten benötigt. API Konsumenten, Endpunkte, *Request* und *Reply* können als solche Kommunikationskomponenten bezeichnet werden. Im Rahmen der Visualisierungsarbeit sollen solche Komponenten erarbeitet werden, um eine einheitliche Verwendung der Bildsprache zu fördern und die Anwender in der Gestaltung ihrer Kompositionen zu unterstützen.

4.4.4 Kompositionen

Ziel der Visualisierungen ist es, dass sie flexibel in unterschiedlichen Bereichen eingesetzt werden können. Beobachtungen auf anderen Gebieten zeigen auf, dass sich der Einsatz der Pattern-Icons je nach Anwender stark unterscheiden kann. Es kann gut vorkommen, dass die Illustrationen auf eine Art eingesetzt werden, welche von den Designern nicht so vorgesehen war. Beispielkompositionen bilden eine effektive Möglichkeit, dem Anwender eine Idee der Einsatzmöglichkeit entworfener Visualisierungen zu geben und tragen implizit dazu bei, ein allgemeines Grundverständnis der Bildsprache aufzubauen. Daher sollen solche Ressourcen den Anwendern in Form von konkreten Visualisierungsvorschlägen zur Verfügung gestellt werden.

4.4.5 Semantische Konzepte

Wird ein Gestaltungselement als Komponente in einem Icon eingesetzt, wird dieser Komponente eine Bedeutung zugesprochen. Somit besitzt jedes Element der Visualisierungen eine Semantik, welche Icon-übergreifend gilt. Diese Eigenschaft kann gezielt genutzt werden, in dem bestimmte semantische Konzepte festgelegt werden, welche zu Verständniszwecken eigenständig visualisiert werden. Als prominentes Beispiel ist der «nicht-spezifizierte Parameter» zu nennen, welcher in erster Linie ein semantisches Konzept repräsentiert und in verschiedenen Visualisierungen zum Einsatz kommt. Die Bedeutung solcher Komponenten muss zwangsläufig zusätzlich zu der Visualisierung dokumentiert werden. Die Verwendung solcher Visualisierungsmethoden erinnert stark an die EIP.

4.5 Design-Richtlinien

Design-Richtlinien beschreiben die wichtigsten Punkte in der Entwicklung von aussagekräftigen Visualisierungen. Durch Einhalten dieser Richtlinien ist eine einheitliche Gestaltung garantiert. Im Web sind viele öffentlich zugängliche Guidelines verfügbar, welche Designer beim Entwurf von Visualisierungen für eine bestimmte Plattform unterstützen. Inspiriert durch bekannte Beispiele [19] hat sich das Projektteam dazu entschieden, Richtlinien für den Entwurf von Visualisierungen der IRP festzulegen.

4.5.1 Übersicht

Die Design-Richtlinien der IRP Visualisierungen sind zu einem grossen Teil während dem Entwurfsprozess selbst entstanden. Da das Projektteam zu Beginn der Studienarbeit über wenig Erfahrung in diesem Bereich verfügte, mussten die nachfolgende Ergebnisse iterativ erarbeitet und verbessert werden. Die Richtlinien sind aufgrund des beschränkten Zeitrahmens des Projektes nicht abschliessend, beschreiben jedoch die wichtigsten Punkte im Zusammenhang mit der Darstellungen der IRP mit Stand Juni 2017. Die Richtlinien beinhalten sowohl grafische Aspekte wie auch technische Umsetzungsdetails. Mit Hilfe der folgenden Angaben soll es Dritten möglich sein, Visualisierungen zu entwerfen, welche sich optisch in die bestehenden IRP-Visualisierungen einreihen.

Die nachfolgende Grafik beschreibt wie das Projektteam die Design-Richtlinien versteht.

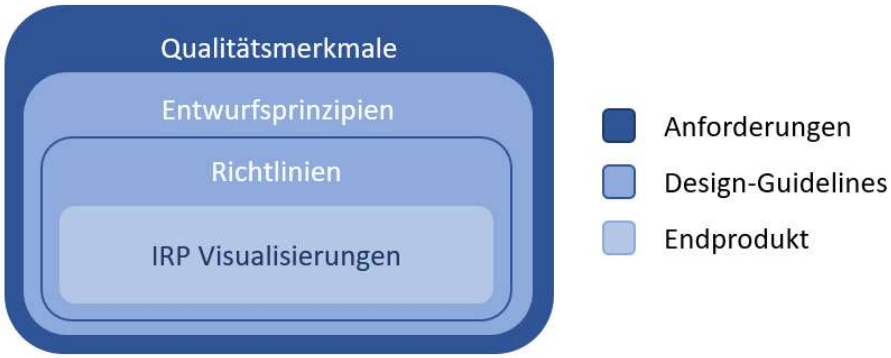


Abbildung 4.4 - Gestaltungrahmen der IRP Visualisierungen

Die Qualitätsmerkmale sind durch den Auftraggeber vorgegeben und bilden somit einen fixen Rahmen, in welchem sich die Entwürfe der IRP Visualisierungen bewegen können. Die Design-Guidelines wurden innerhalb dieser Vorgaben vom Projektteam definiert. Diese Guidelines bestehen aus Entwurfsprinzipien, welchen den Rahmen der Visualisierungsmöglichkeiten nochmals einschränken und detaillierteren Richtlinien, welche die Form der Visualisierungen nochmals konkretisieren. Somit wird aus einer Vielzahl von Darstellungsmöglichkeiten eine Teilmenge gewonnen, aus welcher das Endprodukt entsteht.

4.5.2 Entwurfsprinzipien

Dieser Abschnitt beschreibt generelle Prinzipien, an welche sich ein Designer beim Entwurf der Visualisierungen halten sollte. Zweck der Prinzipien ist es, eine einheitliche, grundlegende Einstellung zum Designprozess im Projektteam zu fördern, welche sich mit den nichtfunktionalen Anforderungen deckt.

Reduzierte Gestaltung Dieses Prinzip unterstützt implizit die nichtfunktionale Anforderung «Multi-Channel-Verwendung», da die Visualisierungen für Whiteboard-Sessions leicht nachzuzeichnen sein sollten. Die Visualisierungen sind in erster Linie zweckgebunden, dies sollte der Designer bei jedem Visualisierungsschritt im Gedächtnis behalten. Es soll bewusst auf Realismus und Detailvielfalt verzich-

tet werden und den Visualisierungen ein Symbolcharakter verliehen werden. Unnötige bzw. ausdrucksschwache Elemente sind wegzulassen. Die Einfachheit in der Gestaltung hat des Weiteren den Vorteil, dass die Visualisierungen auch in kleinerer Ausführung gut zu erkennen sind. Die Darstellungen sollen trotz allem dem Betrachter gefallen und einen ästhetischen Eindruck hinterlassen, daher ist jeweils genau abzuwägen, wie weit man sich auf die Abstrahierung einlässt. Dieses Entwurfsprinzip richtet sich stark nach dem Grundgedanken der Bauhaus-Bewegung.

Skalierung für Zielmedium

Die Visualisierungen finden hauptsächlich Anwendung auf Webseiten und Druckmedien. Entsprechend sollte bereits bei Entwurf der Visualisierung eine adäquate Grösse gewählt werden. Gehen wir beispielsweise davon aus, dass Icons in einem Buch in einer Minimalgrösse von 2x2 cm abgedruckt werden, so ist es sinnvoll, dass Icon in dieser Grösse zu entwerfen. So wird verhindert, dass Bildinhalte in kleinen Icons nicht mehr zu erkennen sind. Der umgekehrte Fall, dass ein Icon zu klein entworfen wird, kann dazu führen, dass es beim Hochskalieren auf einmal leer wirkt. Der Entwurf der Visualisierungen sollte die Skalierung im Rahmen der zu erwartenden Verwendungsgrössen unterstützten, so dass bei Ausführungen in unterschiedlichen Grössen sowohl Informationsgehalt der Darstellung wie auch die optische Wirkung unverändert bleiben.

Anwenderorientierung

Beim Entwurfsprozess ist die Sichtweise der Anwender[3] einzunehmen. Stehen Vorstellungen des Projektteams in Konflikt mit der Auffassung der Anwender, so überwiegt im Zweifelsfall die Meinung des Endbenutzers. Dieses Prinzip kann nur eingehalten werden, wenn regelmässig Feedback der Zielgruppen eingeholt wird und Designer und Anwender in ständigem Dialog miteinander stehen. Durch solch ein Vorgehen, wie man es aus dem User Centered Design in der Softwareentwicklung kennt, wird unter anderem das Anforderungskriterium der internationalen Verwendung geprüft, sofern repräsentative Testpersonen gewählt wurden.

Kanten durch Konturlinien

Eine grundlegende Frage, welche man sich beim Entwerfen von Grafiken stellen sollte ist, ob man mit Konturlinien arbeitet oder Kanten durch unterschiedliche Farben bzw. Kontrasten der betroffenen Fläche hervorhebt. Dieser Entscheid hat einen starken Einfluss auf das Erscheinungsbild der Visualisierung. Die Cloud

[3] Siehe Absatz 4.2.1 “Benutzercharakteristik”

Computing Patterns setzen klar auf Konturen während beispielsweis das Material Design stark durch die Verwendung von Flächen auffällt. In diesen Entscheid fliesst die Farbwahl wie auch der Verwendungszweck der Grafiken mit ein. Da die Visualisierungen der IRP grundsätzlich unabhängig von Farben funktionieren und sie einfach von Hand nachskizziert werden sollten, liegt der Entschluss nahe, mit Konturlinien zu arbeiten.

Ganzheitliches Design

Die Kombination der verschiedenen Visualisierungen für diverse Kompositionen sollte beim Entwurf eines einzelnen Elementes berücksichtigt werden. Icons sollen sich in den Kontext der IRP Visualisierungen eingliedern, ohne dabei aufzufallen. Entsprechend sollten bestehende Visualisierungen genutzt werden, um die Anforderung des einheitlichen Designs iterativ zu überprüfen. So wird eine ungewollte Diversität in der Darstellung der einzelnen Elemente am Ende des Projektes verhindert.

Limitierte Perspektive

Die Anforderung der Zwei-Dimensionalität lässt diesbezüglich wenig Spielraum offen. Gewünscht sind Darstellungen, welche nicht durch unnötige viele Perspektive oder durch räumlicher Gestaltung unübersichtlich wirken. Der begrenzte Einsatz von Perspektiven ist jedoch nicht generell untersagt. So kann beispielsweise eine Box nur mit Hilfe von Perspektive als solche erkennbar gemacht werden, da ansonsten nur ein Rechteck zu sehen wäre. Perspektive soll einheitlich eingesetzt werden, indem die Anzahl unterschiedlicher Blickwinkel minimal gehalten werden und sich neue Entwürfe diesbezüglich an bestehende Visualisierungen orientieren. Wo möglich, sind Visualisierungen zu entwerfen, in welchen die Räumlichkeit vernachlässigt wird.

Geometrische Formen

Beim Entwerfen der Visualisierungen auf dem Computer sollen einfache geometrische Formen wie Kreise oder Rechtecke eingesetzt und wo möglich auf das Zeichenstift-Werkzeug verzichtet werden. Die Verwendung der geometrischen Formen führt implizit zu einem symbolischen Charakter der Visualisierung und ist ein nützliches Hilfsmittel, wenn es darum geht, genau zu arbeiten.

4.5.3 Gestaltungselemente

Die anschliessend aufgeführten Aspekte definieren, nach welchen Vorgaben die Visualisierungen im Zeichnungstool umgesetzt wurden.

Farbwahl

Sämtliche Visualisierungen sind in Schwarz/Weiss gehalten. So erfüllen die Icons zwangsläufig das Kriterium, dass sie in Schwarz/Weiss funktionieren. Das Projektteam hat sich bewusst dazu entschieden, das Farbkonzept aussen vor zu lassen, um sich in einem ersten Schritt nur mit dem Inhalt der Visualisierung zu befassen. Das Einfärben der Icons wird mit diesem Entscheid nicht ausgeschlossen, jedoch aus Gründen der einheitlichen Gestaltung auf einen späteren Zeitpunkt verlegt.

Graustufen sind nur dann einzusetzen, wenn sie eine semantische Bedeutung haben. Der Einsatz von Grau für gestalterische Zwecke wurde wie die Farbgebung selbst noch nicht beschlossen.

Konturlinien

Weisse und graue Flächen verfügen immer über eine schwarze Konturlinie. Schwarze Flächen haben keine Konturlinie, da diese bei der Skalierung lediglich irritiert.

Standartstrichstärke für die Pfade ist 1pt, verwendet man dickere Konturen ist darauf zu achten, dass die Stärke ein Vielfaches von 0,5pt (1,5pt; 2pt; 2,5pt etc.) beträgt. Bei dünneren Konturen ist eine Strichstärke von 0,75pt, 5pt oder 0,25pt zu verwenden.

Abgerundete Ecken

Ecken und Rundungen definieren den Stil von Grafiken. Im Rahmen der IRP werden Rundungen spitzen Kanten vorgezogen. In gewissen Fällen, beispielsweise beim Darstellen eines Blatt Papiers, macht es durchaus Sinn, spitze Ecken einzusetzen. Wenn es jedoch möglich ist, Ecken abzurunden ohne dass ein unnatürlicher Eindruck entsteht, so kann Gebrauch von diesem Stilmittel gemacht werden, um den Visualisierungen einen ruhigeren Charakter zu verleihen.

