Technology
Arts Sciences
TH Köln

Moderne lose gekoppelte Softwarearchitekturen: Wie die Hochschullehre komplexe Systeme „hands-on“ vermitteln kann

MASTER THESIS

ausgearbeitet von

Philipp Felix SCHMEIER

zur Erreichung des Grades

MASTER OF SCIENCE (M.SC.)

an der

TECHNISCHEN HOCHSCHULE KÖLN
CAMPUS GUMMERSBACH

im Studiengang

INFORMATIK / COMPUTER SCIENCE

Erstprüfer:	Prof. Dr. Stefan BENTE Technischen Hochschule Köln
Zweitprüfer:	Dr. Christian KOHLS Technischen Hochschule Köln

Gummersbach, Februar 2022

Kontaktdetails: Philipp Felix SCHMEIER
Kaiserstr. 71
51643 Gummersbach
philipp.schmeier@gmail.com

Prof. Dr. Stefan BENTE
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
stefan.bente@th-koeln.de

Prof. Dr. Christian KOHLS
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
christian.kohls@th-koeln.de

Abstract

Due to ever growing software landscapes, the importance of software architects increases, who have to design software according to different criteria. This thesis describes an approach to enable students to get to know loosely coupled systems with their »corners and edges«. The thesis therefore starts with the definition of important qualifications of software architects. The qualifications are to be used as a basis for the construction of a simulation of loosely coupled systems. The determination of the qualifications happens by means of three different pillars:

(1) What qualifications are currently sought on the labor market?

(2) What is the existing apprenticeship in this field in Germany?

(3) How do renowned experts define and assess qualifications?

Based on this, a simulation is described which was carried out at the TH Köln in a cross-module project with approx. 50 students. In the simulation, an attempt was made to give the students an understanding of the qualifications determined through their own experience. The simulation is based on a complex system landscape that the students themselves helped to design and use.

Finally, a summary is drawn that describes the results of the simulation and defines indications for further iterations. In advance, it can be said that the simulation is considered a great success by both students and teachers. Especially the feedback of the students regarding enjoyment, learning success and quality of their own performance was positive.

Zusammenfassung

Durch immer größer werdende Softwarelandschaften steigt die Bedeutung von Softwarearchitekt:innen, welche Software nach unterschiedlichsten Kriterien gestalten müssen. Diese Thesis beschreibt einen Ansatz, wie es Studierenden ermöglicht werden soll, lose gekoppelte Systeme mit deren »Ecken und Kanten« kennenzulernen. Die Arbeit beginnt daher mit der Definition wichtiger Qualifikationen von Softwarearchitekt:innen. Diese Qualifikationen sollen für den Aufbau einer Simulation lose gekoppelter Systeme zugrundegelegt werden. Die Ermittlung der Qualifikationen geschieht mittels drei verschiedener Säulen:

(1) Welche Qualifikationen werden auf dem Arbeitsmarkt aktuell gesucht?

(2) Wie sieht die bestehende Lehre in diesem Bereich in Deutschland aus?

(3) Wie definieren und beurteilen namhafte Experten geeignete Qualifikationen?

Darauf aufbauend wird eine Simulation beschrieben, welche an der TH Köln in einem modulübergreifenden Projekt mit ca. 50 Studierenden durchgeführt wurde. In der Simulation wurde dabei versucht, den Studierenden die ermittelten Qualifikationen durch eigene Erfahrungen näherzubringen. Die Simulation basiert auf einer von den Studierenden selbst mitgestalteten und genutzten, komplexen Systemlandschaft.

Abschließend wird ein Resümee gezogen, welches die Ergebnisse der Simulation beschreibt und Hinweise für weitere Iterationen definiert. Vorausgreifend kann gesagt werden, dass die Simulation sowohl von Studierenden wie Lehrenden als großer Erfolg angesehen wird. Besonders die Resonanz der Studierenden bzgl. Freude, Lernerfolg und Qualität der eigenen Leistung ist positiv ausgefallen.

Vorwort

Vor sich halten Sie die Masterarbeit mit dem Titel „Moderne lose gekoppelte Softwarearchitekturen: Wie die Hochschullehre komplexe Systeme „hands-on“ vermitteln kann“ von Philipp Schmeier. Die Arbeit habe ich im Zeitraum von Anfang August 2021 bis Mitte Februar 2022 verfasst. Ziel dieser Arbeit ist die Ausarbeitung und Klärung der Frage, welche Eigenschaften und Fähigkeiten gute Softwarearchitekt:innen haben sollten und wie dies praktisch in der Lehre vermittelt werden kann.

Speziell bedanken möchte ich mich bei meinem Betreuer Herrn Prof. Dr. Stefan Bente, welcher mir auch in schwierigen Zeiten ermöglicht hat mich komplett auf meine Arbeit zu konzentrieren. Gerne möchte ich mich auch bei allen befragten Experten nochmals für die aufgebrachte Zeit und Begeisterung für die Sache bedanken. Hervorheben möchte ich weiterhin die Tatkraft vieler Studierender, welche an der Simulation teilgenommen und somit diese ermöglicht haben. Ihr Engagement hat in vielen Bereichen die Simulation vorangetrieben.

Auch gilt mein Dank Familie und Freunden, welche mich auch durch kompliziertere Zeiten begleitet haben und mir die geringere Aufmerksamkeit ihnen gegenüber nicht vergolten haben. Besonderer Dank gilt dabei allen Korrekturleser:innen, ohne die diese Arbeit ein wohl etwas anderes Leseerlebnis beschert hätte.

Ausdrücklich nicht bedanken möchte ich mich bei dem Coronavirus, welches vieles nur schwieriger gemacht hat.

Ich wünsche viel Vergnügen beim Lesen.

Philipp Schmeier

Gummersbach, den 15. Februar 2022

Inhaltsverzeichnis

Abkürzungsverzeichnis

ADR	**A**rchitecture **D**ecision **R**ecords
DDD	**D**omain **D**riven **D**esign
FAE	**F**achspezifischer **A**rchitektur**e**ntwurf
IP	**I**nformatik **P**rojekt
iSAQB	**I**nternational **S**oftware **A**rchitecture **Q**ualification **B**oard
MSA	**M**icro**s**ervice **A**rchitectures
MVP	**M**inimum **V**iable **P**roduct
PL	**P**roject **L**aunch
PO	**P**roduct **O**wner
UML	**U**nified **M**odeling **L**anguage

Kapitel 1

Einführung

Softwarelandschaften und Systeme wachsen mit den Anforderungen der Kunden und Nutzer und werden immer größer, wie Beispiele großer vernetzter Strukturen bei Netflix oder Amazon [1] zeigen. Für einen reibungslosen Ablauf ist daher eine frühzeitige Planung der Abläufe nötig. Die Unternehmen müssen besonders bei solchen Anforderungen hochperformante und skalierbare Systeme bereitstellen. Diese Problematik kann nur bedingt alleine von Entwicklern gelöst werden. Das ist der Tatsache geschuldet, dass nicht jeder Entwickler die gesamte Systemlandschaft und deren Zusammenhänge kennen und während der eigenen Entwicklungsarbeit mit betrachten kann. [C.2.7 Wolff] Die Rolle von Softwarearchitekt:innen festigt damit ihre Bedeutung, eben diese Zusammenhänge als höhere Instanz zu planen und zu überwachen.

Diese großen Systeme müssen lose gekoppelt sein, damit Änderungen oder Fehler einzelner Bereiche nicht den gesamten Betriebsablauf eines Unternehmens stören können. In der Lehre besteht das Problem, dass allein die Größe solcher Systemlandschaften eine Vermittlung erschweren. Durch zeitliche und daraus resultierender inhaltlicher Begrenzungen von einzelnen Modulen können diese Systemlandschaften nicht wirkungsvoll bzw. realitätsnah dargestellt werden. Es ist somit nötig, Abstriche zu machen, um Studierenden die wichtigsten Aspekte und Hürden lose gekoppelter Systeme erlebbar zu machen.

Diese Problematik spiegelt auch den Hintergrund für diese Arbeit wider, deren konkrete Motivation in Kapitel 1.1 dargelegt wird. Um die Motivation in ein abgrenzbares Ziel umzusetzen, wurden zwei Forschungsfragen entwickelt. Diese werden zusammen mit weiteren Details im Kapitel 1.2 Zielsetzung beschrieben. Abgeschlossen wird die Einführung mit der Vorgehensweise (siehe Kapitel 1.3), welche eine Übersicht über die initiale Planung für Thesis, Recherche und betreutes Projekt darstellt.

1.1 Motivation

Die Thesis wird an der TH Köln am Campus Gummersbach erarbeitet und von Herrn Prof. Dr. Stefan Bente betreut. Er betreut außerdem mit seinen Mitarbeiter:innen des ArchiLabs[1] verschiedene Module rund um die Thematik der Softwarearchitektur.

Im Rahmen der Corona-Pandemie wurde die gesamte Lehre der letzten Semester online gestaltet. Darunter fallen auch die beiden Module Softwaretechnik I+II, welche im Informatik Bachelorstudiengang am Campus Gummersbach gelehrt werden. Bei der Umstellung auf 100 %-ige Onlinelehre wurde die Vorlesung komplett umgestellt und mit dem Divekit[2] ein starker Fokus auf individuelle programmierlastige Abgaben gelegt.

Durch die so zusätzlich entstandenen Inhalte wurde der Vorlesungsrahmen leicht umgestaltet, wodurch die im Modul Softwaretechnik II behandelten Konzepte zu Microservices nicht so weit vertieft werden konnten wie zuvor. Dies betraf besonders Konzepte wie lose Kopplung, asynchrone Kommunikation und mehr. Mit dieser Problematik entstand die Grundidee, das Informatikprojekt[3] zu nutzen, um diese Thematiken mit viel Zeit zu vertiefen.

In diesem Kontext kam die Fragestellung auf, wie die Inhalte rund um lose Kopplung für ein entsprechendes Informatikprojekt gestaltet werden können. Den Studierenden sollen im Projekt die Lehrinhalte durch realitätsbezogene, praktische Erfahrungen vermitteln werden. Eine Herausforderung, welche in der Lehre bisher wenig erforscht und beschrieben ist und daher den Ursprung dieser Arbeit darstellt.

1.2 Zielsetzung

Das Ziel der Arbeit ist es, die fehlende Forschungsdichte im Gebiet der Lehre von Softwarearchitekt:innen mit Fokus auf lose gekoppelte Systeme durch eigene Grundlagenrecherchen und Befragungen zu erweitern. Die Ergebnisse dieser Recherche für die Hochschullehre zu bewerten und im Rahmen der Simulation mit den Studierenden einfließen zu lassen. Für eine Eingrenzung der Arbeit wurden zwei Forschungsfragen entwickelt, welche in dieser Thesis beantwortet werden sollen:

(1) Welche Erfahrungen müssen Lernende in der Softwarearchitektur machen, um die Notwendigkeit von modernen und lose gekoppelten Architekturen verstehen und umsetzen zu können?

[1] ArchiLab: Das Software Architecture Laboratory ist Teil des CIDE (Cologne Institute for Digital Ecosystems) und beschäftigt sich mit der Lehre und Forschung von modernen Softwarearchitekturen digitaler Ökosysteme.

[2] Divekit ist eine durch das ArchiLab entwickelte Individualisierungssoftware, welche es ermöglicht mehrere Git-Repositories (Arbeits- und Test-Repositories) je Studierendem zu erstellen.

[3] Ein variables Projekt, welches nach Plan ein Semester nach Softwaretechnik II durchgeführt werden soll. Mehr Informationen in Kapitel 3.4.1

(2) Wie muss eine Simulation gestaltet werden, um die festgestellten Kompetenzen erfolgreich vermitteln zu können?

Die Arbeit beginnt mit der Klärung eines wichtigen Aspekts zur ersten Fragestellung. Welche zu diesem Zweck in Verbindung gebracht wird mit der Frage: Was macht gute Softwarearchitekt:innen aus? Dies wird in Kapitel 2, im Kontext von lose gekoppelten Architekturen definiert.

Nach Klärung dieser Frage wird in Kapitel 3 die Simulation beschrieben, welche aus den gesammelten Erkenntnissen und bestehenden Lehrkonzepten geplant und durchgeführt wurde. Zusammen mit Kapitel 4, mit der Auswertung der Ergebnisse, wird somit die zweite Forschungsfrage beantwortet.

Weitere indirekte Ziele, welche sich aus dem Kontext der Simulation ableiten, sind allen beteiligten Modulen entsprechend ihrer Lernziele passende Aufgaben und Herausforderungen zu bieten. Diese werden jedoch nicht im Detail in dieser Arbeit beschrieben.