Text

Text sollte in den Visualisierungen möglichst vermieden werden, da die Bilder für sich sprechen sollen. In gewissen Fällen macht es jedoch Sinn, eine Darstellung mit einem Begriff zu verdeutlichen. Für den Export werden Schriften in Pfade umgewandelt.

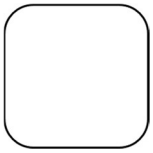
4.5.4 Komponenten

In diesem Abschnitt werden die festgelegten Standardkomponenten beschrieben, welche immer wieder verwendet werden. Einerseits wird kurz erläutert, wozu die Komponente verwendet wird, andererseits soll aufgrund der Detailangaben sichergestellt werden, dass die Komponenten in unterschiedlichen Visualisierungen ein einheitliches Erscheinungsbild aufweisen.

4.5.4.1 Icon

Für jedes Pattern wird ein Icon erstellt, welches in Übersichten bzw. Kompositionen verwendet werden kann. Der Inhalt des Icons soll klar ersichtlich machen, um welches Pattern es sich handelt. Für eine einheitliche Darstellung werden die Icons von solch einem abgerundeten Rahmen umgeben. Die Icons erinnern stark an die App-Icons in iOS oder diejenigen der CCP, diese Anlehnung fördert die Akzeptanz der Icons bei den Anwendern. Grund für diese Gestaltung ist jedoch in erster Linie die Stilrichtlinie der abgerundeten Kanten und das Ziel, Patterns in einer Komposition klar durch eine Abgrenzung hervorzuheben.

Das abgerundete Rechteck hat bei den Visualisierungen folgende Dimensionen:



Breite: 2,5 cm
Höhe: 2,5cm
Eckenradius: 0,5 cm
Kontur: 1pt / Schwarz

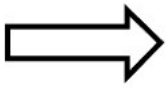
4.5.4.2 Nicht-Spezifizierter Parameter

Der nicht-spezifizierte Parameter wird durch einen einfachen Punkt repräsentiert. Diese Darstellung verrät nichts über die innere Struktur. Entsprechend kann ein nicht-spezifizierter Parameter wiederum weitere Parameter enthalten (z. B. Tree oder Liste) oder einen einfachen Parameter darstellen. Diese Darstellung wird verwendet, wenn die Art des Parameters für das darzustellende Pattern keine bedeutungsvolle Rolle spielt.

Standartmässig sind für den Parameter folgende Dimensionen gesetzt:

Breite: 0,5 cm
Höhe: 0,5cm
Füllung: Schwarz
Kontur: keine

4.5.4.3 Response und Reply



Ein weisser Pfeil mit schwarzer Konturlinie stellt den Kommunikationsfluss in einer Visualisierung dar. Er steht für einen *Request* oder eine *Reply*. Entsprechend kann dieser Pfeil auch in Icons eingesetzt werden, in welchen *Requests* bzw. *Replies* dargestellt werden sollen.

4.6 Technische Aspekte

Anschliessend werden technische Eigenschaften genannt, auf welche beim Entwurf zu achten sind. Es geht dabei darum, die Designer auf Details und ihre Folgen für die Visualisierungen aufmerksam zu machen und dadurch eine konsistente Darstellung zu fördern.

4.6.1 Format

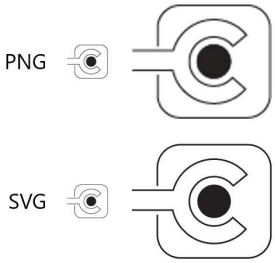


Abbildung 4.5 - Unterschied PNG - SVG

Um ohne Qualitätsverlust zu Skalieren, sollen vektorbasierte Grafiken (SVG) erstellt werden. Diese Grafiken lassen sich einfacher als pixelbasierte Darstellungen skalieren und mithilfe von einem vektorbasierten Zeichnungstool bearbeiten, da sämtliche Pfadinformationen enthalten sind. Microsoft PowerPoint unterstützt unter anderem die Verwendung von SVGs und bietet sogar rudimentäre Funktionen wie das Einfärben der Pfade. Um die verschiedenen Ebenen bzw. Gruppierungen der Grafik zu verändern, ist ein geeignetes Bearbeitungsprogramm[4] nötig. Um den Anwendern grosse Flexibilität im Einsatz der Visualisierungen zu gewähren, sollten diese jeweils zusätzlich als PNG in den verschiedenen Einsatzgrössen exportiert werden.

4.6.2 Skalierung

Bei der Umsetzung der Visualisierungen im Zeichnungstool soll darauf geachtet werden, ist es sinnvoll, die automatische Skalierung der Ecken und Konturen auszuschalten. Somit wird sichergestellt, dass die gesetzte Strichstärke bei Transformationen der Grafiken innerhalb des Tools erhalten bleibt und keine Bruchwerte als Grössenangaben erscheinen. Des Weiteren wird die Konsistenz implizit gefördert. Auf das exportierte Endprodukt hat diese Einstellung keine Auswirkung, sämtliche Konturen werden proportional skaliert.

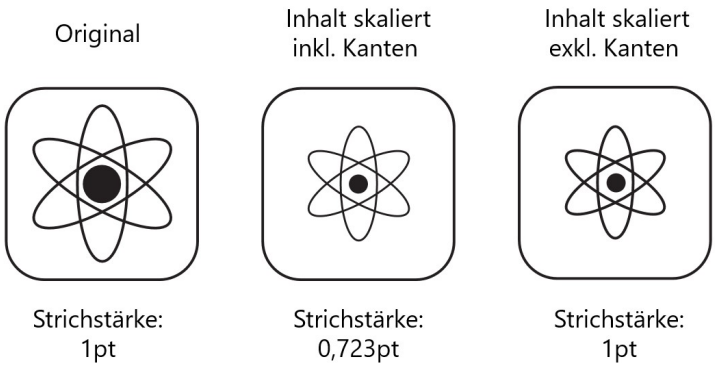


Abbildung 4.6 - Skalierungsdetails

Wie in Abbildung 4.6 zu sehen ist, wurde der Inhalt des ersten Icons verkleineret. Beim zweiten Icon wurde dabei die Konfiguration gesetzt, dass Ecken und Konturen mitskaliert werden. Dies führt dazu, dass beim zweiten Bild die Strichstärke automatisch angepasst wird. Wenn man dies nicht berücksichtig, was bei ständigem verändern der Grösse durchaus vorkommen

[4] Die Visualisierungen wurden mit Adobe Illustrator erstellt, eine kostenfreie Alternative wäre Inkscape.

kann, wächst mit der Anzahl Icons auch die Inkonsistenz des Gesamtlayouts. Ist das automatische Skalieren ausgeschaltet, bleibt die Dicke der Konturen dieselbe, wie im dritten Icon zu sehen ist. Sind die Linien in dieser Grafik in Abbildung 4.6 dem Designer zu stark, kann die Strichstärke immer noch auf 0,75pt angepasst werden. Dies mag ein Detail sein, welches vom blossem Auge kaum erkennbar scheint, es geht jedoch vielmehr darum, den Designer auf die Detailarbeit zu sensibilisieren.

4.6.3 Konturen in Flächen umwandeln

Vor dem Export sollte darauf geachtet werden, dass Pfade welche als Konturlinien eingesetzt werden, in Flächen umgewandelt werden. Denn beim Ändern der Füllfarbe in PowerPoint wird alles eingefärbt, was aus einer Pfadfläche besteht. Das Ändern der Farbe einer Konturlinie wird anders gehandhabt als bei einer Fläche. Durch umwandeln der Konturen in Flächen wird die Verwendung in PowerPoint unterstützt.

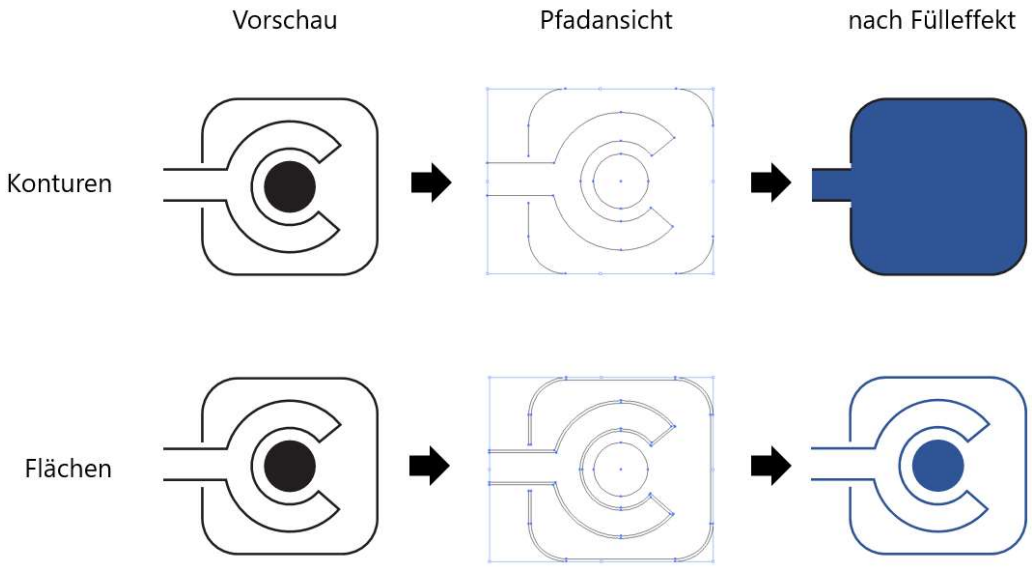


Abbildung 4.7 - Konturen in Flächen umwandeln

Kapitel 5

Umsetzung

5.1 Vorgehen

Das Vorgehensmodell für die Icon-Visualisierungen war zu Beginn der Studienarbeit noch nicht definiert. Das Projektteam hat sich schon früh mit Visualisierungsarbeiten befasst, woraus sich schnell ein Prozessmodell für das Vorgehen ableiten liess. Dieser Prozess wurde während der 15-wöchigen Studienarbeit mehrmals durchlaufen. Angefangen hat der Prozess jeweils in einer der wöchentlichen Sitzungen mit dem Auftraggeber, der zugleich einer der IRP Autoren ist. Dieser wählte neue Patterns aus dem «Pattern-Backlog» für die Weiterbearbeitung (Recherche und Visualisierung) aus, worauf das Projektteam seine Arbeiten gemäss dem in Abbildung 5.1 aufgezeigten Prozessmodell durchführte. Das Modell zeigt die verschiedenen Phasen von der Ideenfindung einer Visualisierung bis hin zur Finalisierung.

5.2 Prozessmodell

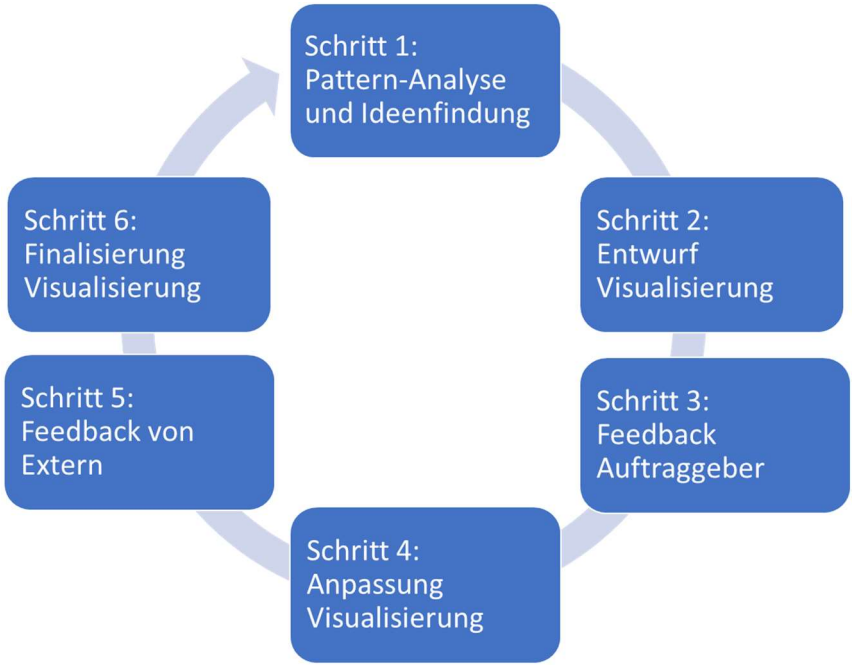


Abbildung 5.1: Prozessmodell zur Icon-Visualisierung

5.2.1 Schritt 1: Pattern-Analyse und Ideenfindung

Für eine korrekt umgesetzte und verständliche Visualisierung können folgende Fragestellungen hilfreich sein:

- Besteht eine detaillierte Beschreibung des Patterns?
- Sind «Known-Uses» bereits bekannt?
- Hat das Projektteam die Funktionsweise des Patterns verstanden?
- Was wird mit dem Pattern gelöst?
- Ist das Pattern mit einem anderen IRP verwandt?

Das Ziel dieser Phase ist, durch Beantwortung der obigen Fragen und unter Berücksichtigung der Erläuterungen im Visualisierungskonzept, erste Ideen zu den Visualisierungen in Form von Handskizzen zu entwickeln. Es empfiehlt sich zudem, erste Visualisierungsideen in einer Whiteboard-Session aufzuzeichnen. Diese effektive Art des Brainstormings prüft Ideen bereits im Ursprung auf ihre Ausdruckskraft und Verwendung am Whiteboard.

5.2.2 Schritt 2: Entwurf Visualisierung

Die technische Umsetzung der Ideen aus Schritt 1 erfolgt in dieser Phase. Aufgrund der Anforderung, die Icons in unterschiedlichen Medienarten einsetzen zu können, werden die Visualisierungen in einem vektor-basierten Zeichnungstool erstellt.

5.2.3 Schritt 3: Feedback Auftraggeber

In diesem Schritt äussert der Auftraggeber ein erstes Mal seine Meinung zu den Visualisierungen. Das Feedback kann in schriftlicher oder mündlichen Form erfolgen. Für den Auftraggeber sind unter anderem folgende Aspekte mit Bezug auf den Visualisierungsentwurf von Bedeutung:

- Entsteht eine Assoziation zwischen der Visualisierung und dem Pattern?
- Ist die Absicht der Visualisierung verständlich?
- Wie hoch ist die Aussagekraft der Visualisierung?
- Wurde die Konsistenz berücksichtigt?
- Wurden die nicht-funktionalen Anforderungen eingehalten?

Der Auftraggeber kann andere IRP-Autoren für ein erstes Review beiziehen.

5.2.3.1 Vorgehen

Die Visualisierungsentwürfe werden in einer Vorlage erstellt. Jede Pattern-Kategorie hat ihre eigene Vorlage, die sich aus den Spalten «Patternname», «Final», «Favourites» und «Early Drafts» und den Zeilen der entsprechenden Patterns in tabellarischer Form zusammensetzt. Abbildung 5.2 zeigt einen Auszug aus der *Basic Representation* Vorlage für das *Atomic Parameter* Pattern.

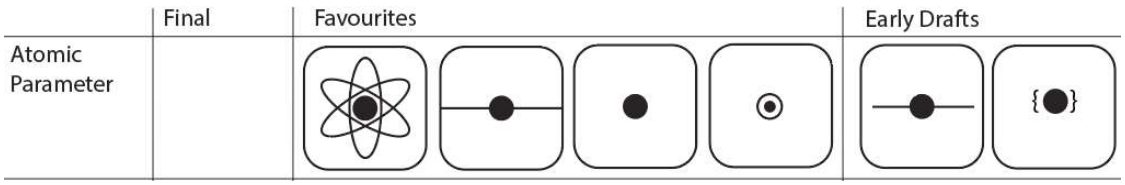


Abbildung 5.2: Auszug aus der Visualisierungs-Vorlage (nicht final)

Alle Visualisierungsentwürfe sind anfangs in der Spalte «Early Drafts» aufgelistet. Je nach Art der Rückmeldung des Auftraggebers wird ein Visualisierungsentwurf (i) in die Spalte «Favourites» verschoben, (ii) in der Spalte «Early Drafts» belassen, (iii) in die «Scribbles-Vorlage» verschoben, (iv) gänzlich verabschiedet.

Sticht eine Visualisierung in den Augen des Auftraggebers hervor und wird diese als passend empfunden, kommt diese in die Spalte «Favourites». Tritt dieser Fall ein, bestehen keine Änderungswünsche.

Es gibt zwei Gründe, weshalb eine Visualisierung in der Spalte «Early Drafts» belassen wird:

- Änderungs- bzw. Anpassungswünsche seitens Auftraggeber
- Geringe Überzeugungskraft

Besteht der Wunsch für eine weitere Anpassung, wird dies im Schritt 4 des Modells vorgenommen, und die neue Grafik startet den Prozess von Neuem in den «Early Drafts».

Ist eine Grafik als passend aber mit geringer Überzeugungskraft eingestuft worden, so verbleibt diese in der bisherigen Spalte. Diese Entwürfe haben nach wie vor die Chance, in die «Favourites» Spalte zu kommen, sofern eine oder mehrere Testpersonen die Grafiken in einer späteren Phase anders einstufen.

Assoziiert der Auftraggeber den Visualisierungsentwurf mit einem anderen Pattern oder stuft er den Entwurf als nicht passend ein, dann wird dieser in die «Scribbles-Vorlage» verschoben. Die «Scribbles-Vorlage» beinhaltet verschiedene Arten von Entwürfen, von einzelnen Komponenten wie beispielsweise einem Stift bis zu fertiggestellten Icons, welche für später dazukommende Patterns von Nutzen sein können. Die Visualisierungen in der Scribbles-Vorlage werden nicht weiterbearbeitet und demzufolge fallen die weiteren Schritte weg.

Die letzte Option ist, den Visualisierungsentwurf zu verabschieden. Tritt dieser Fall ein, dann ist der Entwurf den obigen Fragestellungen nicht gerecht geworden.

5.2.4 Schritt 4: Anpassung Visualisierung

Dieser Schritt setzt sich mit Änderungs- beziehungsweise mit Anpassungswünschen auseinander. Um Änderungen nachverfolgen zu können, wird vor jeder Änderung sichergestellt, dass eine PDF-Kopie der Vorlage besteht.

Die Vorgaben aus dem Visualisierungskonzept werden auch bei der Nachbearbeitung berücksichtigt. Eine Ausnahme bildet eine explizite Anweisung seitens des Auftraggebers.

Die in Abbildung 5.3 aufgeführte Grafik zeigt auf, wie ein Änderungswunsch aussehen kann. In diesem Beispiel besteht der Wunsch, dass der Pfeil, welcher auf das Tachometer zeigt, besser zur Geltung kommen soll.

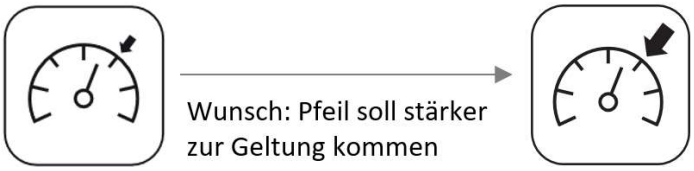


Abbildung 5.3: Änderungswunsch Rate Limit Pattern

5.2.5 Schritt 5: Feedback von Externen

Eine Visualisierung kann je nach Betrachter verschieden aufgefasst werden. Dieser Schritt umfasst das Einbeziehen mehrerer externer Testpersonen und deren Feedback zu den erstellten Grafiken. Externe Personen haben noch kein Vorwissen zu den erarbeiteten Visualisierungen, können aber mit den IRP vertraut sein.

Bei der Wahl der Testpersonen sollten folgende Überlegungen gemacht werden:

- Kennt die Testperson die IRP?
- Ist die Testperson mit den hier vorliegenden Technologien vertraut?
- Wo findet sich eine Testperson?

Der Auftraggeber holte das Feedback der Testpersonen selbst ein. Weitere Informationen zur Testmethode und Art der Befragung kann dem Kapitel 7 «Testverfahren und Ergebnisse» entnommen werden.

Vorgehen

Der Auftraggeber erfasste die Befragungsergebnisse direkt in den Visualisierungsvorlagen in Form von Notizen sowie auch zusammengefasst in einer E-Mail. Aus diesen Befragungen und Diskussionen wurden pro Pattern jeweils die finale und alternative Version der Visualisierungen ausgewählt.

Zu diesem Zeitpunkt wird die Spalte «Final» in der Vorlage mit einem Icon besetzt. Für die Alternativversion gab es bis dato noch keine eigene Spalte. Das Projektteam passte die Vorlage für die Schlussabgabe entsprechend an. Abbildung 5.4 zeigt die ergänzte Vorlage mit der zusätzlichen Spalte *Runner-Up*.

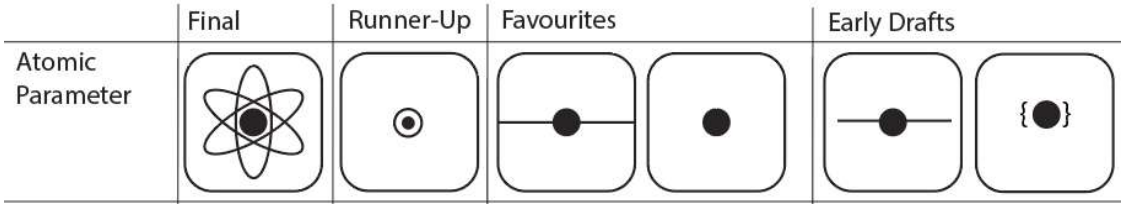


Abbildung 5.4: Auszug aus der finalen Visualisierungs-Vorlage

Eine Testperson kann auch eine neue Idee für eine Pattern-Visualisierung vorschlagen. Ist diese Lösung für den Auftraggeber eine Option für die finale oder alternative Variante, so wird die Grafik nachträglich gezeichnet.

5.2.6 Schritt 6: Finalisierung Visualisierung

Am Anfang dieser Phase stehen die meisten finalen und alternativen Versionen der Visualisierungen fest. Aufgrund der Testergebnisse gibt es verschiedene Möglichkeiten, wie man weiterfahren kann. Enthält eine finale oder alternative Version keine weiteren Bemerkungen, muss das Icon nicht angepasst werden. Sind Kommentare zu Anpassungen gegenüber den finalen oder alternativen Versionen vorhanden, werden diese wunschgemäss und nach Rücksprache mit dem Auftraggeber umgesetzt.

Die finalen und alternativen Grafiken werden in dieser Phase auf deren korrekte Umsetzung gemäss dem Visualisierungskonzept geprüft und für den Export aufbereitet.

5.3 Rückblick

Die Studienarbeit umfasste insgesamt 15 einwöchige Iterationen. Das Projektteam erhielt vom Auftraggeber wöchentlich Rückmeldungen zu den erarbeiteten Visualisierungen. Dies ist ein wichtiger Aspekt, denn ohne Feedback und regelmässigen Austausch wäre es nicht möglich gewesen, die Visualisierungen auf den jetzigen Stand zu bringen.

Kapitel 6

Endprodukt

6.1 Icons

Der nachfolgende Abschnitt zeigt die erstellten Icons und erläutert die Überlegungen sowie die Entwicklung während des Entwurfsprozesses. Die Alternativgrafiken (Runner-Up) werden nicht näher erläutert, können aber bei Interesse im Anhang 15 eingesehen werden.

API Consumer

Der API-Konsumer ist meist eine technische Komponente, welche die Funktionen des API in Anspruch nimmt. Daher wurde ein Roboter als erster Vorschlag visualisiert, der die Idee eines maschinellen Clients repräsentieren soll. Nach diversen Überarbeitungen blieb nur noch der Arm des Roboters übrig, welcher nach einem Parameter greift. Die Visualisierung wurde absichtlich in Anlehnung an die Notation des UML-Interface entworfen, da sich der API-Konsument einer Schnittstelle bedient.

API Endpoint

Äquivalent zum Konsument wurde der API Endpunkt entworfen, nur das diesmal der Punkt durch die schwarze Füllung betont wird. Das Icon hebt das Gegenstück, auf welches zugegriffen wird, hervor. Der Punkt wird mit einem nicht-spezifizierten Parameter in Verbindung gebracht.

Request

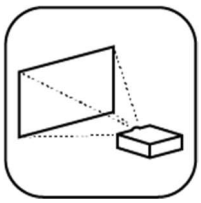
Das ein Icon für einen Call nötig ist, wurde dem Projektteam erst während dem ersten Verwendungstest bewusst. Das Icon wurde mit der Absicht entworfen, um in einem Action Plan [9] die *Response* zu kennzeichnen. Die Grafik soll eine Nachfrage darstellen, welche üblicherweise vom Konsument

ausgeht. Das Verlangen des Konsumenten wird durch das Fragenzeichen ausgedrückt.

Reply

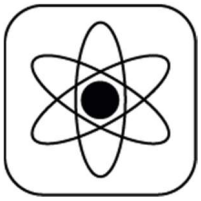
Die Entstehung sowie die Idee hinter dem Icon dürfte nach der Erläuterung zum Request klar sein.

Basic Representation



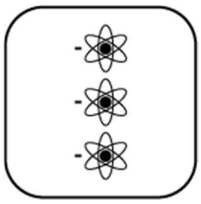
Unterschiedliche Menschen assoziieren mit dem Begriff «Representation» je nach ihrer Prägung unterschiedliche Konzepte (z. B einen politischen Repräsentanten, eine Vertretung oder eine grafische Darstellung). Da es in den Patterns dieser Kategorie um die Darstellung der Datenstruktur geht, wurde das Bild der Projektion gewählt. Wie bei der Projektion ist für den Konsumenten das für ihn Sichtbare von Bedeutung, wobei die Struktur API-intern auch anders aussehen kann.

Atomic Parameter



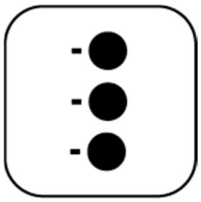
Das Icon macht Gebrauch vom nicht-spezifizierten Parameter. Der Parameter als Kern eines Atoms soll dem Betrachter klarmachen, dass es um eine kleinste, unteilbare Einheit geht. Obwohl längst klar ist, dass sich Atomkerne teilen lassen, bliebt der Begriff Atomos (griechisch «unteilbar») sowie die Semantik, dass damit ein elementares Bauteil bezeichnet wird, in den meisten Mindsets der Menschen verankert. Das Icon macht sich diese Assoziation zu nutzen, um aufzuzeigen, dass es beim Atomic Parameter um das kleinste Bauteil einer *Response*/*Reply* geht.