1.3 Vorgehensweise

Zur Erreichung des gesteckten Ziels wurde eine Planung über das Semester hinweg mit dem Start der Arbeit erstellt. Dies war erforderlich, da sich die Vorgehensweise und der gesamte Ablauf stark an das Wintersemester 2021/2022 gebunden hat. Nur so bestand die Möglichkeit, den Zeitraum der Lehrveranstaltungen zu begleiten, welche Teil der Simulation sein sollten. Der Vorlesungszeitraum des Semesters lag zwischen dem 04. Oktober 2021 und 04. Februar 2022. Mit dem Start der Arbeit am 10. August 2021 wurde ein Planungs- und Vorbereitungszeitraum gegeben, welcher genutzt wurde, um die Modulabläufe zu planen. Im Falle der beteiligten Projekte inkludierte dies auch eine Bewerbungsphase für alle Bewerber:innen. Abbildung 1.1 zeigt die initiale Planung nach Kalenderwochen.

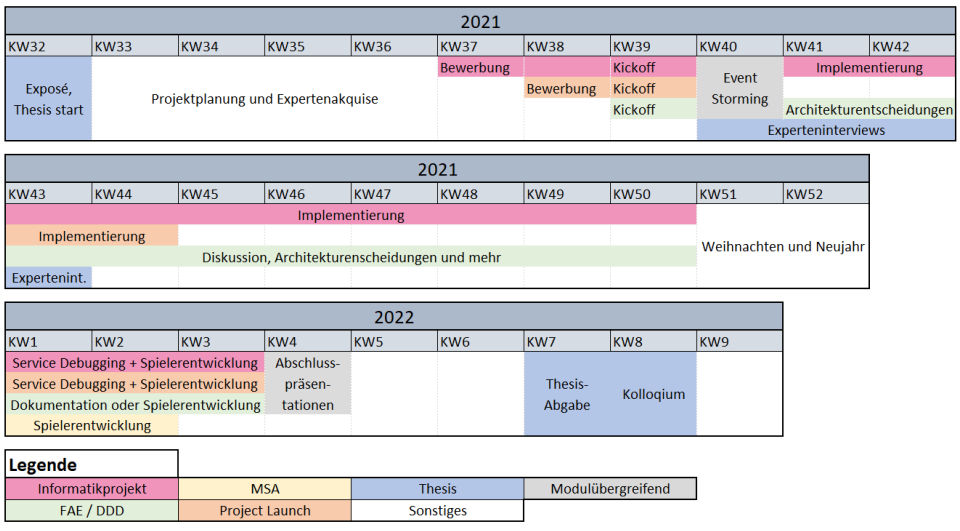


ABBILDUNG 1.1: Initiale Grobplanung des Projektes und der Masterarbeit

Neben der zeitlichen Projektplanung fiel in diese Zeit auch die Akquirierung der Experten und die Festlegung der Interviews. Es wurde versucht, den Zeitraum der Interviews über den gesamten Monat Oktober zu legen. Dies ermöglichte einerseits eine einfachere Terminfindung mit den teilweise stark eingebundenen Interviewpartnern und andererseits ermöglichte es Erkenntnisse und Thesen aus den ersten Interviews in den folgenden Interviews zu diskutieren. Nach Durchführung und Aufbereitung der Interviews folgte eine intensivere Projektphase, da zu diesem Zeitpunkt alle Teams seit einigen Wochen mit der Entwicklung ihrer Services beschäftigt waren und erste Problematiken und Herausforderungen zu bewältigen hatten.

In diesem Zeitraum wurden in einer ersten Umfrage alle zu diesem Zeitpunkt aktiven Projektbeteiligten befragt. Diese Umfrage wurde in erweiterter Form am Ende des Projektes ein weiteres Mal durchgeführt.

Abschließen sollte das Projekt im Januar 2022 mit einem Hackathon und einer Abschlusspräsentation der Studierenden. Der Plan musste teilweise angepasst werden, wodurch der Hackathon in veränderter Form durchgeführt wurde. Mehr Informationen dazu später in Kapitel 4.2.

Nach Abschluss des Projektes wurde die restliche Zeit zur Bewertung der Ergebnisse und zur Einarbeitung in die Thesis eingeplant.

Kapitel 2

Qualifikationen von Softwarearchitekt:innen

In diesem Kapitel soll geklärt werden, welche Qualifikationen zur Ausfüllung der Architekt:innen-Rolle benötigt werden und welche dabei besonders wichtig sind. Als Vorbereitung für die Simulation und Lehre zukünftiger Softwarearchitekt:innen ist die Untersuchung und Feststellung der grundlegenden Qualifikationen wichtig. Hard- wie Soft Skills sowie ihre Gewichtung stehen dabei im Fokus. Sind die Qualifikationen definiert, können sie als Zielkompetenzen direkt bei der Gestaltung der Simulation berücksichtigt werden. Softwarearchitekt:innen müssen eine Reihe von Aufgaben meistern, welche je nach Unternehmen sehr unterschiedlich ausfallen können. [2] Zusätzlich dazu ist die Architekt:innenrolle besonders in kleineren Unternehmen nicht der einzige Aufgabenbereich, den Softwarearchitekt:innen dort abdecken. [C.2.7 Deterling]

Die Qualifikationen sind über drei verschiedene Wege ermittelt worden, (1) eine Analyse des Arbeitsmarktes anhand aktueller Stellenausschreibungen für Softwarearchitekt:innen oder vergleichbarer Stellenangebote. Des Weiteren (2) durch eine Analyse aktueller Studien- und Lehrangebote verschiedener Hochschulen und zuletzt (3) durch die Befragung bekannter Experten, also einem Ausschnitt aus der Community.

In den folgenden Unterkapiteln werden die Erhebungsmethoden aller drei Säulen erläutert und die Ergebnisse analysiert, bevor diese gegenübergestellt werden.

2.1 Der Arbeitsmarkt

Der Arbeitsmarkt und die benötigten Fachkräftekontingente werden in den Stellenausschreibungen widergespiegelt. Die Ausschreibungen helfen dabei, die nötigen Wunschqualifikationen für solche Stellen zu beschreiben und eine Übersicht dafür zu bekommen, welche Standards und Expertisen aktuell besonders benötigt werden.

Die Beschreibungen gehen dabei meist sogar über das Ziel hinaus, da die Profile so umfassend und detailliert beschrieben werden, dass es kaum eine

Person mit sämtlichen passenden Qualifikationen geben kann. [3] Da in den Stellenanzeigen somit meist keine Wunschqualifikationen ausgelassen werden, ist dies für eine Analyse des Inhalts von Vorteil.

2.1.1 Erhebung

Es wurden insgesamt 59 Stellenanzeigen deutschlandweit für Softwarearchitekt:innen gesammelt. Hierbei wurde darauf geachtet, dass eine Variation in den Bereichen Anbieterportale, Branchen und Unternehmen sichergestellt ist. Die Stellenanzeigen stammen von 49 unterschiedlichen Unternehmen aus über 20 verschieden Branchen und wurden dabei aus acht unterschiedlichen Job-Portalen sowie direkt von Unternehmensanzeigen bezogen. Referenzen auf alle analysierten Stellenanzeigen sind in Anhang A zu finden.

2.1.2 Analyse

Für die Analyse wurden aus allen Stellenbeschreibungen die relevanten Teile extrahiert. Dies bedeutet, dass ausschließlich die Aufgaben und Wunschprofile zur Analyse genutzt wurden. Weitere Inhalte der Stellenanzeigen, wie Informationen zu den Unternehmen sowie deren Angebote wurden nicht berücksichtigt. Begonnen wurde mit einer Häufigkeitsanalyse verwendeter Wörter. Für die Häufigkeitsanalyse wurden verschiedene Schritte durchgeführt:

1. Wörter isolieren
2. Entfernung von Satzzeichen u. ä.
3. Stemming[1] aller Wörter
4. Wortstreichungen mittels Stoppwortliste
5. Häufigkeit der Wörter zählen

In den Stellenanzeigen findet sich eine breite Menge an angeforderten Technologien. Durch die Variation an Technologien gibt es nur eine einzige, welche die Top 10 der häufigsten Begriffe erreicht hat: Java. Die meisten weiteren Begriffe der Top 10, wie in Abbildung 2.1 zu sehen, sind dabei allgemeinerer Art.

[1] Zurückführung eines Wortes auf den Wortstamm

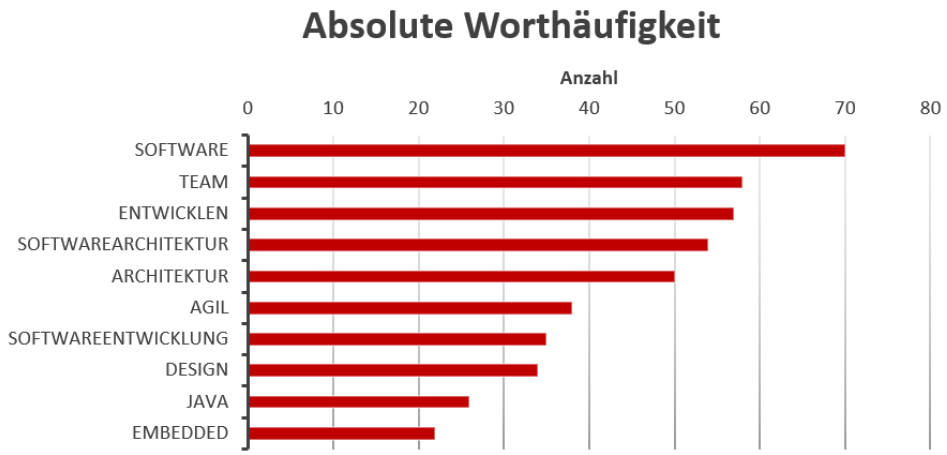


ABBILDUNG 2.1: Die häufigsten Begriffe aller ausgewerteten Stellenanzeigen

Neben zu erwartenden und für die weitere Analyse weniger relevanten Wörtern wie: Software, Entwickeln, Softwarearchitektur, etc. finden sich in den häufigsten 10 Wörtern auch konkrete Thematiken wie Team, agil oder auch Java wieder. Besonders die Arbeit im Team findet dabei in fast 80 % aller untersuchten Anzeigen Erwähnung (siehe Abbildung 2.2).

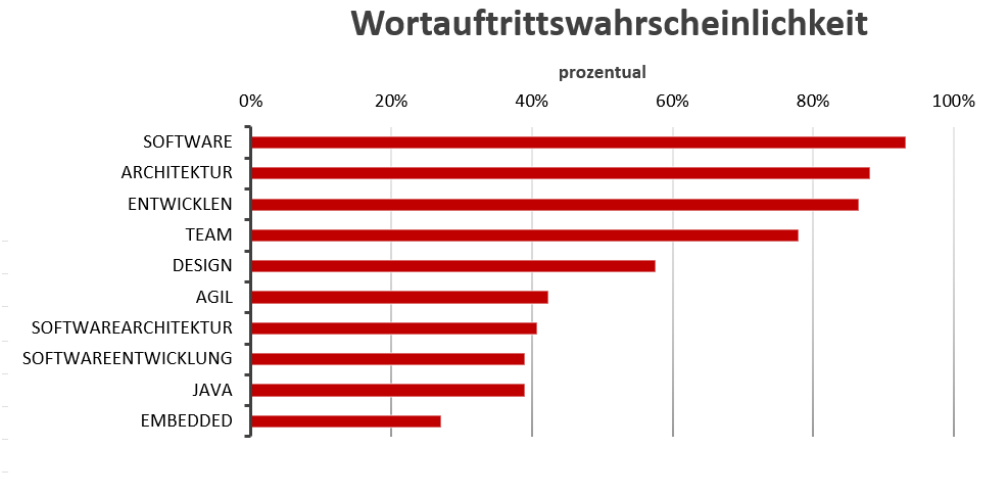


ABBILDUNG 2.2: Prozentuale Wahrscheinlichkeit, dass ein Begriff in einer Stellenausschreibung vorkommt

Die Arbeit im Team und übergreifende Prozesse wie die agile Organisation nehmen damit einen größeren Stellenwert ein als einzelne Technologien. Neben Java ist die Bekanntheit fünf weiterer Programmiersprachen und Frameworks in den Stellenanzeigen relevant, wie Abbildung 2.3 darstellt. Java wurde den Studierenden bereits in vorangegangenen Modulen gelehrt und somit in der Simulation (siehe Kapitel 3.3) von ihnen häufig gewählt.