Atomic Parameter List



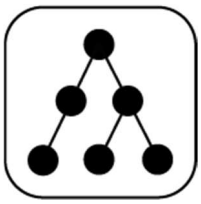
Die Visualisierung stellt eine Aufzählung dar, wie man sie vom Erstellen einer Einkaufsliste kennt. Die Liste beinhaltet Atomic Parameters und visualisiert somit wortwörtlich den Pattern-Namen.

Parameter List



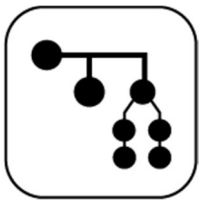
Bei der Parameter List ist nicht vorgegeben, was für Parameter sich in der Liste befinden, daher befindet sich bei jedem Aufzählungszeichen ein nicht-spezifizierter Parameter.

Parameter Tree



Bei diesem Pattern ist die Struktur der Parameter ausschlaggebend. Es gibt lediglich einen Einstiegspunkt. Aufgezeichnet stellt die Struktur einen Baum dar, wie man ihn aus diversen Bereichen der Informatik kennt.

Parameter Comb



Diese Visualisierung wurde oft diskutiert und hinterlässt auch jetzt noch Gesprächsbedarf. Das Bild soll einerseits die Struktur der jeweiligen Daten aufzeigen und gleichzeitig einen Kamm darstellen. Allerdings müssen die «Zähne des Kamms» nicht zwangsläufig gleich aussehen, was es schwierig macht, diese Metapher verständlich zu visualisieren. Der Comb hat grundsätzlich mehrere Einstiegspunkte. Der nicht-spezifizierter Parameter oben links ist daher nicht als Knotenpunkt zu verstehen, sondern als Gesamtobjekt, dessen innere Struktur aufgezeigt wird.

Pagination

Das Kategorie-Icon zeigt drei Papierseiten und bildet die Grundlage für alle Visualisierungen der Kategorie. Die Pattern-Visualisierungen nehmen jeweils diese Darstellung auf.

Cursor Based Pagination

In diesem Pattern zeigt ein Cursor auf die Einträge des Datensatzes. Streng genommen muss dieser nicht in Pages unterteilt sein, zwecks einheitlicher Darstellung und zur besseren Einordnung in die Kategorie, wurde nicht auf die Pages verzichtet.

Page Based Pagination

Bei diesem Icon sind die Pages wichtig für die Semantik. Der Cursor wird wiederverwendet, zeigt jedoch nun auf die Pages, da diese vollständig übertragen werden.

Time Based Pagination

Die Darstellung bezieht sich mehr auf die Bezeichnung als auf die Funktion des Patterns. Mit der Uhr wird der Begriff «Time» visualisiert, wie das Pattern funktionieren könnte wird aus der Visualisierung nicht ersichtlich.

QoS

Das Kategorie-Icon zeigt ein Gütesiegel um die Qualität zu betonen. Die ersten Entwürfe beinhalteten noch einige geschwungene Linien, welche aufgrund der Guidelines durch gerade Kanten ersetzt wurde.

API Key

Bei diesem Icon wurde anfänglich noch mit Text gearbeitet, um die Bedeutung des Schlüssels zu konkretisieren. Darauf wurde jedoch aus ästhetischen Gründen verzichtet. Der Schlüssel ist dadurch für dieses Pattern reserviert. Wie auch der API Key ist die abgebildete Art von Schlüssel nicht die sicherste, dient jedoch ihrem Zweck: Berechtigter/authentifizierter Zugriff auf Ressourcen.

Rate Limit

Für dieses Pattern gab es am meisten Visualisierungsentwürfe, welche allesamt eine Metapher für eine Fluss-Kontrolle zeigten (Barriere, Drehkreuz, Stop-Go usw.). Der Final Kandidat wurde gewählt, da das Bild vielen Personen bekannt sein dürfte und dadurch die Semantik auf einfache Weise darstellen kann. Die Darstellung wird bei Autos verwendet, um den Tempomat zu aktivieren bzw. anzuzeigen. Das Icon soll ein beschränktes Höchstlimit, welche nicht überschritten wird, aufzeigen.

Error Reporting

Ein Request oder Reply wird in der IRP auch mit einem Pfeil dargestellt um die Kommunikationsrichtung zu kennzeichnen. Der gebrochene Pfeil zeigt einen durch Fehler gestörten Kommunikationsversuch. Zuerst war lediglich der Fehler visualisiert, da es jedoch um das Reporting geht, wurde die Grafik in einer Sprechblase eingefügt. Die Sprechblase wird auf dieselbe Art wie bei Request und Reply verwendet, da der Fehler in der Regel als Reply zurückgegeben wird.

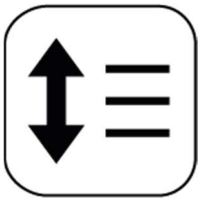
SLA-SLO

Das Icon zu SLA-SLO soll sowohl das Objective wie auch das Agreement zum Ausdruck bringen. Daher wurden die Gütesiegel als Bulletpoint zur Beschreibung eines Objectives und die Unterschrift für das Agreement eingesetzt.

Foundation Kategorie

Der Vorschlag des griechischen Tempels gefiel den Testpersonen, allerdings kam der Begriff «Foundation» zu wenig zum Ausdruck, da sämtliche Linien ausgezogen und die Flächen schwarz eingefärbt waren. Es kam die Idee auf, die Grundlage schwarz und den Rest der Konstruktion weiss einzufärben, allerdings war das Resultat nicht befriedigend. In einem weiteren Schritt wurde der ganze Tempel weiss dargestellt und nur die Kontur der Grundlage ausgezogen. Die gestrichelten Linien weisen darauf hin, was zum Bau gehören würde, jedoch noch nicht ist.

Vertical Integration



Das wichtigste bei diesem Pattern ist einerseits die Richtung, andererseits die Tatsache, dass verschiedene Systeme, Komponenten bzw. Schichten in dieser Richtung zusammenarbeiten. Daher die Pfeile und die drei Schichten in der Visualisierung.

Horizontal Integration

Dieses Icon ist das horizontale Äquivalent zur Vertical Integration.

Public API

Auf den Begriff «API» kann aus der Visualisierung nicht geschlossen werden. Nach ersten Versuchen war klar, dass es wichtiger ist, den öffentlichen Zugang statt die Schnittstelle selbst zu Visualisieren. Die Grafik stellt demnach eine globale Kollaboration verschiedener Personen bzw. Entwickler zusammen.

Community API

Das Icon der Community API verbindet Visualisierungen des Public API und des Solution-Internal API.

Solution Internal API

Im Gegensatz zum Globus stellt das Dach einen privaten Bereich dar, wodurch sich der Unterschied zwischen dem öffentlich zugänglichen Public API und dem Solution-Internal API visualisiert wird.

Service Contract

Dieses Icon wurde als letztes Entworfen. Es fällt klar beim Kriterium der Nachzeichenbarkeit durch. Ob am Computer oder von Hand ist es eine grosse Herausforderung. Da der Vertrag in Papierform bereits durch SLA-SLA vorbelastet war, entschied man sich jedoch trotzdem für das Motiv des Handschlags. Ob das erwähnte Kriterium zum Verwerfen des Icons führt, wird sich in den ersten Praxistests zeigen.

Wish List

Diese Visualisierung stellt das «Skeleton» einer Response dar, aus welchem ausgewählt wird, welche Parameter gewünscht werden. Daher die Checkboxes, in welchen die Häkchen gesetzt sind. Aus Konsistenzgründen ist dieses Icon an die Atomic Parameter List und die einfache Parameter List angelehnt.

Wish Object

Das Wish Object unterscheidet sich von der Wish List lediglich durch die verschachtelte Struktur. Dementsprechend wurde der Visualisierung der Wish List eine weitere Substruktur gegeben, um diesen Unterschied klar zu machen.

Request Bundle

Wie bereits erwähnt, steht der weisse Pfeil unter anderem für einen *Request/Response*. Für die Visualisierung wären gebündelte Sprechblasen mit einem Fragezeichen streng genommen besser angebracht. Solch eine Darstellung ist aufgrund der Sprechblasenform jedoch schwieriger zu erkennen und von Hand zu zeichnen. Daher ist die Entscheidung auf diese Visualisierung gefallen.

Metadata Parameter

Metadata Parameter sind Parameter, daher der schwarze Kreis. Innerhalb des Parameters befinden sich Informationen über die gesendeten Daten. Aus diesem Grund wurde das *i*, welches man oftmals bei Auskünften vorfindet, eingesetzt.

Aggregated Metadata Parameter

Das Summenzeichen trifft den Begriff «Aggregated» terminologisch zwar nicht genau, ist jedoch für den Einsatz im schwarzen Punkt geeignet, da dort nochmals weniger Platz vorhanden ist als bei anderen Icons. Das Zeichen lässt in gewisser Weise darauf schliessen, dass es um das Zusammenfassen der Daten geht.

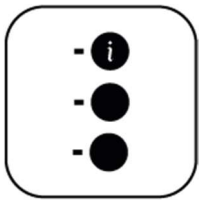
Provenance Metadata Parameter

Bei den Testpersonen setzte sich beim Provenance Metadata Parameter die Darstellung mit der Etikette durch. Dies hätte man auch generell zur Illustration des Metadata Parameters verwenden können. In diesem Fall gibt der Patternname der Visualisierung eine konkretere Bedeutung.

Control Metadata Parameter

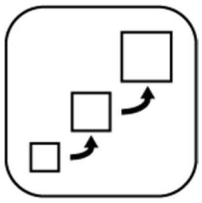
Eine Antenne wird unter anderem auch bei Fernbedienungen unterschiedlicher Art verwendet, um Befehle an ein Objekt weiterzugeben. Während zu Beginn solche Fernbedienungen visualisiert wurden, entstand am Ende des Designprozesses diese vereinfachte Version des Icons.

Annotated Parameter List



Dieses Pattern ist eine Kombination der Parameter List und der Metadata Parameter. Daher wurde in der Visualisierung ein Parameter der Parameter List durch einen Metadata Parameter ersetzt, womit nun konkreter auf den Inhalt der Liste geschlossen werden kann.

Evolution Kategorie



Während für das Projektteam der Runner Up dieses Icon (Evolutionslinien) der Favorit war, entschieden sich die Testpersonen für diese Version. Die Visualisierung zeigt auf, wie etwas schrittweise wächst und bringt damit den Kern dieser Patterns zum Ausdruck.

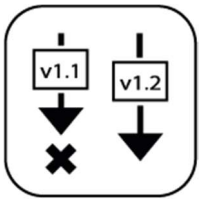
Semantic Versioning

Versionsnummern gehören zu den wenigen Visualisierungskomponenten der IRP, welche Buchstaben und Zahlen verwenden. Dieser Einsatz ist Zweckgebunden, da ohne grosse Umschreibung klar ist, wenn sich etwas ändert. Das v und die Zahl bringt dies besser zum Ausdruck, als durch willkürlich gewählte Ziffern.

Version Identifier

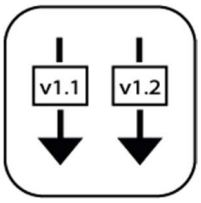
Dieses Icon stellt einen Batch dar, wie man ihn aus der Geschäftswelt kennt. Durch einen Batch wird eine Person innerhalb eines Betriebes identifiziert. Die Version eines APIs wird in diesem Pattern ähnlich über die URL der API ausgewiesen, ein sichtbares Kennzeichen, welches die richtige API identifiziert.

Aggressive Deprecation



Diese Visualisierung entstand auf Basis des Two-In-Production-Icons. Es hält den Zeitpunkt fest, an welchem das ältere der beiden APIs heruntergefahren wird.

Two in Production



Im Gegensatz zur Aggressive Deprecation verlaufen diese beiden Pfeile parallel, ohne dass einer davon zum Ende kommt.

Lifetime Guarantee

Lifetime Guarantee verspricht unbegrenzte Laufzeit einer API Version. Aus diesem Grund der Versionnummer in der Visualisierung das Unendlichkeitszeichen angefügt.

Semantic Category

Den Begriff «Semantik» zu visualisieren stellte die grösste Herausforderung in diesem Projekt dar, da dieser für etwas Abstraktes und Unfassbares steht. Das Resultat überzeugt noch nicht ganz. Das Icon soll ein Nachschlagewerk darstellen, in welchem Bedeutungen nachgelesen werden können.

Embedded Reference Data

Die Visualisierung zeigt einen Briefumschlag, in welche sich ein nicht-spezifizierter Parameter befindet. Das Einbetten des Parameters in einer Struktur soll durch das Einpacken zu Ausdruck kommen. Der Brief hat keine tiefere Bedeutung und ist nicht zwingend mit einer Nachricht in Verbindung zu bringen.

Linked Reference Data

In diesem Pattern wird nicht ein Objekt selbst eingepackt bzw. eingebettet, sondern die Referenz dazu. Links bringen solche Referenzen gut zum Ausdruck, daher wurde die Kette, welche in vielen Tools für die Verlinkung steht, in der Visualisierung eingesetzt.

Master Data

Der Zylinder wird allgemein für Visualisierungen eines Datastores verwendet. Die Krone auf dem Zylinder betont die Wichtigkeit der Daten und wird gleichzeitig mit dem Begriff «Master» in Verbindung gebracht.

Transaction Data

Während Master Data sich auf die Stammdaten bezieht, werden als Transaction Data solche Datensätze bezeichnet, welche oft ändern und deshalb ständig in Bewegung sind. In der Regel werden diese zur Verarbeitung der Master Data verwendet. Die Pfeile repräsentieren die Bewegung dieser temporären Daten.