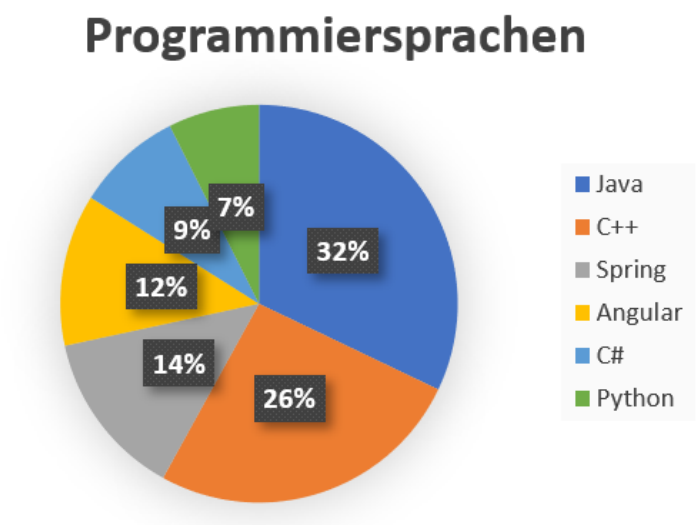


ABBILDUNG 2.3: Verteilung der häufigsten Programmiersprachen und Frameworks in Stellenanzeigen

Weitere nennenswerte Begrifflichkeiten (siehe Abbildung 2.4) wie UML, Microservices oder Kubernetes bilden dabei das hintere Mittelfeld der mehrmals genannten Thematiken ab. UML liegt dabei mit insgesamt 12 Nennungen sogar noch vor Spring mit 11 (siehe Abbildungen 2.3, 2.4). Dies verdeutlicht das breite Spektrum der Anforderungen des Arbeitsmarktes. Für die Simulation bedeutet dies, dass es gut ist Grundkenntnisse verschiedener Technologien kennenzulernen sowie diese in einem teamfokussierten und agilen Prozess zu lehren und es weniger wichtig ist bspw. Java-Experten auszubilden.

Anzahl	Begriff
12	UML
11	Cloud
11	Microservice
9	Linux
9	Scrum
7	Kubernetes
5	git
5	SQL
3	TDD
3	DevOps

ABBILDUNG 2.4: Nennenswerte Begriffe in Stellenanzeigen

Die Essenz der Analyse ist, dass es keine einheitlichen Wunschqualifikationen gibt, welche man eindeutig Softwarearchitekt:innen zuordnen kann.

Wichtige Merkmale liegen in der Fachkompetenz und Vertrautheit mit branchennahen Abläufen, so müssen bspw. Architekt:innen bei der »Volkswagen Infotainment GmbH« bereits Berufserfahrung in der Automobilindustrie vorweisen. Beim Personaldienstleister »Professional Scientists GmbH & Co. KG« werden hingegen keine branchenspezifischen Vorkenntnisse verlangt. Dieser legt mehr Wert auf generelle Berufserfahrung mit verschiedenen Programmiersprachen und agilen Methodiken. [Vgl. Anhang A]

2.2 Die Hochschullehre

Die Ausbildung von Softwarearchitekt:innen, Engineers und weiteren informatikbezogenen Berufen ist fester Bestandteil vieler Informatikstudiengänge. Besonders auf Analyse- und Projektmanagement-Fähigkeiten, sowie Entwurfs- und Realisierungs-Kompetenzen wird dabei großer Wert gelegt. Empfehlungen, wie sie die Gesellschaft für Informatik für Bachelor- und Master Studienprogramme gibt. [4]

Die Forschung und Lehre mit speziellem Fokus auf der Vermittlung von größeren komplexen Systemen, wie lose gekoppelten Microservice-Landschaften, bietet nur sehr vereinzelt konkrete Lösungsstrategien.

An der »University of São Paulo« (USP) wurde im Rahme des Kurses »Laboratory of Complex Computational Systems« ein Projekt mit einer großen Gruppe[2] von Studierenden durchgeführt, in welchem eine Microservicelandschaft entwickelt wurde. [5] Das aus dem Projekt entstandene Paper geht dabei weniger auf die Ziele rund um die Vermittlung ein, als auf den konkreten Ablauf und die Ergebnisse. Weitere Lehrbeispiele in publizierter Form konnten während der Recherchen nicht gefunden werden.

Daher wurde auch hier eine eigene Analyse erstellt, um die angebotenen Lehrmodule rund um die Softwarearchitektur vergleichen zu können. Diese Analyse wurde äquivalent dem Vorgehen der Analyse der Stellenanzeigen aufgebaut.

2.2.1 Erhebung

Als Grundlage für die Analyse wurden die Modulbeschreibungen von 57 Modulen von insgesamt 15 verschiedenen deutschen Hochschulen ausgewertet. Die Auswahl der Hochschulen orientierte sich am Hochschulranking für Informatik durch die »Initiative für transparente Studienförderung gemeinnützige UG« (MyStipendium.de). Deren Bewertung basiert auf den folgenden Internationalen und Nationalen Hochschulrankings [6]:

- QS World University Ranking
- Shanghai Ranking

[2] 4-mal als zweiwöchiger Kurs mit insgesamt 65 Studierenden und einmalig einsemestrig mit 18 Studierenden

- CHE Hochschulranking
- Ranking der WirtschaftsWoche

2.2.2 Analyse

Die Analyse wurde nach dem gleichen Verfahren, wie in Kapitel 2.1.2 beschrieben, durchgeführt. Äquivalent zur Stellenanzeigenanalyse wurden rein organisatorische Informationen nicht mit berücksichtigt. Der Fokus wurde auf Beschreibungen und Lerninhalte gelegt. Anders als bei den Jobtiteln der Stellenanzeigen (Softwarearchitekt:in) wurden unterschiedliche Modulbezeichnungen durchsucht. Von »Softwarearchitektur« über »Softwaretechnik« bis hin zu »Software Engineering«. Hierfür wurden sowohl Master- wie Bachelormodule einbezogen. Die folgende Abbildung 2.5 zeigt die Top 10 der häufigsten Begriffe in Modulbeschreibungen.

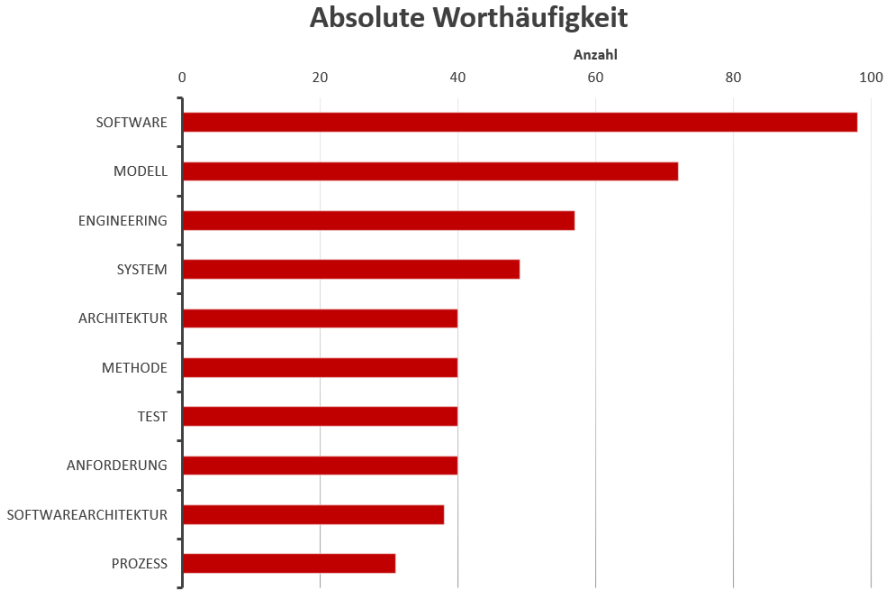


ABBILDUNG 2.5: Die häufigsten Begriffe aller ausgewerteten Module

Die ermittelten Begriffe zeigen die stärkere Abstraktion der Hochschullehre. Die Beschreibungen erläutern dabei nicht die konkreten Technologien, sondern die dahinterliegenden Konzepte. Ein Fokus auf die Top-Ergebnisse ist daher nicht zielführend.

Die zugrundeliegenden Technologien sind für die Professor:innen und Ersteller:innen der Modulbeschreibungen eher zweitrangig und wurden daher oftmals nicht mit darin aufgenommen. Die spezifische Herangehensweise und Technologiewahl ist dann von Lehrendem zu Lehrendem unterschiedlich.

Eine der wenigen der genannten Technologien oder Konzepte ist UML, welches mit 30 Nennungen sogar den 12. Platz einnimmt (siehe Abbildung 2.6).

Die Programmiersprachen sind hauptsächlich durch Java mit nur fünf Nennungen vertreten.

Rang	Wort	Anzahl	Auftrittwahrscheinlichkeit
1	Software	98	93,0 %
2	Modell	72	80,7 %
3	Engineering	57	75,4 %
4	System	49	71,9 %
5	Architektur	40	57,9 %
6	Methode	40	56,1 %
7	Test	40	54,4 %
8	Anforderung	40	54,4 %
9	Softwarearchitektur	38	50,9 %
10	Analyse	31	50,9 %
11	Prozess	31	15,8 %
12	UML	30	29,8 %
13	Konzept	29	26,3 %
14	Technik	28	49,1 %
15	Modellierung	26	47,4 %
16	Design	23	29,8 %
17	Objektorientiert	22	26,3 %
18	Softwaresystem	22	26,3 %
19	Entwicklung	20	33,3 %
20	Entwurf	19	40,4 %
21	Softwareentwicklung	19	26,3 %
22	Code	18	21,1 %

ABBILDUNG 2.6: Liste der häufigsten Begriffe aller augewerteten Module

Die erweiterte Liste der analysierten Wörter in Abbildung 2.6 zeigt (ausgenommen UML), dass auch in einer erweiterten Betrachtung konkrete Technologien oder Konzepte in den Hintergrund treten. Hochschulen halten es durch flexible und abstraktere Beschreibungen allgemein. Somit haben sie die Möglichkeit, ihre Lehre variabel zu gestalten. Direkte Folgerungen für die Simulation, bauen daher stärker auf den beiden anderen Analysen auf.

2.3 Expertenwissen

Die alleinige Sicht auf die bisherige Lehre und den Arbeitsmarkt reicht nicht aus. Betrachtet man nur die Lehre, würden Fortschritte in Technik und Innovationen nur langsam mit einfließen. Von der Entstehung einer Technologie bis hin zur erfolgreichen Lehre im Hochschulkontext stehen zumindest in herkömmlichen Modulen einige Hindernisse.

Wird die Sicht auf den Arbeitsmarkt erweitert oder sogar alleinig darauf fokussiert, entsteht eine weitere Gefahr. Stellenanzeigen müssen mit wenigen Worten ein komplexes Themenfeld abdecken. Durch diese Tatsache bedingt werden meist verschiedene Buzzwörter genutzt. Dabei kann es vom Kontext abhängen, welche Details sich dahinter verbergen. Eine erste Wissensübertragung und »stille Post Effekt« tritt bereits bei der Erstellung der Anzeigen auf, da diese besonders in größeren Unternehmen durch die Personalabteilung verwaltet werden.

Allgemeingültige Aussagen über nötige Qualifikationen von Softwarearchitekt:innen können durch die bisherige Analyse von Hochschullehre und Arbeitsmarkt nicht abschließend getroffen werden. Es wird Expertenwissen benötigt.

Das Expertenwissen soll einen großen Teil der Architekt:innen-Community abbilden. Bei der Auswahl der Experten ist es dabei wichtig, zum einen die Meinung von Personen einzuholen, welche sich in öffentlichen Diskussionen oder Vorträgen zur Thematik äußern. Ebenso wichtig ist die Befragung von Neueinsteigern und Dozenten. Neueinsteiger, da sie die ersten Kontakte mit der Arbeitswelt und den realen Anforderungen mit einem frischen Blick betrachten und Dozenten, die ihre Studierenden bei diesen Schritten vorbereiten und begleiten.

Zu diesem Zwecke wurden fünf Interviews mit den folgenden Personen geführt (Reihenfolge rein chronologisch des Interviewtermins):

- Dr. René Wörzberger, Professor an der TH Köln
- Jann Deterling, Senior Software Developer bei ThoughtWorks
- Eberhard Wolff, Fellow bei innoQ
- Sebastian Gauder, IT Architekt bei REWE Digital
- Dr. Gernot Starke, Fellow bei innoQ

René Wörzberger ist Professor an der TH Köln und lehrt dort in den Bereichen Softwarearchitektur, Softwaresysteme und Software-Engineering. Vor dieser Tätigkeit arbeitete er als IT-Architekt und Projektmanager. [7] Herr Wörzberger ist damit der einzige der Befragten, welcher aktuell Vollzeit in der Lehre tätig ist.

Jann Deterling hingegen ist der jüngste Befragte und hat selbst ca. 2 Jahre vor der Befragung seinen Abschluss an der TH Köln im Masterstudiengang »Informatik / Computer Sciences« gemacht. Aktuell arbeitet Jann Deterling bei Thoughtworks, einer in über 17 Ländern agierenden Technologieberatung. Thoughtworks ist damit global in einer führenden Position und beschäftigt über 10.000 Mitarbeiter:innen. [8] Herr Deterling liefert damit, die im Kreis der Befragten einzigartige Sicht, des erst kürzlichen Einstiegs in die Branche und damit verbundenen Einstiegsschulungen, Erfahrungen und mehr.

Die Herren Wolff und Starke sind beide Fellows bei innoQ. Die Bezeichnung Fellow ist eine Erweiterung eines Senior-Consultants, so „gehört zu

einem Fellow große Reputation in einem von innoQ als strategisch wichtig definierten Umfeld. Idealerweise ist ein Fellow eine bekannte Größe in diesem Themenfeld, an die sich Experten und interessierte Laien gleichermaßen wenden. Fellows sind als Autoren von einschlägigen Fachartikeln und ggf. Büchern unterwegs und sprechen regelmäßig auf Konferenzen mit großer Publikumswirkung.“ [9]

Eberhard Wolff ist durch Veröffentlichungen wie »Microservices Grundlagen flexibler Softwarearchitekturen« oder »Microservices-Praxisbuch Grundlagen, Konzepte und Rezepte« anerkannter Experte zur Thematik der Softwarearchitektur. Des Weiteren ist er Mitglied des iSAQB[3].

Gernot Starke ist ebenfalls Mitglied des iSAQB und mehrfacher Buchautor zu Thematiken rund um Softwarearchitektur und mehr. Des Weiteren ist er seit über 15 Jahren auch als Trainer in diesem Bereich tätig. [11]

Sebastian Gauder arbeitet als IT-Architekt bei REWE Digital. Dort erwarb er wertvolle Erkenntnisse, die er, u. a. bei Vorträgen auf Konferenzen weiter gibt. Sebastian Gauder ist in der Rolle des IT-Architekten im E-Commerce Bereich von REWE Digital mitverantwortlich für über 200 Microservices. [C.2.7 Gauder (1)] REWE Digital hat ihre Infrastruktur des Onlineshops in den letzten Jahren von einem Monolithen in eine Microservice-Struktur umgebaut. Sebastian Gauder, welcher einen Teil dieser Umstrukturierung begleitet hat, konnte hierbei Vor- und Nachteile im großen Format erkennen und bewerten. [C.2.7 Gauder (1)]

2.3.1 Vorgehen

Für alle Interviews wurden Leitfäden mit Fragenkatalog entwickelt, welche in Anhang C dargestellt sind. Durch dieses Vorgehen sollte die erste Forschungsfrage explizit behandelt werden und weiterhin eine Vergleichbarkeit zu den Ergebnissen aus Arbeitsmarkt und Hochschullehre geschaffen werden.

Die Forschungsfrage lautet: »Welche Erfahrungen müssen Lernende in der Softwarearchitektur machen, um die Notwendigkeit von modernen und lose gekoppelten Architekturen verstehen und umsetzen zu können?«. Es wird dabei zwischen zwei leicht verschiedenen Fragebögen unterschieden. Einem Fragebogen, welcher sehr auf die Lehre fokussiert ist und daher hauptsächlich auf René Wörzberger zugeschnitten ist und eine zweite Version, welche den Fokus auf die direkte Arbeit im Unternehmen richtet. Letzterer Fragebogen hat sich im Laufe der Interviews leicht verändert, um gewonnene Erkenntnisse mit weiteren Experten diskutieren zu können.

Die Fragen zielen, für die Beantwortung der Forschungsfrage, auf mehrere Rubriken ab:

[3] „Das International Software Architecture Qualification Board (iSAQB) ist ein Zusammenschluss führender Fachexpert:innen aus Industrie, Beratung, Ausbildung, Wissenschaft und anderen Organisationen. “ [10]

(1) Wie sah der eigene Einstieg in die Thematik der Softwarearchitektur bei den Experten aus? Dies ermöglicht Parallelen zu ziehen und kritisch zu hinterfragen, welche Punkte hilfreich oder auch hinderlich dabei waren.

(2) Wie arbeiten sich die Experten aktuell in neue Systeme ein? Gibt es dort Parallelen zwischen den Befragten und können diese Techniken auf die Lehre übertragen werden?

(3) Gibt es Methodiken oder Technologien, welche den Arbeitsprozess bei den Befragten begleiten oder schlicht über mehrere Kunden immer wieder auftauchen und daher anstrebenden Softwarearchitekt:innen bekannt sein sollten?

(4) Im Umgang mit Kollegen, besonders auch weniger erfahrenen Architekt:innen, gibt es dort weitere Eigenschaften oder Skills, welche besonders geschätzt werden bzw. wichtig sind für eine gute Zusammenarbeit?

2.3.2 Analyse

Einstieg

Der Einstieg in die Softwarearchitektur ist bei den meisten Experten ein fließender Übergang über verschiedene Arbeitsstellen oder Projekte gewesen. Einzig Jann Deterling hatte einen etwas direkteren Einstieg in die Thematik. Thoughtworks veranstaltet für alle neuen Mitarbeiter eine sechswöchige Einstiegsfortbildung. In dieser Zeit müssen die Neulinge interdisziplinär ein Projekt bewältigen. Der Fokus liegt neben der Architektur auch auf Programmierung und weiteren Methodiken, da die Teilnehmenden stark unterschiedliche Wissensstände mitbringen. [C.2.1 Deterling (1)]

Bei REWE Digital gibt es für neue Mitarbeiter kleine Einführungen, welche nach Bedarf stattfinden. In diesen werden den Teilnehmenden über einen 2-tägigen Zeitraum die wichtigsten Hintergründe zur Architektur im Unternehmen vermittelt und erläutert, warum diese so gewählt wurde. Die Einführung ist dabei in neun Module eingeteilt, welche Thematiken wie Domain Driven Design, synchrone und asynchrone Kommunikation, die mobile Lösung bei REWE und vieles mehr behandelt. [C.2.4 Gauder (4)]

Einarbeitung in neue Systeme

Die Einarbeitung in neue Systeme folgt bei allen Experten ähnlichen Mustern. Der Hauptfokus liegt zu Beginn beim Verständnis der Domäne. Neben der Anwendungsdomäne selbst wird versucht zu ermitteln, welche konkreten Aufgaben das vorliegende Softwaresystem bzw. die Systemlandschaft hat und warum diese so entstanden ist. [C.2.1] Ist die Grundlage geschaffen, kann der Fokus auf die technische Realisierung gelenkt werden. Geschieht dies bspw. in einer beratenden Tätigkeit gibt es verschiedene Methodiken

diese zu ergründen. Zum einen können Systemdiagramme besprochen werden, welche von Unternehmen zu Unternehmen jedoch meist unterschiedlich gestaltet sind. Dabei ist es wichtig, die Semantik der Diagramme zu ergründen und Übereinstimmung mit dem Code sicherzustellen. [C.2.2 Wolff (3), C.2.2 Starke (3)] Die Problematik unterschiedlicher Diagramme wird in Kapitel 2.3.2 näher erläutert. Eine andere Methodik ist, sich den aktuellen Entwicklungsprozess genau anzusehen, indem bspw. in Pair Programming an der Entwicklung eines Features teilgenommen wird, um so den Ablauf kennenzulernen. [C.2.2 Deterling (3)]

Technologien

Als Architekt:in, besonders in der Tätigkeit als Consultant oder Berufseinsteiger, ist es wichtiger breit gefächerte Technologiegrundlagen zu kennen, als umfangreiches spezifisches Detailwissen zu besitzen. Dies ist wichtig, um Probleme erst einmal richtig einordnen zu können. Es muss Architekt:innen jedoch möglich sein, sich dann konkretes Detailwissen für einen Auftrag anzueignen bzw. das Problem zu lösen.

REWE Digital musste bspw. bisher keine Architekt:innen von außerhalb des Unternehmens einstellen, da genügend qualifiziertes und interessiertes Personal über interne Stellenausschreibungen zur Verfügung steht. REWE Digital kann somit bei der Auswahl die im Unternehmen genutzten Technologien voraussetzen und andere Kriterien fokussieren. [C.2.4 Gauder (1)]

Weiterhin handelt es sich bei der Architekt:innen Rolle häufig um eine seniorige Position. [C.2.1 Wolff (2)] Durch entstandene mehrjährige Berufserfahrung im verwandten Umfeld können bereits viele Berührungspunkte mit Architekturwissen und verschiedenen Technologien gesammelt werden.

Methodiken

Ähnlich zu den Technologien gibt es auch bei den Methodiken keine zwingend zu kennenden Standards. Es ist hier ebenfalls situationsabhängig. Eine einzige durchgängig genutzte Methodik ist die ähnliche Darstellung von Architekturdiagrammen. In vielen Unternehmen wird jedoch die Semantik ganz unterschiedlich ausgelegt. Eine Tatsache, welche selbst eine Problematik darstellen kann und in Kapitel 2.3.2 weiter erörtert wird.

Die Zuhilfenahme von festen Standards in diesem Bereich, wie UML wird i. d. R. nur in sicherheitskritischen Anwendungen zur Rate gezogen. [C.2.7 Starke (3)]

Softskills

Die Kategorie der Softskills war in der ursprünglichen Befragung nur als Nebenpunkt gedacht. Über die verschiedenen Interviews hinweg steigerte sich die Bedeutung jedoch deutlich.

> *„[...] Was ich unglaublich schätze, wenn Leute auf Basis einer robusten, soliden technischen Skill-Sammlung, auch über entsprechende zugehörige, notwendige Kommunikationsfähigkeiten verfügen.“* [C.2.5 Starke (1)]

Herr Starke beschreibt damit eine Eigenschaft, welche von nahezu allen Interviewten stark hervorgehoben wurde. Es handelt sich um ein „People Business, [...] wir versuchen, mit vielen Personen zusammen, Systeme zu bauen. Und das impliziert, dass wir da miteinander reden müssen.“ [C.2.5 Wolff (1)] Das Besprochene muss dabei meist auch auf verschiedenen Ebenen kommuniziert werden können. Architekt:innen müssen sowohl mit dem Management, als auch mit dem Fachbereich sprechen können. [C.2.5 Wörzberger]

Die Kommunikation ist auch ein wichtiger Bestandteil jeder Teamarbeit und Architekt:innen betreuen i. d. R. mehr als ein einzelnes Team. Für die Umsetzung und Fehlererkennung ist es dabei wichtig, ehrliche Feedbackschleifen im Team und Unternehmen zu gewährleisten. Dies ist durch eine Vertrauenskultur möglich, „also so eine Art Feedback-Kultur im Team zu haben. Das [...] basiert natürlich alles auf Vertrauen im Team. [...] wenn man offen Feedback in einem Team teilen kann.“ [C.2.5 Deterling (2)]

Weiterhin ist es als Architekt:in wichtig neugierig zu sein und zu bleiben, um „Probleme lösen und verstehen zu wollen“ [C.2.5 Deterling (1)], da es durch den Beruf nötig ist sich wiederholt in neue Thematiken einzuarbeiten. Dies beinhaltet auch über den „Tellerrand zu gucken, eine Ebene höher“ und zu fragen: „Was kann ich hier verändern? Was kann ich hier besser machen? [...] also auch große Dinge anfassen, nicht zu denken: Okay, das ist hier was schon außerhalb meiner Grenzen, da kann ich nichts daran ändern, sondern Mut, auch an den großen Sachen was zu machen.“ [C.2.5 Gauder].