ID Parameter

Bei diesem Icon wird wiederum der nicht-spezifizierte Parameter eingesetzt, auf welchem sich ein Strichcode befindet. Der Strichcode wird zur Identifizierung verwendet, wie der ID Parameter selbst, daher wurde diese Metapher gewählt.

Entity Parameter

Für dieses Icon wurde eine Visualisierung gewählt, wie man sie aus UML oder dem ERM kennt. Das weisse Rechteck stellt ein Objekt bzw. eine Entität dar.

Link Parameter

In diesem Parameter befindet sich ein Link, entsprechend kommt die Kette als weitverbreitetes Symbol für einen Link zum Einsatz.

6.2 Kompositionsbeispiel

Das nachfolgende Kompositionsbeispiel beschreibt einen API-Aufruf für eine Eventliste aus dem Google Calendar. Aus der Abbildung 6.1 ist die Response- und Reply-Struktur, sowie die Eigenschaften des Providers, welche für den Call von Bedeutung sind, ersichtlich.

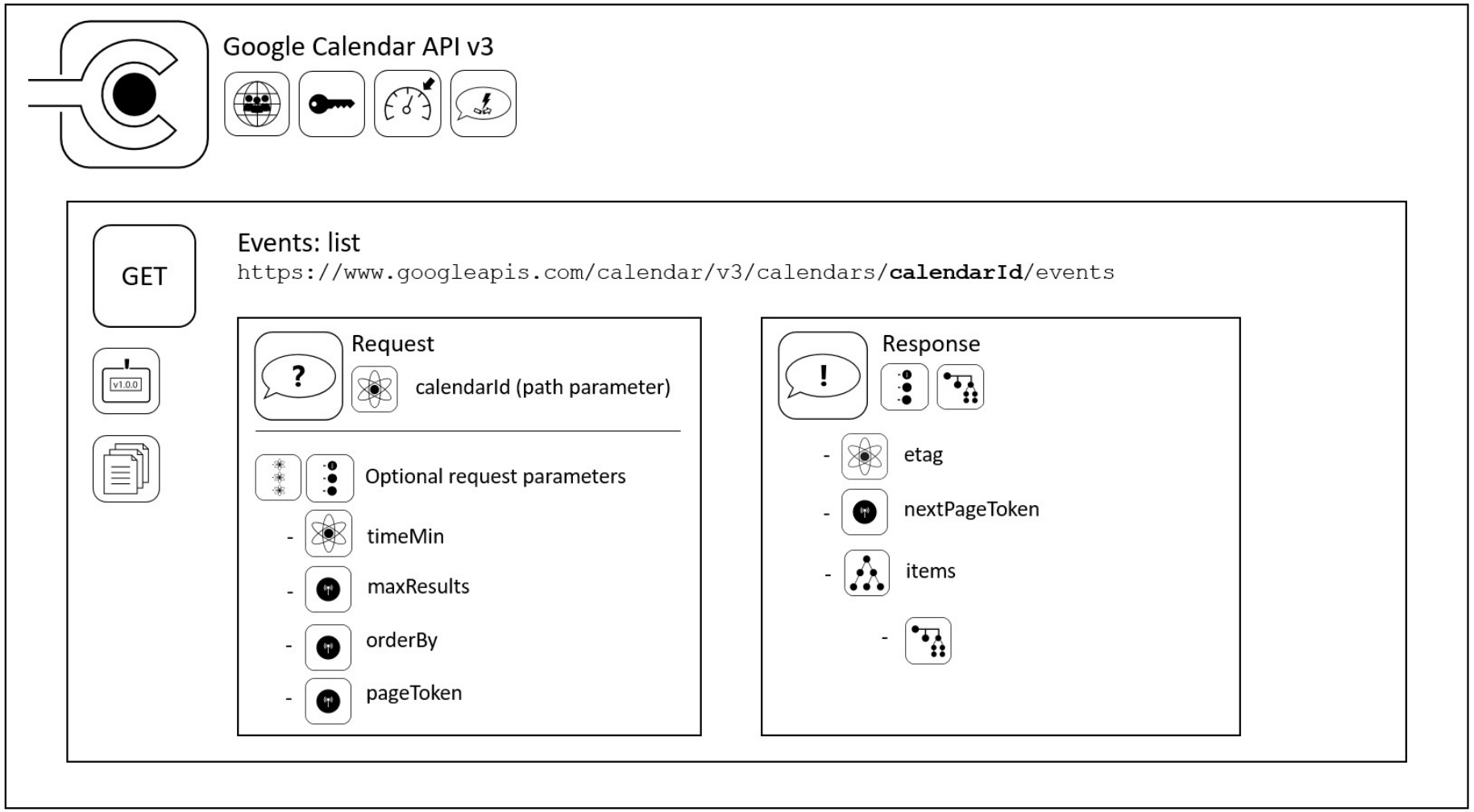


Abbildung 6.1: Kompositionsbeispiel für ein API Aufruf

6.3 Reflektion

In der folgenden Tabelle werden die erstellten Visualisierungen anhand der definierten Qualitätsmerkmale reflektiert.

Kriterium	Erfüllt?	Erläuterung
Visuelle Konsistenz	Ja	Durch Berücksichtigung der Guidelines und einem einheitlichen Gestaltungsstil ist die Konsistenz gegeben.
Abstraktionsgrad	Ja	Die Visualisierungen verwenden dieselben Konzepte auf einheitlicher Abstraktionsstufe.
Unterscheidbarkeit	Ja	Ähnliche Darstellungen wiederspiegeln dieselbe Semantik. Wo solche Ähnlichkeiten nicht erwünscht sind, unterscheiden sich die Visualisierungen stark. Kleine Unterschiede zwischen den Icons, wie beispielsweise bei den Pagination Patterns, sind gerechtfertigt.
Wiedererkennungswert	Keine Bewertung	Die Icons sind einheitlich Gestaltet und verfügen über einen potentiellen Wiedererkennungswert. Dies muss sich in Zukunft jedoch erst noch bewähren.
Farbkonzept	Teilweise	Die Visualisierungen sind in keiner Hinsicht abhängig von Farben und erfüllen das Schwarz/Weiss-Kriterium. Allerdings wurde das Farbschema vernachlässigt.
Multi-Channel Verwendung	Je nach Icon	Was elektronische und Druck-Kanäle angeht, erfüllen die Visualisierungen dieses Kriterium. Einzig was Handskizzen auf Papier bzw. Whiteboard angeht, fallen einige Icons durch. Es wurde darauf geachtet, dass Icons welche in Handskizzen-Szenarien oft zum Einsatz kommen das MCV-Kriterium stärker erfüllen als beispielsweise die Kategorie-Icons. Unter Umständen müssen gewisse Icons für diese Verwendung vereinfacht werden.

International verwendbar	Ja	Es wurden bis auf den Tempel keine regionsspezifischen Motive eingesetzt. Das Foundation Icon verliert dadurch jedoch nicht an Ausdruck, da sich dieser allgemein auf ein Bauwerk bezieht. Kulturbedingte Farbassoziationen können ebenfalls nicht auftreten.
Political Correctness	Ja	
Skalierbarkeit	Ja	Nicht alle Icons können gleich stark nach unten skaliert werden, ohne an Ausdruck zu verlieren. Auf einer Minimalgrösse von 2x2cm sind jedoch sämtliche Motive noch erkennbar.
Master / Detail View	Nein	Die Darstellungen der internen Abläufe wurden nicht umgesetzt. Diese sind stark von der Implementation des Patterns abhängig.
Zweidimensionale Darstellung	Je nach Icon	Die Mehrheit der Icons ist zweidimensional. In einigen Patterns werden dreidimensionale Perspektiven verwendet (z.B. Request Bundle). Diese hält sich jedoch in Grenzen und dient lediglich der besseren Erkennung des Motives. Die visuelle Konsistenz wird dadurch nicht gestört.
Erweiterbarkeit	Ja	Der Rahmen ist durch das Visualisierungskonzept gegeben. Die Visualisierungen können beliebig erweitert werden.
Pattern-Kategorien	Nein	Die Patterns derselben Kategorie weisen oft Gemeinsamkeiten auf (z.B. Pagination oder Metadata Parameters). Allerdings können nicht alle Patterns einzig aufgrund der visuellen Informationen des Icons einer Kategorie zugeordnet werden.
Lizenzen	Ja	Die Grafiken wurden eigenständig und unter Berücksichtigung von vorhandenen Visualisierungsideen entworfen.

Tabelle 6.1: Reflektion anhand der Qualitätsmerkmale

Durch die Tabelle wird deutlich, dass bei Visualisierungen je nach Kriterium immer wieder ein Trade-Off zwischen den Qualitätsmerkmalen und der Ästhetik im Auge der Betrachter/Designer stattgefunden hat. Durch den iterativen Prozess der gesamten Studienarbeit verloren gewisse Kriterien wie die Master / Detail – View oder das Farbkonzept an Bedeutung. Der Auftragsgeber wurde diesbezüglich informiert. So wurde auch das Qualitätsmerkmal der Zweidimensionalität relativiert. Nichtdestotrotz legt diese Tabelle offen, dass die Visualisierungen noch über Verbesserungspotential verfügen. Was die Pattern-Kategorie-Icons angeht, wäre es wünschenswert, wenn die Visualisierungen so verbessert werden, dass man sie auf den ersten Blick einordnen kann. Das Qualitätsmerkmal der Multi-Channel-Verwendung sollte in Hinsicht auf Handskizzen nochmals überdacht werden. Je nach Entschluss müssen entweder einige bestehende Visualisierungen oder die Anforderungen an die Visualisierungen angepasst werden.

Kapitel 7

Testverfahren und Ergebnisse

Dieses Kapitel zeigt die Testmethoden auf, welche in Bezug auf die Prüfung der finalen Grafiken zum Einsatz kamen.

Da bekannte Testverfahren für das Testen der Illustrationen eingesetzt werden können, wird vorab auf die verschiedenen Arten von Testmöglichkeiten eingegangen.

Die bis dato im Studium in verschiedenen Projekten angewandten Testverfahren stammen aus den Bereichen Software Engineering und Human Computer Interaction Design (nachfolgend HCID genannt).

Der Software Engineering Bereich deckt Testmethoden wie Komponenten-, Integrations-, System- und Abnahmetests ab. [8] Weitere Testmöglichkeiten aus diesem Bereich können mit White-Box und Black-Box Tests umgesetzt werden. Während in einem Softwareprojekt bei einem Black-Box Test die innere Struktur des zu testenden Elements nicht bekannt ist, ist es bei einem White-Box Test zwingend, die innere Struktur des Testobjekts zu kennen. [10]

Ein Usability-Test zeigt die Benutzbarkeit eines Systems. Michael Richter et al. gehen vertieft in dieses Fachthema ein. [6] Ihre Aussage ist, dass eine hohe Usability darauf hinweise, dass ein System von der entsprechenden Zielgruppe einfach erlernt und effizient verwendet werden kann.

7.1 Icon-Test

Betrachtet man die oben aufgeführten Testmethoden, kann ein «Icon-Test» im weiteren Sinne mit einem Komponententest verglichen werden. Der Testgegenstand ist die Pattern-Visualisierung und wird in erster Linie als einzelne Visualisierungskomponente der IRP betrachtet. Wird im Testverfahren einer Testperson eine Grafik ohne jeglichen Kontext (wie Patternname) vorgelegt, kann der Black-Box Test hierfür verwendet werden. Wird jedoch während des Tests der Patternname und die Funktionsweise bekanntgegeben, so handelt es sich um einen White-Box Test.

Zum Einsatz kam diese Testmethode in den Schritten 3 «Feedback Auftraggeber» und 5 «Feedback von Extern» gemäss dem Prozessmodell in Kapitel 5. Das Projektteam legte im Schritt 3 dem Auftraggeber die ersten Visualisierungsentwürfe vor. Dieser testete die Grafiken

auf die im Absatz 5.2.3 aufgelisteten Fragestellungen (meist im Beisein des Projektteams), woraufhin unter anderem Diskussionspunkte und Vorschläge für Verbesserungen entstanden oder ein Visualisierungsentwurf als gut bewertet wurde.

In Schritt 5 holte der Auftraggeber die Rückmeldungen zu den Icons bei insgesamt sechs externen Testpersonen ein. Diese Befragungen fanden im letzten Drittel der Studienarbeit statt, womit ungefähr 90% der Visualisierungen einer externen Kontrolle unterstanden. Die Befragung der Testpersonen (ausgenommen Pattern-Autoren) fand in einem persönlichen Gespräch statt. Der Vorteil einer mündlichen Befragung ist, dass der Auftraggeber auf jegliche Art von Reaktionen schnell reagieren und das Feedback detailliert aufnehmen kann.

Die Testpersonen hatten bei den Befragungen unterschiedliche Hintergrundinformationen zu den IRP, sie wurden einerseits vorab per E-Mail instruiert und andererseits fand am Tag der Befragung, je nach befragter Person, eine mündliche Orientierung statt. Die Befragten durchliefen einen *Icon-Walkthrough* Test, in welcher ihnen die Grafiken vorgelegt wurden und sie gebeten wurden sich zu äussern zu:

- Kann das Pattern-Icon dem korrekten Pattern zugewiesen werden?
- Was symbolisiert die Grafik?

Die Pattern-Autoren, welche solides Vorwissen zu den IRP mitbringen, bewerteten die Patterngrafiken mittels White-Box Testverfahren.

Die Tabelle 7.1 zeigt das externe Feedback in vereinfachter Form auf.

Testperson	Vorwissen	Feedback (Auszüge)	Datum	Feedback
Doktorand an einem deutschen industrienahen Forschungsinstitut	Forschungskontext: Ähnliche Patterns wie IRP, jedoch noch kein umfassendes Vorwissen zu IRP	- Die meisten Patterns sind verständlich umgesetzt - Grafiken zu den Pagination Pattern nicht verständlich und neue Designvorschläge - Neuer Vorschlag für Evolution Pattern Icon - Rate Limit Icon mit Monitoring assoziiert - Version Identifier Icon einfacher darstellen	27.04.17	persönlich
Pattern Co-Autor	Vorwissen vorhanden	- Visualisierungen sehen sehr gut aus und sind gut mit Patternaussagen assoziierbar. - Icons sind schwierig per Hand zu zeichnen	12.05.17	schriftlich
Dozent an deutscher Fachhochschule	Wenig Vorwissen (Instruktion der IRP über E-Mail)	- Insgesamt guter Eindruck (sinnvoll und gut entworfen)	19.05.17	persönlich
Software-Architeckt	IRP aus einer vorherigen mündlichen Kurzvorstellung bekannt	- Vorschlag: SLA Icons zusammenführen - Papierkorb bei Icon Aggressive Deprecation nicht erkannt, Vorschlag für neues Icon - Verbesserungsvorschlag Foundation Category Icon (weisse Balken mit schwarzer Umrandung)	22.05.17	persönlich
Pattern Co-Autor	Vorwissen vorhanden	- Detailliertes Feedback erfolgt nach der Abgabe dieser Arbeit. Erster Eindruck: Konsistenz gut eingehalten	Mai 2017	schriftlich

Professor an deutscher Fachhochschule	Wenig Vorwissen (Instruktion der IRP über E-Mail)	- Neuer Vorschlag für Error Reporting, da Reporting aus der Visualisierung nicht hervorgeht - Kette bei Parameter Link nicht intuitiv	Mai 2017	persönlich

Tabelle 7.1: tabellarische Übersicht der externen Feedbacks

Die Interviewergebnisse der Testpersonen wurden vom Auftraggeber an das Projektteam schriftlich kommuniziert und in einem zweiten Schritt in einer Sitzung besprochen. Das Projektteam setzte aufgrund der Erkenntnisse dieser Sitzung die Verbesserungsvorschläge um und erarbeitete neue Visualisierungsentwürfe für neue Gestaltungsideen.

Umsetzungsbeispiel

Eine konstruktive Rückmeldung enthielt die Aussage, dass bei der *Agressive Depreciation* Visualisierung ebenfalls die Timeline von *Two in Production* verwendet werden kann. Aus diesem Grund ist die mittlere Grafik in Abbildung 7.1 neu ausgearbeitet worden. Diese beinhaltet nun ebenfalls die Timelines und bildet mit den anderen Patterns aus dieser Kategorie eine Einheit.

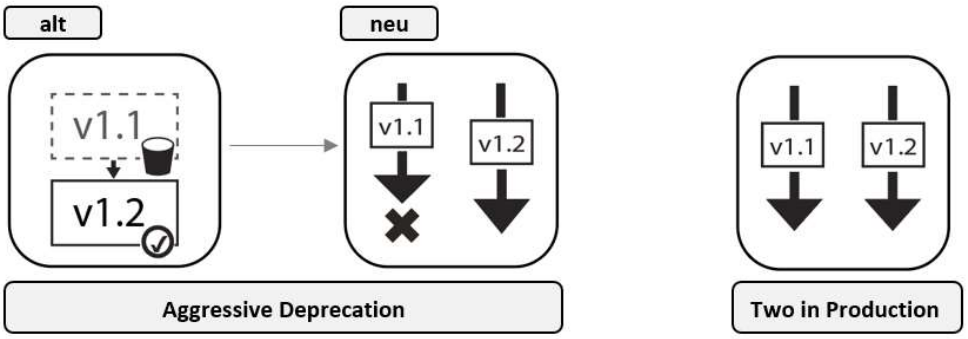


Abbildung 7.1: Umsetzungsbeispiel Aggressive Deprecation nach externem Feedback

In einer anderen Rückmeldung wurde die Visualisierung des Rate Limit Pattern nicht korrekt verstanden und mit einem Überwachungssystem assoziiert. Die Visualisierung wurde auf die Tachometer-Komponente reduziert. Des Weiteren bestand der Bedarf für eine neue Alternativvisualisierung. Abbildung 7.2 zeigt auf, wie die Umsetzung des *Rate Limit* Icons stattgefunden hat. Einerseits wurde der ältere Entwurf komplett neu überarbeitet (neu: mittlere Grafik) und andererseits eine Grafik neu erstellt (neue Alternative).

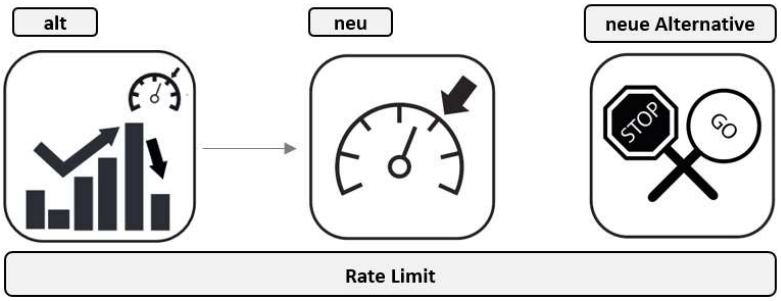


Abbildung 7.2: Umsetzungsbeispiel Rate Limit nach externem Feedback

7.2 Whiteboard-Test

S. Ambler erläutert in seinem Buch Agile Modeling [2] unter anderem, dass der Einsatz eines Whiteboards einem Team helfen kann, neue Designalternativen herauszufinden und die Kommunikation innerhalb des Teams zu stärken. Im Software Engineering erlaubt es diese Variante, innerhalb des Projektes agil vorzugehen.

Das Whiteboard kam auch für die Visualisierungen in Frage. Es sollten einzelne Komponenten für die Umsetzung eines API-Designs (zum Beispiel API-Aufruf oder API-Endpunkt) zusammengesetzt werden. Dieser Ansatz ist vergleichbar mit einem Integrationstest.

Das Whiteboard-Testing fand nach der Hälfte der Studienarbeit mit dem Auftraggeber statt. Ziel dieses Testverfahrens war es, herauszufinden, ob die in Absatz 4.2 beschriebenen Kernaufgaben der User Stories mit den bis dato bestehenden Visualisierungsentwürfen umgesetzt werden können. Zusätzlich wurde geprüft, ob sich die Visualisierungen von Hand zeichnen lassen und wie ein API-Aufruf mit den einzelnen Komponenten zusammengesetzt werden kann.

Die Kompositionsbeispiele im Kapitel 6 «Endprodukt» basieren auf den Ergebnissen der Whiteboard-Session.

Kapitel 8

Rechercheergebnisse

8.1 Ausgangslage

Dieses Kapitel befasst sich mit den Ergebnissen der Web-Recherche mit Bezug auf die Analyse der IRP und deren Vorkommnisse in öffentlich zugänglichen Web-APIs. Am Ende der Studienarbeit ist ein 80-seitiges Web-Recherche Dokument entstanden, in welchem die nachfolgend aufgeführten APIs analysiert und dokumentiert wurden. Für die Analyse wurden APIs aus verschiedenen Kategorien ausgewählt:

- Facebook Graph API (v2.9)
- Twitter REST API
- YouTube Data API (v3)
- GitHub v3 API
- Google Calendar API
- Stripe API

Das Dokument kann von den Pattern Autoren konsultiert werden, um sich über die «Known Uses» zu informieren.

Diese Dokumentation reflektiert den aktuellen Stand der Patterns und ist nicht als abschliessend oder final zu betrachten.

In Absatz 8.3 «Ergebnisse» erfolgt eine Zusammenfassung der Web-Recherche.

8.2 Einleitung

Die nachfolgenden Absätze zu den einzelnen Patterns beinhalten keine detaillierten Beschreibungen der IRP, sondern Erläuterungen zu den Recherche-Ergebnissen. Es wird aufgezeigt, welche Patterns in welcher Form in den untersuchten Web-APIs vorkommen und wie häufig sie Anwendung finden. Wo sinnvoll, werden die Patterns mit Beispielen ergänzt.

Im Absatz 8.3 werden 20 der analysierten IRP mit Bezug auf die Web-Recherche näher erläutert. Die Recherchetätigkeiten basierten auf öffentlichen APIs. Aus diesem Grund wurden die *Foundation Patterns* vom Projektteam nicht näher analysiert.

Des Weiteren wurden die *Semantic Patterns* und *Parameter Types* aus Zeitgründen nicht eingehend recherchiert und dokumentiert, weshalb diese nicht Teil dieser Zusammenfassung sind.

8.3 Ergebnisse

8.3.1 Basic Representation Patterns

8.3.1.1 Atomic Parameter

Dieses Pattern wird von 5 der 6 analysierten Web-APIs umgesetzt und zwar immer dann, wenn nur ein einziger Anfrageparameter in einer Anfrage enthalten ist und diese für eine bestimmte Ressource gemacht wird. Ein Event kann als eine Ressource angesehen werden und die URL des Aufrufs sieht im Fall von Stripe API wie folgt aus:

```
https://api.stripe.com/v1/events/<event-id>
```

8.3.1.2 Atomic Parameter List

Dieses Pattern findet in allen analysierten APIs häufige Anwendung. Im Vergleich zum *Atomic Parameter* enthält die Anfrage mehr als einen Anfrageparameter. Diese Parameter sind meistens optional.

```
https://api.stripe.com/v1/events?type=customer.deleted&limit=5
```

8.3.1.3 Parameter Tree

Die untersuchten APIs zeigen auf, dass dieses Pattern nicht oft anzutreffen ist. Die Twitter REST API setzt dieses Pattern beispielsweise ein, wenn eine GET-Anfrage für eine Liste gemacht wird, indem nur ein «Root Node» retourniert wird.

Eine Fehlermeldung wird oftmals auch in Form eines *Parameter Tree* repräsentiert. Der Absatz 8.4.3.3 «Error Reporting» zeigt solch eine Fehlermeldung auf.

8.3.1.4 Parameter Comb

Dieses Pattern findet in allen untersuchten APIs sehr häufige Anwendung. Bei diesen APIs wird auf einen API-Aufruf mehrheitlich ein *Parameter Comb* zurückgegeben. Nebst dem Einstiegsobjekt, in welchem sich die nachgefragten Ressourcen befinden, werden meist Metadata-Parameter retourniert. Nachfolgendes Beispiel stellt die Antwort in Form eines *Parameter Comb* dar und zwar in Bezug auf eine Videosuche in der YouTube Data (v3) API.

```
{
   "kind": "youtube#searchListResponse",
   "etag": etag,
   "nextPageToken": "CAUQAA",
   "regionCode": "CH",
   "pageInfo": { ...},
   "items": [
     {...},
   ]
}
```

8.3.2 Pagination Patterns

Die Analyse hat gezeigt, dass die Pagination Patterns in allen analysierten Web-APIs vorkommen, jedoch in unterschiedlichen Varianten. Durch Pagination wird erreicht, dass nur eine überschaubare Menge an Ressourcen pro Seite aufgelistet wird. Je nach API, sind Navigationsinformationen im entsprechenden Response Body oder im HTTP-Link-Header zu finden.

Arten von Pagination

8.3.2.1 Offset (Page)-Based Pagination

Der numerische Offset verweist auf den ersten Eintrag einer Seite. Diese Variante wird von vielen APIs angeboten. Es werden zwei Anfrageparameter gebraucht (`offset`, `limit`). Je nach API können unterschiedliche Bezeichnungen gebraucht werden, wie zum Beispiel `page` für offset oder `count` für limit.

```
https://graph.facebook.com/v2.9/search?q=HSR&type=page&offset=2&limit=3
```

Diese Art der Pagination kann jedoch zu Duplikaten oder fehlenden Einträgen führen, wenn während der Navigation Einträge neu eingefügt oder gelöscht werden.

8.3.2.2 Time-Based Pagination

Bei dieser Art der Pagination beinhalten die einzelnen Einträge je einen Zeitstempel (z.B. Unix Zeitstempel wie dies bei Facebook Graph API der Fall ist). Die Navigation durch die Ergebnisse erfolgt durch diese Zeitstempel. Es werden drei Anfrageparameter gebraucht (`since`, `until`, `limit`), wobei auch diese Parameter andere Bezeichnungen haben können.

```
https://graph.facebook.com/v2.9/164202036981850/feed?since=1483257600&until=1485849600&limit=10
```

8.3.2.3 Cursor-Based Pagination

Diese Art der Pagination hat eine ähnliche funktionsweise wie «Time-Based Pagination», benutzt aber für die Navigation einen oder mehrere eindeutige Zeiger (cursors), welche zum Beispiel den ersten oder letzten Seiteneintrag identifizieren. Oftmals wenn eine Anforderung an Echtzeitdaten besteht, wird diese Variante der Pagination bevorzugt. Die nachfolgende URL beinhaltet zwei Parameter (`limit`, `after`), wobei der `after` Zeiger sich auf das Ende der Seite bezieht.

```
https://graph.facebook.com/me/albums?limit=25&after=MTAyMDkzNjEyNDA2NDA4MTMZ
```

8.4 Advanced Representation Patterns

8.4.1.1 Wish List

Dieses Pattern wird in 3 der 6 untersuchten Web-APIs eingesetzt. Die Facebook Graph API, Google Calendar API und YouTube Data API (v3) verwenden im API-Aufruf den `field` Parameter, um die API-Antwort nach den gewünschten Attributen zu filtern. Die aktuelle Version der GitHub API unterstützt dieses Pattern nicht, jedoch hat sie ein «early access programm» namens GitHub GraphQL (noch nicht für Produktion verfügbar), welches dieses Pattern umsetzt.[20]

Besonders in der YouTube Data API (v3) ist der Gebrauch dieses Patterns sinnvoll, da die Zugriffe auf ihr API über Quotas geregelt werden und so den API-Konsumenten mehr Kontrolle über die *Rate Limitation* gegeben wird.

```
https://graph.facebook.com/v2.9/340703149643148?fields=description,name
```

Die Twitter REST API und Stripe API bieten gemäss den Recherchen keine Filterungsmöglichkeit an.