Bekannte Probleme

Die meisten Architekturprobleme sind eher spezifisch. Eine Gemeinsamkeit besteht allerdings bei vielen Problemen, sie entstehen entweder unwissentlich z. B. durch Fehler bei der Entwicklung durch Entwickler:innen und Architekt:innen oder wissentlich unter dem Einfluss äußerer Faktoren wie bspw. Termindruck. [C.2.3 Wolff (2)] „Das Klassische ist, [...] dass man, während man das Coded, schon weiß, dass das technische Schuld werden wird. [...] Aber häufig gibt es dann doch Gründe, die sagen [...] wir müssen jetzt doch lieber schneller in den Markt mit diesem neuen Feature.“ [C.2.3 Gauder (1)]

Eine weitere Problematik steckt in der Dokumentation. Viele Dokumentationen beschreiben zwar den Aufbau der Architektur, bilden allerdings nicht den Zweck und die Entscheidungsgrundlage ab. Soll das System benutzerfreundlich oder performant sein? Beide validen Anforderungen können u. a. kontrahierende Anforderungen an die Architektur stellen. Dies zu definieren und weiterhin zu beschreiben, wie die Anforderung geprüft werden kann, sollte Bestandteil der Dokumentation sein. [C.2.3 Wolff (1)]

Wie wichtig eine gute Dokumentation ist, zeigen auch die oft für die Einarbeitung wichtigen Architekturdiagramme, welche in Kapitel 2.3.2 erwähnt werden und bei schlechtem Design selbst ein Problem darstellen können. Die Diagramme werden je Unternehmen nicht nur unterschiedlich gestaltet, sondern basieren meist auf keinen einheitlichen Standards. Weiterhin entsteht ein großer Interpretationsspielraum durch die fehlende Dokumentation von Eigenheiten in Legenden, wie die exemplarische Abbildung 2.7 zeigt. Durch den Interpretationsspielraum kann es bereits nach kurzer Zeit schwierig werden, die Architektur korrekt nachzuvollziehen, geschweige denn zu erweitern oder Fehler zu suchen. [C.2.3 Starke (1)]

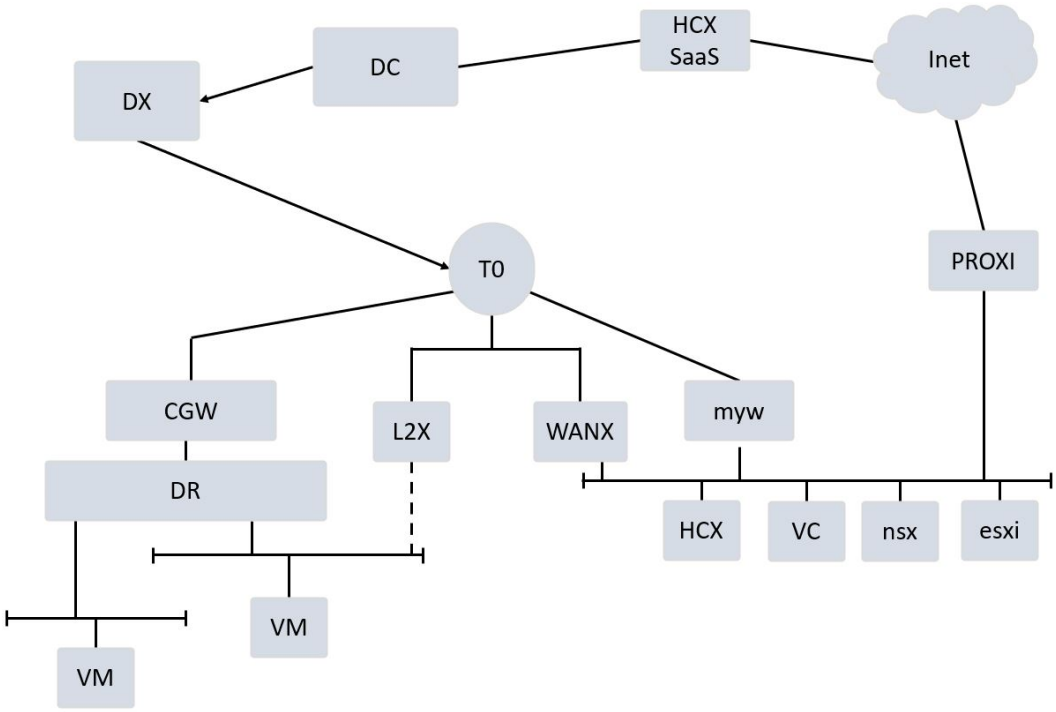


ABBILDUNG 2.7: Exemplarisches Ad-Hoc-Diagramm, welches an ein Erklärvideo angelehnt ist.[4]

Die Semantik der Pfeile, gestrichelten und durchgezogenen Linien in der Abbildung können bspw. komplett unterschiedliche Bedeutung haben, welche ohne Wissen des Erstellers nicht oder nur schwierig ergründbar sind. [C.2.3 Starke (1)]

2.4 Resümee

Die Qualifikationen von Softwarearchitekt:innen lassen sich nicht leicht anhand von spezifischen technologischem Verständnis allein benennen. Arbeitsmarkt und Hochschullehre geben viele Konzepte vor, welche die Grundterminologie und Grundkenntnisse von Architekt:innen ausmachen. Dabei ist

[4]Das erwähnte original Diagramm kann unter folgendem Link abgerufen werden und hat keinen Bezug zu dieser Arbeit. https://www.youtube.com/watch?v=uvqZgRohVZs

es weniger wichtig, die »neusten Technologien« perfekt zu beherrschen, als ein breites technisches Verständnis zu besitzen, um sich projektspezifisch einarbeiten zu können.

Die Freiheit in der Simulation keine konkrete Technologie lehren zu müssen, kann genutzt werden, um das Eigenengagement der Studierenden bspw. durch freie Wahl der Technologie zu fördern.

Ferner liegen viele wichtige Qualitäten von Softwarearchitekt:innen im Bereich der Softskills. Die Zusammenarbeit mit verschiedenen Gruppen, wie Fachbereichen und Entwicklungsabteilungen, erfordert eine gute Kommunikation und Teamfähigkeit. Die Teamfähigkeit findet sich sowohl in der Arbeitsmarktanalyse als auch in den Experteninterviews prominent wider.

Weiterhin ist die Dokumentation durch Modellierung von Architekturdiagrammen eine über alle drei Analysen hinweg aufkommende Thematik. Modellierung oder sogar spezifischer UML ist eines der wenigen in der Analyse der Module oft genannten konkreten Konzepte. Besonders in den Experteninterviews wurde die Bedeutsamkeit von guten Architekturdokumentationen sowohl für die Einarbeitung in neue Systeme als auch als Fehlerquelle bei schlechter Umsetzung genannt. Zwar wird nach Aussage der befragten Experten UML selbst nur in sicherheitskritischen Bereichen konsequent genutzt, aber die sonst genannte teils völlige Abwesenheit von Strukturen und Legenden in anderen Bereichen ist ein Problem. Den Studierenden wurde allerdings in vorangegangenen Modulen bereits der Umgang mit UML gelehrt. Eine Vertrautheit mit einem Standard ist somit bereits geben und kann für eigene Ad-Hoc-Diagramme daher sinnvoll eingesetzt werden. Für die Simulation ergibt sich daraus die Aufgabe alle Teams dazu anzuhalten gewissenhafte Dokumentation der Architektur und besonders der Entscheidungen, welche zur Architektur geführt haben zu dokumentieren.

In Gesamtheit bedeutet dies für die Simulation, dass neben den Technologien, welche für die lose gekoppelte Systemlandschaft benötigt werden keine weiteren Herausforderungen in dieser Hinsicht eingebaut werden müssen. Alle gewählten Technologien und Strukturen sollten jedoch gut und nachvollziehbar dokumentiert sein. Besonders wichtig ist die Schaffung von Rahmenbedingungen, welche es den Teilnehmenden ermöglichen, das Projekt in einem arbeitsnahem Umfeld mit kommunikativen Herausforderungen (z. B. Feedback, Absprachen teamintern und -übergreifend) zu meistern.

Kapitel 3

Simulation

Die Simulation wurde unter dem Namen »Microservice Dungeon« geplant und durchgeführt. Dieser Titel verbindet für die Studierenden einen konkreten Lehrinhalt mit der Aussicht auf eine Spielethematik. Lose gekoppelte Systeme »hands on« zu vermitteln ist nicht nur Teil des Thesis-Titels, sondern auch direktes Ziel der Simulation. Die entstehende Systemlandschaft soll auf einer lose gekoppelten Microservice-Architektur aufgebaut werden. Dabei werden Vorteile von Microservices genutzt, um bspw. die verteilte Arbeit über mehrere kleine Teams hinweg umzusetzen. [Vgl. 12, S. 28f] Weiterhin soll zwischen den Services auch asynchrone Kommunikation stattfinden, um den Studierenden die Konsequenzen und Vorteile dieses Konzeptes näherzubringen. Begonnen wird der Microservice Dungeon jedoch mit der Definition des Gameplays und des daraus abgeleiteten Komponentenschnitts. Die API aller Komponenten soll dabei in einem API-First Ansatz[1] entwickelt werden. Dies bringt den Vorteil, dass bereits frühzeitig gegen die API anderer Services entwickelt werden kann. [14]

Die genaue Zuordnung der zu vermittelnden Kompetenzen dieses Rahmens in Kombination mit den in Kapitel 2 definierten Qualifikationen wird im Kapitel 3.2 beschrieben. Das genaue Konzept der dafür genutzten Systemlandschaft schließt sich in Kapitel 3.3 an.

Der Entwurf eines neuen Moduls, welches explizit die gefundenen Qualifikationen fokussiert und den genannten Kontext abdeckt, wäre eine Möglichkeit, die Lehre zu erweitern. Neben dadurch entstehenden neuen organisatorischen Herausforderungen bei der Eingliederung und Akkreditierung neuer Module, steht noch eine weitere essenzielle Herausforderung: Den Umfang und die Komplexität des eingangs beschriebenen Rahmens in ein einzelnes Modul zu integrieren.

Durch frühzeitiges Erkennen dieser Herausforderung wurde bereits zu Beginn des Wintersemester 2021/22 mit der Planung eines modulübergreifenden Projektes begonnen. Dabei wurden vier Module in den »Microservice Dungeon« eingebunden:

[1] Bei einem API-First Ansatz wird vor der eigentlichen Entwicklung des Services die API-Spezifikation definiert. Durch einen größeren Fokus auf das API-Design und weniger den Code kann die Qualität der API gesteigert werden. [13]

- MSA - Microservice Architectures (Bachelor)
- PL - Project Launch (Bachelor)
- IP - Informatikprojekt (Bachelor)
- FAE/DDD[2] - Fachspezifischer Architekturentwurf bzw. Domain-Driven Design of Large Software Systems (Master)

Die Module haben in der Simulation teilweise unterschiedliche Rollen, wie in Kapitel 3.4.1 näher erläutert wird. Neben den beteiligten Modulen wird in Kapitel 3.4 der organisatorische Rahmen mit einer genauen Zeitplanung des Projektes ergänzt. Außerdem werden die verwendeten, teilweise vorgegebenen Technologien beschrieben.

Begonnen wird jedoch mit einem kurzen Exkurs zur Thematik der Didaktik, um einzelne didaktische Elemente zu definieren, welche im Projekt eine Rolle spielen.

3.1 Didaktik

„Die Didaktik ist [...] die »Wissenschaft« des Lernens und Lehrens“ [15] und spielt eine wichtige Rolle in jeder Lehrveranstaltung. Ohne die richtigen Lehrmittel und Methodiken sowie Motivation und Grundinteresse der Lernenden wird eine erfolgreiche Vermittlung von Lehrinhalten erschwert. Der Fokus der Simulation und dieser Arbeit liegt dabei jedoch ausdrücklich nicht auf der Findung neuer didaktischer Möglichkeiten. Einige bekannte und projektbezogene Ansätze werden in diesem Kapitel dennoch kurz erläutert, damit eine Referenzierung und Diskussion dieser innerhalb der restlichen Arbeit ermöglicht wird.

Begonnen wird dabei mit einigen wenigen Grundlagen zu biologischem und soziologischem Lernen (siehe Kapitel 3.1.1). Der wichtige Aspekt der Motivation kann in intrinsische und extrinsische Motivation aufgeteilt werden. Die Förderung beider Varianten wird in Kapitel 3.1.2 beschrieben. Abgeschlossen wird das Thema Didaktik mit der Vorstellung von Gamification, welche teilweise auch im Projekt wiedergefunden werden kann und in Kapitel 3.1.3 beschrieben wird.

3.1.1 Biologisches & soziologisches Lernen

Nachfolgend wird biologisches Lernen behandelt, um die natürlichen Lernmöglichkeiten der Studierenden im Projekt unterstützen zu können. Das biologische Lernen richtet sich nach den natürlichen Strukturen im Gehirn. [16, S. 50f] Eine optimale Nutzung der biologischen Gegebenheiten kann das

[2]Inhaltlich identisches Modul, welches mit der Neuakkreditierung des Masterstudiengangs umbenannt wurde. Mehr Informationen im Kapitel 3.4.1

Lernpotential steigern. Dazu gehört die Nutzung der Stärken beider Gehirnhälften. So finden in der linken Gehirnhälfte Logik, Planung, Ordnen, Analyseprozesse uvm. statt. Während die rechte Gehirnhälfte die Verarbeitung von Bewegungen, Geräuschen, Farben, Bildern u. a. steuert. Die Kombination vieler Sinne beider Gehirnhälften erhöht den Lernerfolg. [16, S. 60ff]

Durch Wiederholungen und Verknüpfungen mit Bekanntem kann der Lernerfolg ebenfalls gesteigert werden, besonders wenn zusätzliche Lernformen genutzt werden. [16, S. 45f]

Soziologisches Lernen orientiert sich hingegen an der Gesellschaft und „[...] allen Formen und Bedingungen menschlichen Zusammenseins“. [16, S. 51] Dies beinhaltet z. B. Lernen mit anderen zusammen, voneinander und im gegenseitigen Austausch.