8.4.1.2 Wish Object

Dieses Pattern wird von den gleichen Web-APIs umgesetzt wie in *Wish List* aufgeführt und gibt den API-Konsumenten die Möglichkeit, die Struktur der Antwort bzw. deren Repräsentation selber zu bestimmen. Untenstehendes Beispiel zeigt, wie das Pattern in der Google Calendar API angewendet wird (der API-Konsument möchte die Antwort auf den Titel sowie die Start- und Endzeit der bevorstehenden drei Events aus dem Primary Calendar begrenzen).

```
https://www.googleapis.com/calendar/v3/calendars/primary/events?timeMin=
2017-04-25T00:00:00Z&orderBy=startTime&singleEvents=true&maxRe-
sults=3&fields=items(summary,start/dateTime,end/dateTime)
```

Die Rechercheergebnisse zeigen auf, dass die Patterns *Wish List* und *Wish Object* sehr nahe beieinanderliegen und in Kombination auftreten. Die Hauptvorteile in Verwendung dieser Patterns sind (i) Erhöhung der Performance sowie (ii) Reduzierung der Latenzzeit (da der Server nicht zusätzlich unnötige Metadaten zurückgeben muss).

8.4.1.3 Request Bundle

Dieses Pattern wird von den gleichen Web-APIs wie in *Wish List* aufgeführt und zusätzlich von der Twitter REST API umgesetzt. Die Rechercheergebnisse zeigen auf, dass es in den APIs zwei unterschiedliche Arten gibt, API-Aufrufe zu bündeln und die Anzahl der HTTP Verbindungen zu reduzieren. Auch dieses Pattern reduziert den allgemeinen HTTP Overhead.

Multiple-ID Lookup Requests

Diese Variante ermöglicht es dem API-Konsumenten, in einer einzigen GET-Anfrage mehrere Ressourcen, welche eine eindeutige ID haben, abzufragen. Die Facebook Graph API benutzt für solch eine Abfrage den `ids` Endpunkt mit den entsprechenden Ressourcen-IDs.

```
https://graph.facebook.com/v2.9?ids=164202036981850,768635709814450
```

Auch die Twitter API verfolgt eine ähnliche Variante für die Umsetzung dieses Patterns:

```
https://api.twitter.com/1.1/sttuses/lookup.json?id=20,432656548536401920
```

Hier ist zu beachten, dass die Anzahl Aufrufe in einer einzigen HTTP beschränkt sein kann, bei Facebook ist die Grenze bei 50, bei Twitter beläuft sich die Anzahl `ids` auf 100.

Der *Request Bundle* kann mit den Patterns *Wish List* und *Wish Object* kombiniert werden und gibt dem API-Konsumenten die Flexibilität, die Antwort nach Bedarf zu strukturieren.

Batch Requests

Im Gegensatz zu *Multiple-Id Lookup Requests* ermöglicht diese Variante, mehrere unabhängige HTTP Operationen (GET, POST, PUT, DELETE) in eine einzige HTTP-Anfrage einzuschliessen. Eine Anwendung dieses Patterns konnte in der Twitter REST API und YouTube Data API (v3) nicht gefunden werden.

Facebook unterstützt in ihrer API diesen Vorgang und arbeitet die abhängigen Operationen sequentiell ab, die unabhängigen Operationen parallel. Der API-Konsument erhält nach erfolgreicher Ausführung aller Operationen eine einzelne Antwort.

Auch Google Calendar API setzt dieses Pattern um.

Zu beachten ist, dass es auch bei Batch Requests eine Beschränkung in Bezug auf die Anzahl Aufrufe in einer einzigen HTTP Nachricht gibt.

8.4.1.4 Metadata Parameter

In der Recherchetätigkeit lag das Augenmerk auf Metadaten, welche der API-Provider über den *Response Body* mitgibt. Die Analyse des *Link Header* war nicht Teil der Recherche.

Recherchiert wurden die am Anfang des Kapitels erwähnten APIs. Metadaten sind in diesen APIs gängig und werden je nach API-Aufruf verschieden angewandt. Bei einem API-Aufruf für eine *Collection* werden zum Beispiel Metadaten für die Navigation retourniert (im *Link Header* oder *Response Body*).

Die Verwendung von Metadaten variiert stark je nach verwendeter API. Die Facebook Graph API gibt zum Beispiel wenig bis keine Metadaten im *Response Body* zurück. Die Twitter REST API beinhaltet hingegen sehr viele Metadaten in Bezug auf die Entität, aber nicht auf die Antwort selbst. Das Design der YouTube Data API (v3) ist so aufgebaut, dass in jeder Antwort die Metadaten `kind`, `etag`, `pageInfo` mitgegeben werden, die verschiedene Arten von Metadatentypen verkörpern. Die GitHub API v3 und Stripe API verwenden ebenfalls Metadateninformationen (je nach Aufruf).

a) Control Metadata Parameter

Die Facebook Graph API liefert auf eine *GET Collection* Anfrage in ihren Antworten im Response Body ein paging-Objekt für die Navigation. Dies ist ein Verwendungszweck dieses Patterns.

Auch die Twitter REST API, YouTube Data API (v3) und Google Calendar API setzen dieses Pattern mit Bezug auf die Navigation, wie nachfolgend im Beispiel von Twitter gezeigt, ein.

```
{
  "users": [
   ...
  ],
  "next_cursor": 1489467234237774933,
  "next_cursor_str": "1489467234237774933",
  "previous_cursor": 0,
  "previous_cursor_str": "0"
}
```

Die GitHub API v3 benutzt keine Navigations-Metadaten im *Response Body*, hat aber ein `incomplete_results` Parameter im Einsatz, welcher bei einer Suchanfrage im *Response Body* zurückgeliefert wird. Diese Metainformation gibt Auskunft darüber, ob die API-Antwort auf die Suchanfrage alle Resultate anzeigt. Ist dieser Wert auf `true` gesetzt, weist das darauf hin, dass die Suchanfragen das Zeitlimit überschritten hat und jene Antworten zurückgeliefert werden, die vor dem Timeout bereits gefunden worden waren.

Die Stripe API hat in ihren Antworten im Response Body den `has_more` Parameter, welcher auch für die Navigation verwendet wird. Ist der Wert dieses Parameters auf `false` gesetzt, so ist man am Ende der Liste angelangt.

b) Provenance Metadata Parameter

Die YouTube Data API (v3) sowie die Google Calendar API verwenden die Metadaten `kind` und `etag`, welche dieses Pattern repräsentieren. Der `kind` Parameter identifiziert den Typ der Ressource. Die GitHub API v3 retourniert zum Beispiel auf eine *GET Collection* Anfrage ein Array mit den entsprechenden Ressourcen, wobei jede Ressource mit einem `id` Parameter identifiziert wird. Die Stripe API retourniert ein `object` Parameter, womit der zurückgegebene Objekttyp beschrieben wird. Das nachfolgende Beispiel zeigt die Antwort der Stripe API auf eine *GET /v1/customers* Anfrage.

```
{
  "object": "list",
  "has_more": true,
  "url": "/v1/customers"
  "data": [
    {...}
  ],
}
```

c) Aggregated Metadata Parameter

Dieses Pattern findet unter anderem Anwendung in der YouTube Data API (v3). Im *Response Body* wird das Objekt `pageInfo` standardmässig mitgegeben, welches die Metadata Parameter `totalResults` und `resultsPerPage` enthält.

```
{
 "kind": "youtube#searchListResponse",
 "etag": etag,
 "pageInfo": {
    "totalResults": 1662,
    "resultsPerPage": 10
  },
 "items": [
  search Resource
  ]
}
```

Die GitHub API v3 gibt auf eine Suchanfrage das `total_count` Parameter zurück, was im obigen Beispiel dem `totalResults` gleichgestellt werden kann.

Die Tabelle 8.1 zeigt eine Übersicht der Metadatentypen und deren Vorkommen in den entsprechenden APIs.

API	Provenance Metadata Parameter	Control Metadata Parameter	Aggregated Metadata Parameter
Facebook Graph API	Nein	ja	nein
Twitter REST API	ja	ja	nein
YouTube Data API (v3)	ja	ja	ja
GitHub API v3	ja	ja	ja
Google Calendar API	ja	Ja	nein
Stripe API	ja	Ja	Ja

Tabelle 8.1: Übersicht Metadata Parameter

8.4.1.5 Annotated Parameter List

Die untersuchten API-Provider setzen dieses Pattern um. Zumeist wird das Ergebnis einer Anfrage in Form einer *Annotated Parameter List* repräsentiert, wenn der API-Aufruf sich auf eine Liste von Ressourcen bezieht bzw. eine Suchanfrage gemacht wird. Die Antworten der API-Provider beinhalten mehrheitlich dieses Pattern, wie auch unten dargestellt (GitHub Data v3 API). Nebst den eigentlichen Ressourcen enthält die Antwort Metadaten.

```
{
  "total_count": 10,
  "incomplete_results": false,
  "items": [
    {
      "login": "mottosso",
      "id": 2152766,
      …
    },
    {
        Further items
    }
  ]
}
```

8.4.2 Basic Evolution

8.4.2.1 Semantic Versioning

Alle der untersuchten APIs haben dieses Pattern umgesetzt, jedoch in unterschiedlichen Varianten. Nachfolgende Tabelle gibt Auskunft über die Art und Weise wie *Semantic Versioning* eingesetzt wird.

Name API	Version	Kommentar
Facebook API	V2.9	Facebook bietet eine 2-jährige Garantie für die Verfügbarkeit ihrer Hauptkomponenten.
Twitter REST API	V1.1	Keine ausführliche Dokumentation zur Versionierung gefunden. Diese aktuelle Version wurde im September 2012 veröffentlicht.
YouTube Data API (v3)	v3	Setzt «Semantic Versioning» in vereinfachter Form um. Die Versionen v1, v2 und v3 haben fundamentale Unterschiede, woraus zu schliessen ist, dass diese «Major-Releases» darstellen.
GitHub v3 API	v3	Gleicher Kommentar wie bei YouTube Data API (v3), die Versionen v1 und v2 werden seit Juni 2012 nicht mehr unterstützt.
Google Calendar API	v3	Setzt auch eine vereinfachte Form von «Semantic Versioning» um. Die Versionen v1 und v2 werden seit November 2014 nicht mehr unterstützt. Es ist daraus zu schliessen, dass diese Versionen «Major-Releases» darstellen.
Stripe API	v1 – 2017-04-06	Setzt auch eine vereinfachte Form von «Semantic Versioning» um. In den Anfragen wird jeweils in der URL die Version v1 angegeben, welche als «Major-Version» bezeichnet werden kann. Gemäss Dokumentation ist die aktuellste Version ihrer API 2017-04-06, welche als «Minor-Version» beziehungsweise zeitbasierte Version bezeichnet wird. Diese API Version wird bei der ersten Implementierung von Stripe automatisch definiert.

Tabelle 8.2: Übersicht Umsetzung Semantic Versioning

8.4.2.2 Version Identifier

Alle der untersuchten APIs setzen dieses Pattern ein. Die Recherche mit Bezug auf die Versionierung hat aufgezeigt, dass verschiedene Varianten für die Umsetzung vorhanden sind. Die GitHub v3 API benutzt für die Versionierung ihren eigens erstellten Media Type, welcher dem «Accept Header» hinzugefügt werden kann. Die restlichen APIs, ausgenommen die Stripe API, setzen die Versionierung in ihren URLs um.

Die nachfolgende Tabelle zeigt einen Überblick über die Art der Versionierung.

Name API	**Versionierungsart**	**Beispiel**
Facebook API	URL	https://graph.facebook.com/**v2.9**/2031763147233/
Twitter REST API	URL	https://api.twitter.com/**1.1**/search/tweets.json
YouTube Data API (v3)	URL	https://www.googleapis.com/youtube/**v3**/videos?id...
GitHub v3 API	Accept Header	Accept: application/vnd.github.**v3**+json
Google Calendar API	URL	https://www.googleapis.com/calendar/**v3**/calendars/calendarId/events/eventId
Stripe API	URL	GET https://api.stripe.com/**v1**/customers *

Tabelle 8.3: Versionierungsart der untersuchten API

* Gemäss Dokumentation der Stripe API wird die API Version bei der ersten Implementation der API gesetzt. Man kann dies zu einem späteren Zeitpunkt manuell über das Dashboard anpassen. Die Stripe API garantiert ihren Kunden, eine funktionierende Integration bestehen zu lassen und stets eine Rückwärtskomptabilität zu gewährleisten, somit hat sie eine sehr grosse Anzahl API Versionen.[21]

Die Facebook Graph API sowie GitHub v3 API ermöglichen auch API-Aufrufe ohne Angabe einer Versionsnummer. Bei Facebook Graph API ist hier Achtung geboten, da solch ein Aufruf sich auf die älteste verfügbare Version der API bezieht.