3.1.2 Intrinsische und extrinsische Motivation

Motivation ermöglicht das Erreichen eines festgesteckten Ziels, auch wenn dieses längeres Durchhaltevermögen benötigt. [17][18, S. 224] Das »Microservice Dungeon« Projekt zieht sich für fast alle Projektbeteiligten über ein gesamtes Semester und erfordert dabei langfristiges und dauerhaftes Engagement. Eine durchgängige und hohe Motivation aller Teilnehmer:innen ist daher wichtiger Bestandteil für den Gesamterfolg des Projektes. Motivation oder Selbstregulation gehören zu den entscheidenden Faktoren für effektives Lernen. [18] Die Motivation zu fördern, ist daher wichtiger Aspekt von Lehrveranstaltungen. Es gibt zwei Arten von Motivation, intrinsische und extrinsische Motivation. Diese unterscheiden sich dabei durch ihren Ursprung. Intrinsische Motivationen, »von Innen heraus«, beschreiben dabei Motivationen, welche aus dem eigenen Antrieb einer Person stammen. Dazu gehören z. B. Neugierde oder emotionale Anreize wie Anerkennung und Zugehörigkeitsgefühl. Bei extrinsischer Motivation liegt eine Außenwirkung vor. Eine Person wird durch positive oder negative Verstärkung zu einer Handlung bewegt. [19, 20]

„Intrinsische Motivation gilt im Allgemeinen als der extrinsischen Motivation überlegen in Bezug auf die Freude am Lernen und auf den Lernerfolg.“ [18] Intrinsischen Motivationsfaktoren wie die Neugierde und die innere Freude an der Arbeit können so langfristig und kontinuierlich die eigene Motivation erhalten und steigern. Es entsteht eine durchgehend höhere Lernbereitschaft und Leistungsfähigkeit, welche durch eine fortlaufende Zufriedenheit unterstützt wird. [21]

„Widmet man sich einer Tätigkeit wegen ihrer Konsequenzen (Erreichen positiver Konsequenzen oder Vermeiden negativer Konsequenzen), so wird diese Tätigkeit als extrinsisch motiviert bezeichnet.“ [22]

Das wohl häufigste Realbeispiel für eine positive Konsequenz ist der monetäre Anreiz für jeden Arbeitnehmer. Für Studierende könnten positive Konsequenzen bspw. gute Noten sein. Negative Konsequenzen im Allgemeinen

können die Vermeidung von Bestrafungen oder sonstigen entstehenden Nachteilen darstellen.

Extrinsische Motivation bietet den meisten Menschen damit weniger Leistungsbereitschaft als intrinsische Faktoren. Die Kombination beider kann dazu führen, dass die extrinsische Motivation die intrinsische abschwächt oder sogar negiert. [Vgl. 22]

3.1.3 Gamification

Gamification zu Deutsch Spielifizierung beschreibt die Verwendung spielerischer Elemente in einem spielfremden Kontext. [23] Dieser Motivationsfaktor wurde in der Endphase der Simulation genutzt, um den Studierenden einen zusätzlichen Anreiz in der arbeitslastigen Endphase des Projekts zu bieten.

Im Bereich der Gamification ist Yu-kai Chou einer der Pioniere. Mit dem »Octalysis Framework« hat er acht typische, durch Spiele angesprochene Motivationsfaktoren ausgearbeitet. Diese bezeichnet er als »Core Drives«, also als treibende Kraft hinter dem Handeln (siehe Abbildung 3.1). Dabei spiegelt jeder Core Drive Aspekte wider, durch welche Personen mittels Gamification motiviert werden können. [24, S. 23ff]

Im Folgendem werden zwei der acht Core Drives beschrieben, welche die größte Relevanz für die Simulation besitzen (siehe Player-Entwicklung in Kapitel 3.3). Begonnen wird mit dem zweiten Core Drive »Development & Accomplishment«. Dieser beschreibt, dass durch die eigene Entwicklung und die eigene Leistung ein Fortschritt erzielt werden kann. [24, S. 89ff] Eine Beschreibung, die den Drive als Teil der intrinsischen Motivationen kennzeichnet. Ein weiterer verbundener Punkt, welchen Chou als »besonders wichtig« bezeichnet, ist der Aspekt der Herausforderung, denn ohne eine Herausforderung ist auch die Leistung unbedeutend. [Vgl. 24, S. 25f]

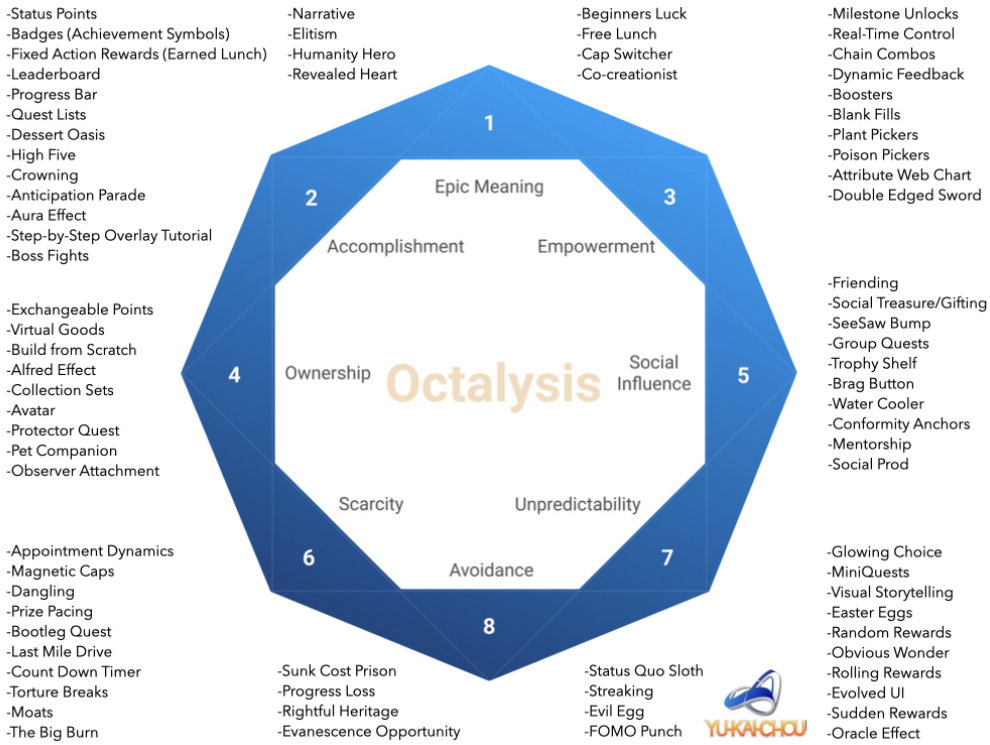


ABBILDUNG 3.1: Octalysis Framework: Die acht »Core Drives« mit Zuordnung zu bekannten Spielefeatures[3]

Der andere im Fokus des Projektes stehende Core Drive, ist der Vierte »Ownership & Possession«. Ownership & Possession oder zu Deutsch »Besitz und Eigentum« beschreibt das Verlangen Dinge zu besitzen, zu verbessern oder zu beschützen. Dazu kann auch die Zugehörigkeit zu einem Prozess oder Projekt gehören. Verstärkend kann dabei wirken, wenn die zusammenhängenden Gegenstände individuell anpassbar sind. [24, S. 160ff]

Gamification wurde bereits in vielen Bereichen erprobt. In der Lehre wird Gamification oft zur Begleitung von Vorlesungen eingesetzt. Die Studierenden werden hierdurch motiviert, schon früh im Semester Lerninhalte zu wiederholen. [25]

Gegensätzlich der durch die Spielthematik möglichen Erwartung, dass die Simulation einen Fokus auf Gamification legt, spielt die Gamification eine kleinere Rolle. Die allgemeine Erfahrung und Vertrautheit zum Spielkontext im Vergleich zu bspw. Finanzsoftware zielt jedoch auf eine starke intrinsische Motivation der Teilnehmenden ab, welche sich aktiv für das Projekt bewerben müssen (siehe Kapitel 3.4.1). Über das Semester hinweg sind keine Gamificationelemente rund um die Entwicklung konzipiert worden. In der Abschlussphase des Projektes wird jedoch auf Gamificationelemente zurückgegriffen, wie in Kapitel 3.3 beschrieben.

[3]Quelle: https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

3.2 Kernkompetenzen

Die Qualifikationen, die bereits in Kapitel 2 definiert wurden und Softwarearchitekt:innen besitzen sollten, werden folgend der Simulation und den einzelnen Modulen zugeordnet. Die Kompetenz, welche über alle vier Module hinweg gelehrt wurde und damit im Fokus stand, ist das Erleben und Arbeiten in einer lose gekoppelten Systemlandschaft. Dabei sind viele verbundene Konzepte wie asynchrone Kommunikation, Domain Driven Design u. a. allen Teilnehmenden näher gebracht worden. Diese sind wichtige Kompetenzen für den Einstieg in die Berufswelt. [C.2.4 Gauder (4)] Durch die Technologiefreiheit konnten zwar Programmiersprachen und Frameworks frei gewählt werden, jedoch war im Projekt ähnlich zur Wirtschaft die Programmiersprache Java zusammen mit dem Framework Spring weit verbreitet. [Vgl. Kapitel 2.1.2, C.2.4 Deterling (2)]

Neben diesen »Hard Skills« standen auch einige Softskills im Vordergrund. Diese sind bspw. durch die Simulation einer arbeitsnahen Umgebung und den damit verbundenen Kommunikationshürden Teil des Projekts gewesen.

Dieser geforderte Austausch war besonders für die Module FAE, IP und PL großer Teil des Projektes, da diese über das gesamte Semester Absprachen treffen mussten. Die drei Module waren weiterhin Teil der ersten Detailplanung und des Komponentenschnitts der Microservice Systemlandschaft. In diesem wurden erneut Kompetenzen im Domain Driven Design sowie Bounded Context vermittelt.

Die Teilnehmer:innen des FAE/DDD-Moduls mussten durch ihre besondere Stellung als Architekt:innen und Product Owner (PO) auch weitere Softskills beweisen (mehr Details in Kapitel 3.4.1). Es war nötig, die Absprachen auf Architekt:innenebene zu treffen und in die eigenen Serviceteams zu tragen. Die daraus resultierenden Aufgaben mussten aus PO Sicht priorisiert werden. Weiterhin war es erforderlich, die Entscheidungen auf beiden Ebenen in einem Decision Log zu dokumentieren. Verständliche und gut gepflegte Architekturdokumentationen sollten geübt sein, da deren Fehlen oder Unvollständigkeit oft zu Problemen in reellen Projekten in der Wirtschaft führen können. [C.2.3 Wolff (1), C.2.3 Gauder (2)]

Teile der genannten Kompetenzen waren nicht an die Benotung der Studierenden gebunden. Dies trifft besonders auf alle Kompetenzen im Bereich der Softskills zu. Die Simulation einer arbeitsnahen Umgebung war dabei geplante Erfahrung des Projektes und nicht Teil des Notenrahmens.

Eine Benotung wird jedoch ausgeschlossen, da die Nachvollziehbarkeit für die Studierenden sowie die faire Kriteriengebung durch die Lehrenden nicht in einem vertretbaren Zeitrahmen gestaltet werden kann. Besonders durch die fast ausschließlich Remote gestaltete Projektumgebung. Der Lehrauftrag für diese Module liegt per Definition nicht auf der Vermittlung von Softskills, sondern auf der Lehre von spezifischen »Hard Skills«, wie der asynchronen Servicekommunikation. Neben den technischen Skills ist auch die korrekte Dokumentation Teil der Benotung gewesen.

3.3 Konzept

Die Simulation benötigt eine Thematik, welche komplex genug ist, eine asynchrone Servicelandschaft zu rechtfertigen. Weiterhin sollte die Thematik trotzdem für die Studierenden leicht verständlich sein, sodass Probleme beim Verständnis der Materie bspw. einer Finanzsoftware und den damit verbundenen Fachbegriffen nicht zu architektonischen Fehlentscheidungen führen.

Die gewählte Thematik ist durch das Praktikum im Modul »Softwaretechnik II« des vorangegangenen Semesters inspiriert. In diesem wurden den Studierenden u. a. der Umgang mit Spring Boot gelehrt. Der Kontext war hierbei die Entwicklung eines eigenen Roboters, welcher sich über eine kleine Karte bewegen konnte. Dieser spielerische Ansatz wurde auch für die Simulation übernommen, da dadurch direkt drei Vorteile kombiniert werden konnten.

(1) Es gab bei den Teilnehmenden aus dem Modul Softwaretechnik großes Interesse an einem solchen möglichen Projekt.

(2) Die Zielgruppe der Studierenden ist durch ihren Altersbereich generell Spieleerfahren und bringt somit eine Affinität für Begrifflichkeiten und Abläufe mit. [26]

(3) Die Gamification ermöglicht auch aus Sicht der Didaktik einen weiteren Motivationsfaktor, wie bereits in Kapitel 3.1.3 beschrieben wurde.

Die Thematik wurde somit dahingehend erweitert, dass eine größere Systemlandschaft entstehen konnte. Im ersten Schritt mussten das Spiel und seine Mechaniken konzipiert und entwickelt werden. Die Konzeption genauer Spielregeln wurde, abgesehen von der Roboter-Idee, einer Teilgruppe der Studierenden übertragen (siehe Kapitel 3.4.1).

Die spezifizierten Spielregeln wurden allen Teilnehmenden präsentiert. Mit dem gesammelten Wissen wurde ein »Event Storming« Workshop abgehalten, welcher die Thematik vertiefen sollte. Ein gutes Domänenverständnis ist für die Einarbeitung in ein neues System wichtig. [Vgl. C.2.2] Ziel des Workshops war der Serviceschnitt der Komponenten.

Event Storming ist eine von Alberto Brandolini entwickelte Workshopform mit dem Ziel, Prozess und Ablauf einer Domäne verstehen und abbilden zu können. Im Workshop wird dabei durch die Nutzung verschiedenfarbiger »Sticky Notes« ein Ablauf von Domänen Events in eine logische Reihenfolge gebracht. Als Beispiel einer so entstehenden Übersicht ist in Abbildung 3.2 ein Auszug des Resultates des Workshops dieser Simulation abgebildet. Der Workshop lebt von der Interaktion zwischen allen Beteiligten. Es gibt dabei keine klassische Leitungsrolle, sondern nur Moderation, um die Teilnehmenden bei Fragen zum Workshopablauf selbst zu unterstützen. [Vgl. 27]

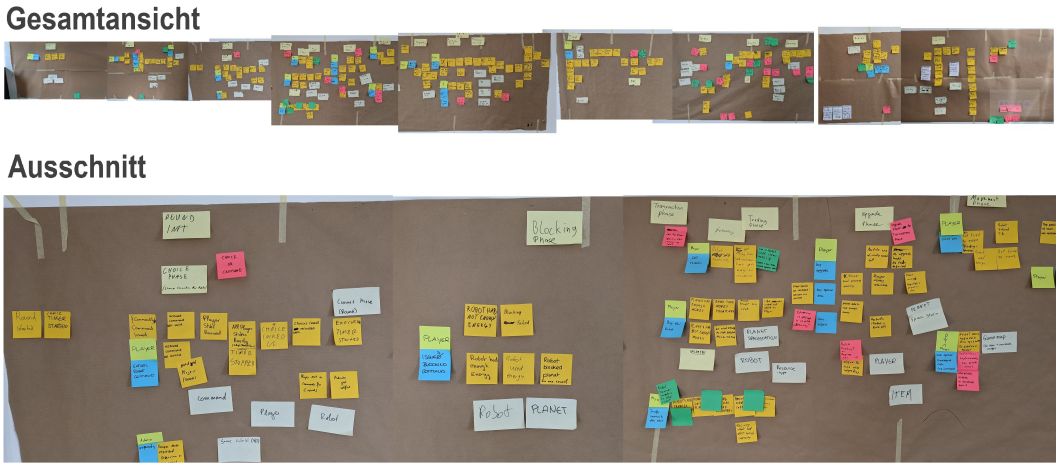


ABBILDUNG 3.2: Übersicht und Detailausschnitt der Ergebnisse des Event Storming Workshops

Der Beginn des Projektes mit einem Event Storming Workshop bringt mehrere Vorteile. Zum einen fördert der Einstieg in die Materie über Event Storming viele Lernsinne und festigt von Beginn an das Verständnis der Materie (siehe Kapitel 3.1.1). Des Weiteren fördert die durch den Workshop forcierte Interaktion zwischen den Teilnehmenden die Kommunikation und Feedback-Kultur bereits zu Beginn des Projektes.

Nach dem Event Storming Workshop kann durch den daraus resultierenden Komponentenschnitt die Entwicklung der Services starten. Die Services werden in Projektteams von je zwei bis fünf Studierenden entwickelt. Die so entstehende Projektlandschaft simuliert eine reelle Entwicklung mit verschiedenen hierarchischen Strukturen (mehr Details in Kapitel 3.4.1).

Nach Abschluss der Serviceentwicklung können die Teilnehmenden mit den von ihnen entwickelten Robotersteuerungen in der selbstgebauten Systemlandschaft antreten. Die Player-Services sind Teil der Systemlandschaft und erfordern damit in der Service-Entwicklung Gelerntes zu wiederholen, wodurch eine Steigerung des Lerneffekts erreicht werden soll. [Vgl. Kapitel 3.1.1]. Weiterhin zielte diese Projektphase direkt auf einen Motivationsschub zum Ende des Projektes durch Nutzung von Gamification-Elementen ab. Besonders die beiden in Kapitel 3.1.3 beschriebenen »Core Drives« sind durch die Playerentwicklung abgedeckt. Bspw. wird eine starke Ownership durch die komplette Anpassbarkeit des eigenen Spielers gegeben.

Die Player-Komponenten nutzen dabei verschiedene Vorteile von Microservices sowie der Systemlandschaft (siehe Kapitel 3.3.1) aus. Für die Player gilt, wie auch für alle anderen Services, dass die Wahl der Technologie komplett freigestellt ist. Dadurch sind bei der Playerentwicklung Services in bspw. Java, Kotlin, TypeScript, Elixir und Erlang entstanden. Die Systemlandschaft war weiterhin so designt, dass es theoretisch für beliebig viele Player möglich ist nebeneinander zu existieren. Dies wurde nur aus Speicherplatz- und Spieldesigngründen begrenzt. Des Weiteren wurde die Einhaltung aller Spielregeln durch die restliche Systemlandschaft sichergestellt, wodurch auch der

Quellcode der Player nicht auf Täuschungsversuche hin geprüft werden musste.

3.3.1 Systemlandschaft

Der Komponentenschnitt erzeugte dabei fünf verschiedene Komponenten, welche in eigenen Microservices umgesetzt wurden. Abbildung 3.3 stellt die Services sowie synchrone Aufrufe in der Systemlandschaft dar.

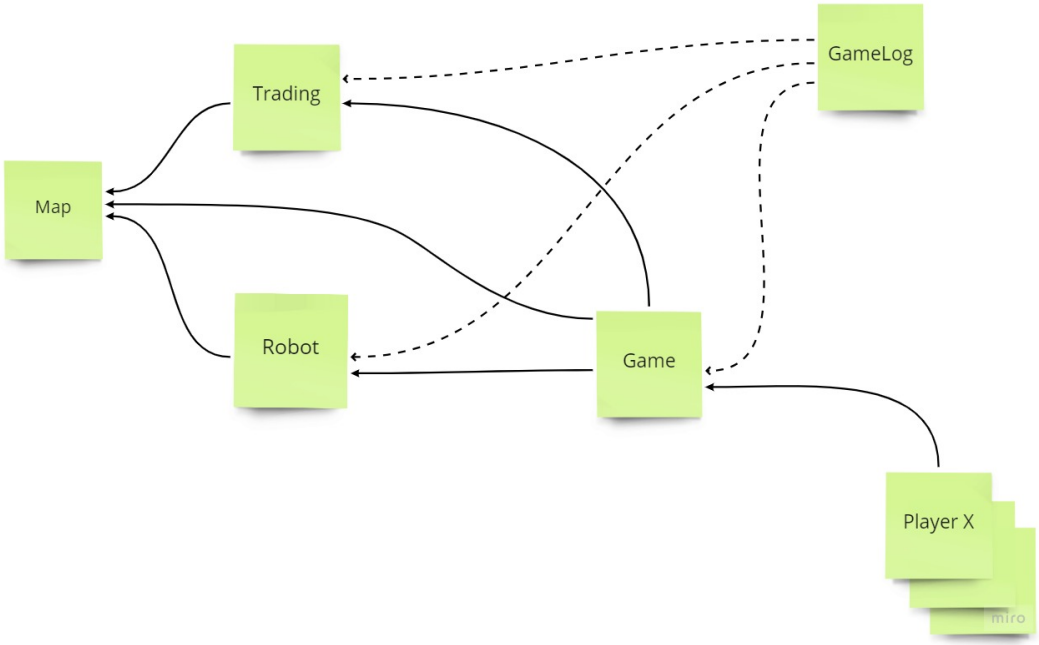


ABBILDUNG 3.3: Systemlandschaft der Simulation (Die Pfeilrichtugen stellen synchrone Aufrufbeziehungen dar. Die unterschiedliche Darstellung der Pfeile dient einzig der Übersichtlichkeit)

Zwar entsprach die aus dem Event Storming generierte Systemlandschaft nicht 1:1 den in Vorplanung durch die Lehrenden generierten Ideen, allerdings stellt dies keine Problematik dar und spiegelt den interaktiv designten Projektablauf wider.

Die fünf daraus resultierten Services (Player ausgenommen) lauten:

- Game
- Robot
- Trading
- Map
- Gamelog

Der Game-Service verwaltet alle Spieler, indem er Anmeldung und Authentizität der Spieler sicherstellt. Des Weiteren nimmt er alle Kommandos entgegen und stellt damit eine Schnittstelle zwischen den Playern und der Systemlandschaft dar. Der Gamelog-Service ist nicht am direkten Spielablauf und an der Kommunikation mit den Playern beteiligt. Der Service sammelt Informationen aller anderen Services und stellt Ranglisten und Errungenschaften dar. Der im Komponentenschnitt größte Service ist der Robot-Service, welcher die Verwaltung aller Roboter übernimmt. Dieser berechnet die Ergebnisse von Kämpfen oder bewegt die Roboter. Der Trading-Service verwaltet die gesamte Währung im Spiel. Jeder Spieler besitzt pro Spiel ein Startguthaben und kann beim Trading-Service Rohstoffe verkaufen oder Upgrades für die Roboter erwerben. Als letzter stellt der Map-Service die Spielwelt zur Verfügung und generiert diese anhand der teilnehmenden Spieler. Zur Spielwelt gehören weiterhin alle Rohstoffe und Handelsposten.

Neben den Teams für die einzelnen Komponenten der Systemlandschaft wurde ein DevOps-Team zusammengestellt, dessen Aufgabe das Aufsetzen und Betreiben der geeigneten Infrastruktur ist. Aufgaben, welche im strengen agilen Sinne auch in den eigenen Teams durchgeführt werden. Um die Arbeitslast der einzelnen Teams jedoch zu reduzieren, wurde der DevOps Aufgabenbereich ausgelagert.

3.3.2 Spielablauf

Läuft der Game-Service, können sich alle Player für das Spiel anmelden. Ist die Anmeldephase vorüber, können die Administratoren, welche durch das Gameplayteam von Project Launch gestellt werden, das Spiel starten.

Der Game-Service initiiert das Spiel, indem er die Erstellung der Spielwelt im Map-Service anstößt. Alle nötigen Informationen zu Startguthaben u. Ä. wird von den entsprechenden Services via Event an die Player übermittelt. Die Player haben nun die Möglichkeit ihr Startguthaben auszugeben, um Roboter zu kaufen und diese entsprechend über die Karte zu bewegen. Mögliche Ziele der Player sind dabei:

- Erkundung der Karte
- Sammeln und Verkauf von Rohstoffen
- Zerstörung gegnerischer Roboter
- Aufwertung der eigenen Roboterflotte
- sowie Spaß und Freude am Spiel

Einzelne Aktionen sind pro Roboter und Runde begrenzt. Der Game-Service regelt dabei die Rundenzeit und veröffentlicht mittels Events, wenn diese voranschreiten.