8.4.3 QoS Patterns

8.4.3.1 Rate Limit

Alle der recherchierten APIs unterstützen dieses Pattern, indem sie Anfragen auf Ressourcen so begrenzen, dass beispielsweise nur eine bestimmte Anzahl von Anfragen während einer bestimmten Zeitdauer gestellt werden kann. Mit diesem Vorgehen kann die Verfügbarkeit der Ressourcen für die API-Konsumenten gewährleistet werden. Die Umsetzung folgt auf unterschiedliche Weisen. In vielen Fällen folgt zusätzlich eine Unterscheidung, ob ein authentisierter oder ein nicht authentisierter Aufruf stattfindet.

GitHub verwendet eigene HTTP Header (`X-RateLimit-Limit`, `X-RateLimit-Remaining` und `X-RateLimit-Reset`) für die Umsetzung. Die YouTube Data API (v3) verwendet Quotas, je nach Art der Operation (Lese-, Schreiboperation) kostet diese 1 – 50 Units.

8.4.3.2 API Key

Dieses Pattern wird von allen untersuchten APIs umgesetzt. Teilweise wird in diesen APIs der Begriff *Access Token* verwendet, was auch diesem Pattern zugeordnet werden kann. Der Hauptunterschied zwischen einem *API Key* und *Access Token* besteht in deren Verwendung. Ein *Access Token* ist temporär gültig und wird meistens dann erstellt, wenn OAuth benutzt wird. Ein *API Key* wird zusammen mit einer Anfrage an einen Web Service geschickt, meist um das aufrufende Programm zu identifizieren. Der API Key ist im Gegensatz zu einem Access Token nicht temporär und bleibt in der Regel unverändert.

8.4.3.3 Error Reporting

Alle untersuchten APIs setzen Error Reporting auf eine gute Weise um, indem die API eine für den Menschen verständliche Fehlernachricht sowie einen eindeutigen Fehlercode retourniert. In den entsprechenden API-Dokumentationen kann der Fehlercode jeweils nachgeschlagen werden. Die GitHub v3 API und Facebook Graph API haben beispielsweise in ihren Antworten auch einen Link zur Dokumentation. Nachfolgend eine Fehlermeldung, wie sie von Facebook für eine nichtexistierende Ressource generiert wird.

```
{
  "error": {
    "message": "Unsupported get request. Object with ID '2031763147231' does
not exist, cannot be loaded due to missing permissions, or does not support
this operation. Please read the Graph API documentation at https://develop-
ers.facebook.com/docs/graph-api",
    "type": "GraphMethodException",
    "code": 100,
    "fbtrace_id": "AeVsq4tMvJt"
  }
}
```

8.4.3.4 SLA-SLO

Ein Service Level Agreement konnte bei keinem der analysierten APIs gefunden werden, somit waren auch keine Angaben zur gewährleisteten Verfügbarkeit oder zu Rückvergütungen zu finden. Alle APIs stellen jedoch sicher, dass sie durch *Rate Limits* die gleichen Bedingungen für ihre API-Konsumenten setzen.

Die GitHub API hat ein Kapitel «API Terms» in ihren Servicebestimmungen, welche für die Nutzung des API akzeptiert werden müssen. Diesem Beispiel folgt auch die You-Tube Data API.

Kapitel 9

Inhalte für Folgearbeiten

Dieses Kapitel hält fest, auf welche Aspekte bei einer Folgearbeit geachtet werden sollte und welche Arbeiten im Rahmen dieser Studienarbeit nicht fertiggestellt werden konnten.

9.1 Tipps und Tricks

Das Visualisierungskonzept beinhaltet alle relevanten Punkte für die Erstellung der Grafiken.

Bei dem Visualisieren sollte von Anfang an darauf geachtet werden, dass ein echtes Schwarz eingesetzt wird und die Strichstärken stimmen. Diese Details sollten bereits im Entwurfsstadium berücksichtigt werden, um spätere Nachbesserungen zu vermeiden bzw. zu reduzieren.

9.2 Nicht fertiggestellte Arbeiten

9.2.1 Visualisierungen

Die finalen Pattern-Icons wurden für alle in der Übersicht in Kapitel 3 aufgezeigten Patterns erstellt und gemäss den Anforderungen aus dem Visualisierungskonzept fertiggestellt und exportiert.

Da für die Patterns und Pattern-Kategorien im Laufe des Entwurfsprozesses mehrere Visualisierungsentwürfe entstanden sind, gibt es nicht nur die in Kapitel 3 abgebildeten Grafiken. Für jedes Pattern und für jede Pattern-Kategorie sind neben den finalen Icons auch die Alternativicons bestimmt worden, sozusagen die «Runner-Up». Aus zeitlichen Gründen hat es das Projektteam nicht geschafft, die Alternativgrafiken nachzubearbeiten, um deren korrekte Funktionsweise in anderen Programmen wie PowerPoint sicherzustellen. Insbesondere die im Absatz 4.6.3 «Konturen in Flächen umwandeln» und Absatz 4.6.2 «Skalierung» erwähnten Themen müssten in einer Folgearbeit für die Alternativgrafiken in Betracht gezogen werden.

9.2.2 Web-Recherche und Dokumentation

Die im Kapitel 8 dokumentierten Rechercheergebnisse zu den IRP und deren Vorkommen in öffentlichen Web-APIs basieren auf einer separaten vom Projektteam ausgearbeiteten Dokumentation. Letztere bildet die Basis für die Kurzzusammenfassungen im Kapitel 8. Das Projektteam hat aus Zeitgründen die folgenden zwei Patterns nicht in den im Absatz 8.1 aufgelisteten Web-APIs recherchiert, getestet und dokumentiert:

- Semantic Patterns
- Parameter Types

Dies ist ordnungshalber hier aufgeführt. Es ist zu einem späteren Zeitpunkt mit dem Auftraggeber zu prüfen, ob diese Recherchen noch durchgeführt werden sollen. Die *Foundation* Patterns sind oben nicht aufgeführt, da diese Patterns für öffentlichen APIs nicht von Bedeutung sind.

Kapitel 10

Schlussfolgerung

10.1 Zusammenfassung

Die Studienarbeit baut auf die IRP Patternsprache auf, welche aktuell von einem Autorenteam der HSR zusammen mit Kooperationspartnern erarbeitet wird. Zu Beginn lagen detaillierte Ausformulieren für einige elementare IRP vor. Das Projektteam hat aufgrund dieser Beschreibungen bereits in der zweiten Woche begonnen, Visualisierungsentwürfe auszuarbeiten. Die Entwürfe wurden in einem iterativen Prozess jeweils, wenn nötig, angepasst und verbessert. Zeitgleich erstellte das Projektteam den ersten Entwurf des Visualisierungskonzeptes, um eine Grundlage für den Designprozess zu haben, die Einflüsse und User Stories festzuhalten sowie Dritten die Grundideen zu den Visualisierungen aufzuzeigen.

Der iterative Prozess war ein wichtiger Faktor für den Stand der jetzigen Grafiken. Die erarbeiteten Visualisierungen durchliefen mehrere Testverfahren und damit zusammenhängende Anpassungen, bis diese als final bezeichnet werden konnten.

Das Projektteam beschäftigte sich intensiv mit Web-APIs und deren Analyse zu den IRP. Die Recherchetätigkeiten inklusive Dokumentation der Ergebnisse machten ungefähr 30% der Studienarbeit aus. Die aus der Recherche gewonnen Erkenntnisse wurden dem IRP-Autorenteam jeweils zur Verfügung gestellt.

Die 10.1 zeigt die am Anfang der Studienarbeit gesetzten Ziele und stellt sie dem Endprodukt gegenüber:

Zieldefinition	**Stand Ende Studienarbeit**
Web Recherche: **Patterns in populären Web APIs auffinden und die Verwendung dokumentieren.**	Die Known Uses der Patterns sind in sechs öffentlichen APIs analysiert und verifiziert worden. Die untersuchten APIs sind: • Facebook Graph API (v2.9) • Twitter REST API • YouTube Data API (v3) • GitHub v3 API

	• Google Calendar API • Stripe API Das Projektteam hat die daraus gewonnenen Erkenntnisse dokumentiert erstellte für das IRP-Autorenteam ein Nachschlagwerk über die Known Uses.
Visualisierung-Design: **Icons und Solutions Sketches zur Illustration der Patterns à la Enterprise Integration Patterns und Cloud Computing Patterns konzipieren und aufgrund von Zwischenfeedback inkrementell verbessern und erweitern.**	Die Grafiken wurden mittels Adobe Illustrator, einem vektorbasierten Zeichnungstool, erstellt. Eine Übersicht über die erarbeiteten Grafiken kann dem Kapitel 6 entnommen werden. Zusätzlich wurde ein Visualisierungskonzept als Grundlage für den Designprozess ausgearbeitet, welches als Leitfaden zur einheitlichen Gestaltung der Visualisierungen diente. Alle finalen Visualisierungen haben mehrere Reviews durchlaufen und konnten inkrementell verbessert werden. Das Visualisierungskonzept kann künftig bei der Entwicklung von Visualisierungen in ähnlichen Projekten zur Unterstützung konsultiert werden.
Prototypen-Implementierung: **Beispielprogramme erstellen, die die Verwendung der Patterns (und damit auch der Icons und Solution Sketches) zeigen.**	In der 4 Woche hat das Projektteam mit einer simplen Webapplikation das erste Pattern umgesetzt. Aufgrund Priorisierung anderer Aufgaben, stand die Implementierungstätigkeit nicht im Fokus und wurde zurückgestellt.

Abbildung 10.1: Überblick umgesetzte Ziele

10.2 Ausblick

Die in dieser Studienarbeit erstellten Visualisierungen können für die Patternsprache eingesetzt werden. Die Grafiken werden in der Sommerpause 2017 weiteren Tests unterzogen und je nach Output, wird die eine oder andere Grafik zu einem späteren Zeitpunkt angepasst werden. Die erarbeitete Übersicht der Web-APIs mit den Rechercheergebnissen kann vom IRP-Autorenteam für Known Uses konsultiert werden.

Kapitel 11

Literaturverzeichnis

1. Allamaraju S (2010) Restful web services cookbook: solutions for improving scalability and simplicity. O'Reilly Media, Inc.

2. Ambler SW (2002) Agile modeling: effective practices for eXtreme programming and the unified process. J. Wiley, New York

3. Buschmann F, Meunier R, Rohnert H et al (1996) Pattern-Oriented Software Architecture Volume 1: A System of Patterns. John Wiley & Sons Inc

4. Hohpe G, Woolf B (2004) Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional

5. Moody D (2009) The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans Softw Eng 35:756–779.

6. Richter M, Flückiger MD (2013) Usability Engineering kompakt: benutzbare Produkte gezielt entwickeln, 3. Aufl. Springer Vieweg, Berlin

7. Spichale K (2016) API-Design - Praxishandbuch für Java- und Webservice-Entwickler, 1. Auflage 2017. dpunkt.verlag GmbH

8. Spillner A (2014) Praxiswissen Softwaretest - Testmanagement: Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard, 4., überarb. und erw. Aufl. dpunkt.-Verl, Heidelberg

9. Sturgeon P (2016) Build APIs You Won't Hate.

10. Wallmüller E (2001) Software-Qualitätsmanagement in der Praxis: Software-Qualität durch Führung und Verbesserung von Software-Prozessen, 2., völlig überarb. Aufl. Hanser, München

11. Zimmermann O, Stocker M, Lübke D, Zdun U (2017) Interface Representation Patterns — Patterns for Crafting and Calling Message-Based Remote APIs.

12. (2017) Bauhaus. Wikipedia

13. Enterprise Integration Patterns - Introduction to Integration Styles. http://www.enterpriseintegrationpatterns.com/patterns/messaging/IntegrationStylesIntro.html. Zugegriffen: 26. Mai 2017

14. Cloud Computing Patterns. http://www.cloudcomputingpatterns.org/. Zugegriffen: 26. Mai 2017

15. Network Topology Icons - Doing Business With Cisco. http://www.cisco.com/c/en/us/about/brand-center/network-topology-icons.html. Zugegriffen: 26. Mai 2017

16. Material icons - Material Design. https://material.io/icons/. Zugegriffen: 26. Mai 2017

17. Components · Bootstrap. http://getbootstrap.com/components/. Zugegriffen: 26. Mai 2017

18. Font Awesome Icons. http://fontawesome.io/icons/. Zugegriffen: 26. Mai 2017

19. Design Guidelines — The way products are built. http://designguidelines.co/. Zugegriffen: 26. Mai 2017

20. Early Access Program | GitHub Developer Guide. https://developer.github.com/early-access/. Zugegriffen: 26. Mai 2017

21. Stripe - API Upgrades. https://stripe.com/docs/upgrades. Zugegriffen: 26. Mai 2017

Kapitel 12

Abbildungsverzeichnis

Kapitel 13

Tabellenverzeichnis