Das Spiel hat keine direkte Siegbedingung und endet nach einer definierten Rundenzahl. Die besten Spieler verschiedener Kategorien werden im Anschluss durch die Ergebnisse des Gamelogs ermittelt.

3.4 Rahmenbedingungen

Die Rahmenbedingungen befassen sich mit den beteiligten Modulen, durch welche definiert ist, wer und wie viele Personen am Projekt teilnehmen. Die einzelnen unterschiedlichen Wissensstände der Teilnehmenden müssen folglich beachtet werden, um ihnen dadurch die geeigneten Rollen zuzuweisen und somit die größten Lernerfolge zu ermöglichen.

Aus Sicht der Lehrenden muss weiterhin auch ein technischer Rahmen gesteckt werden, welcher zum einen eine gewisse Kontrolle und Unterstützungsmöglichkeit schafft und zum anderen die Lehrenden in die Lage versetzt, eine faire Benotung durchzuführen.

Abschließend werden der zeitliche Rahmen und die damit verbundene Planung, aber auch Planungsänderungen im Verlauf des Projektes beschrieben.

3.4.1 Beteiligte Module

Der Umfang des Projektes erfordert auch in der Auswahl der Projektteilnehmer:innen eine größere und diversere Gruppe. Eine so entstehende Anzahl von ca. 50 Personen kann dabei nicht aus einem einzelnen Modul zustande kommen.

Im gesamten Projekt soll eine komplexe Simulation nicht nur anhand der erweiterten Systemlandschaft, sondern auch des dazugehörigen organisatorischen Rahmens gewährleistet werden. Dies schafft die Komplexität und einhergehenden Problematiken in Kommunikation und gegenseitigen Anforderungen, welche feste Bestandteile echter Projekte sind. Es entsteht eine Miniaturorganisation, welche nach »Conway's law« die Systemlandschaft widerspiegelt. [28]

Die Teams für jeden Service bestehen aus Entwicklern, Architekten und Product Ownern. Die Arbeitsweise wird dabei an agilen Methodiken angelehnt, soweit dies im Hochschulkontext und im Zeitrahmen eines Semesters möglich ist. Arbeitsschritte werden bspw. in 2-wöchigen Sprints geplant und bearbeitet. Stets mit dem Fokus, ein Fundament zu schaffen, auf welches im Anschluss aufgebaut werden kann. Falls nötig können Features in den ersten Schritten reduziert werden, um spätere Erweiterbarkeit zu ermöglichen.

Die Kommunikation und Abstimmungen im gesamten Projektteam werden hauptsächlich auf Architekten- und Product Owner Ebene getroffen und anschließend in die Teams getragen.

Project Launch

Das Modul »Project Launch« nimmt mit fünf Personen am Projekt teil, besitzt ein Arbeitsvolumen von 13 ECTS[4] und ist wie MSA (siehe Kapitel 3.4.1) eines der zwei Module aus dem »Code and Context« Studiengang. Code and Context ist ein Bachelorstudiengang, welcher am Campus Köln-Mühlheim gelehrt wird. Das Lehrkonzept wird dabei als „bunt, innovativ, kreativ, interdisziplinär und kritisch“ [29] beschrieben. Dies spiegelt sich auch in der Modulbeschreibung des »Project Launch« wider: „Die Studierenden können [...] eine innovative Projektidee vom ersten Prototypen zum Minimum Viable Product entwickeln, Betreiber- oder Geschäftsmodelle für kommerzielle Produkte, Dienstleistungen, freie Software oder kollaborative Anstrengungen beschreiben, [...] und die Kompetenzen der begleitenden Kurse anwenden und in einen Zusammenhang bringen“. [30]

Die Teilnehmer:innen von Project Launch mussten sich ebenso wie die Teilnehmenden des Informatikprojektes (siehe Kapitel 3.4.1) für die Teilnahme am Projekt bewerben. Bei den Teilnehmenden beider Module ist somit das Projektkonzept bekannt. Durch die bewusste Entscheidung zur Bewerbung wurde bei den Teilnehmenden beider Module eine verstärkte intrinsische Motivation (siehe Kapitel 3.1.2) gezeigt.

Das Modul Project Launch wird in drei Blockphasen gelehrt. Die ersten beiden dauern jeweils zwei Wochen. Die dritte Blockphase beinhaltet vier Wochen.

Im Projekt nimmt Project Launch dadurch direkt mehrere Rollen ein. (1) Als Gameplaydesigner[5]: Der Entwurf des Gameplays ist neben organisatorischen Aufgaben und dem Event Storming am Ende des Blocks die Hauptaufgabe der ersten zwei Wochen. Diese erste Iteration des Gameplays dient als Grundlage für den Entwurf der Systemlandschaft. Auch im weiteren Projekt gilt das Team des Project Launch als Hauptansprechpartner bei Gameplayfragen. (2) Als Service-Entwickler: Im zweiten Block steht die Entwicklung eines eigenen Services aus der Systemlandschaft im Vordergrund, welcher im dritten Block verfeinert werden kann. (3) Als Player-Entwickler: Im dritten und größten Block beteiligen sich die Teilnehmer:innen von Project Launch auch an der Entwicklung eines eigenen Player-Services und beweisen so, dass sie nicht nur den eigenen Service, sondern auch das gesamte Zusammenspiel verstehen und nutzen können.

Der letzte Block wird mit einer Abschlusspräsentation des gesamten Teams abgerundet.

[4]ECTS oder »European Credit Transfer System«, ist ein System, welches den Aufwand einzelner Studiengänge, Projekte, Module, etc. beschreibt. Die Anzahl stellt dabei dar, wie viel Aufwand durchschnittlich aufgewendet werden muss, um bspw. ein Modul zu bestehen. 1 ECTS entspricht dabei 30h Arbeitszeit.

[5]Gameplaydesigner kreieren oder entwerfen Spiele. Vom Szenario und der Welt um das Spiel herum bis hin zu den einzelnen Spielregeln und Belohnungen.

Informatikprojekt

Das Informatikprojekt ist Teil des Informatik Bachelorstudienganges, welcher am Campus Gummersbach gelehrt wird. Mit 15 Personen und 10 ECTS je Person handelt es sich hierbei um die Teilgruppe mit dem größten Arbeitsvolumen. Der Hauptteil der Module im Informatikbachelor, wie »Softwaretechnik II«, beträgt zum Vergleich 5 ECTS. Das Informatikprojekt stellt dabei selbst kein klassisches Modul mit Vorlesungen und Klausur dar, sondern betitelt ein Projekt des fünften und damit vorletzten Plansemesters des Informatikstudiengangs, in welchem die Studierenden die bis dahin erlernten Fähigkeiten in einem komplexeren Umfeld anwenden sollen. [31]

Solche Projekte können daher von allen Professor:innen nach eigenem Ermessen und in Absprache mit den Studierenden durchgeführt werden.

Die Aufgabe des Informatikprojektes in der Simulation ist nach Teilnahme am Event Storming die Umsetzung der verbleibenden Services. Hierfür wurde eine Einteilung in Gruppen vorgenommen. In Abhängigkeit von der absehbaren Komplexität einzelner Services variierte hierbei die Gruppengröße.

Nach Abschluss der Serviceentwicklung müssen die Teilnehmer:innen auch an der Entwicklung eigener Player mitwirken, wobei die Gruppenzusammenstellungen mit weniger Begrenzungen neu und nach eigenem Ermessen geplant werden konnten. Auch die Zusammenarbeit mit anderen Modulen ist in diesem Zeitraum möglich.

Fachspezifischer Architekturentwurf /
Domain-Driven Design of Large Software Systems

Das Mastermodul besitzt aktuell zwei Bezeichnungen, da dieses durch die Neuakkreditierung für das Wintersemester 2021/22 umbenannt und für internationale Studierende geöffnet wurde. Sowohl FAE (Fachspezifischer Architekturentwurf) wie auch DDD (Domain-Driven Design of Large Software Systems) haben einen starken Softwarearchitektonischen Schwerpunkt, wie die folgenden »Learning Outcomes« des Moduls DDD [32] zeigen:

Learning Outcomes DDD

- As an experienced programmer or architect, I can design and implement a reasonably complex greenfield application in a multi-team approach, using the domain-driven design paradigm,
- by
  - exploring the domain and defining appropriate bounded contexts for the teams,
  - picking the suitable architectural style, according to the goals of for my software,
  - understanding the organisational preconditions wrt. DevOps,
  - defining service boundaries,
  - defining and implementing REST APIs in a suitable style,
  - defining and implementing events, using the appropriate architecture patterns,
  - roadmapping the UI architecture, and
  - reflecting my architecture and development process
- so that the prototype that is jointly created during the course is sound and sustainable wrt. architecture and coding style.

Dies spiegelt sich auch in der Rolle der Teilnehmer:innen von FAE und DDD wider, welche mit einem Arbeitsvolumen von 6 ECTS am Projekt teilnehmen. Sie besitzen eine Doppelfunktion als Architekt:innen und Product Owner.

Als Architekt:innen müssen Entscheidungen für die Services im Team selbst, sowie über das gesamte System getroffen und dokumentiert werden. Als Product Owner sind sie für die Planung und Priorisierung der Aufgaben der einzelnen Service-Teams zuständig. Für beide Rollen wurden je Service zwei Personen aus FAE/DDD zugewiesen. Die Masterstudierenden haben somit entsprechend dem höheren Grad, eine größere Verantwortung für das Gesamtprojekt als die einzelnen Service-Teams aus den Bachelormodulen.

Microservice Architectures

Das zweite Modul aus dem Code and Context Studiengang ist Microservice Architectures (MSA). MSA wird ebenfalls in Blockform von 2 Wochen gelehrt. [33] Durch den geringen Umfang von MSA ist es nicht möglich, Teile der Systemlandschaft mitzuentwickeln oder zu gestalten. Da MSA zeitlich jedoch erst gegen Ende des Semesters stattfindet, ist es für die Teilnehmer:innen möglich, eigene Player zu entwickeln. Nach einem kurzen theoretischen Input zu der bestehenden Systemlandschaft beginnt die Entwicklung.

3.4.2 Technologien

In vielen Bereichen sollte den Service-Teams größtmögliche Technologiefreiheit gewährt werden. So konnten Teams Programmiersprachen, Frameworks

oder Datenbankumgebungen für ihre Services frei wählen, wodurch Services in Ruby, Java oder Kotlin entwickelt wurden.

In manchen Bereichen musste diese Freiheit jedoch eingeschränkt werden, um eine Bewertbarkeit durch die Lehrenden sowie einen reibungslosen Ablauf zu gewährleisten. Das Hosting wurde bspw. mittels einer von der TH bereitgestellten virtuellen Maschine (VM) durchgeführt. Diese Ubuntu-VM hostet die komplette Systemlandschaft mittels Docker.

Eine weitere Einschränkung war die Nutzung von GitHub als Sourcecode Versionierungsplattform. Mit eigener Organisation[6], welche für dieses Projekt angelegt wurde. So konnte eine einheitliche Form für die Codeablage und Dokumentation gewährleistet werden und ebenso eine gute Übersicht über aktuelle Entwicklungsstände.

Für die Architekt:innen wurde eine weitere Bedingung formuliert. Alle Architekturentscheidungen mussten im Decision Log[7] festgehalten werden. Der Decision Log ist eine Eigenentwicklung des ArchiLabs und ist eine Dokumentationsform, welche die übersichtliche Darstellung mittels Jekyll[8] und GitHub-Pages[9] ermöglicht, indem bspw. Entscheidungen mit Verantwortlichem, aktuellem Status und Deadlines dargestellt werden. Aufgabe und Idee des Decision Logs sind dabei Architecture Decision Records (ADR) nachempfunden.

3.4.3 Zeitgestaltung

Die Zeitgestaltung der Simulation bindet sich an das Wintersemester 2021/22, dessen Vorlesungszeitraum vom 04.10.2021 bis zum 04.02.2022 war. Durch diese Bindung wurde ermöglicht, über den gesamten Projektzeitraum die volle Personenstärke einplanen zu können. Einzig hiervon ausgenommen sind die Module des Code and Context Studiengangs Project Launch und Microservice Architectures. Wegen ihrer festgelegten Blockwochen war es eine Herausforderung sie korrekt mit in den Zeitplan einzubeziehen.

Die gesamte Zeit wird dabei in drei Hauptblöcke eingeteilt:

1. Gameplay-Grundzüge und Event-Storming
2. Service-Entwicklung
3. Player-Entwicklung im Hackathon

[6]Die GitHub-Organisation ist unter: https://github.com/The-Microservice-Dungeon verfügbar.
[7]Der Decision Log ist unter https://the-microservice-dungeon.github.io/decisionlog/ zu finden.
[8]Jekyll ist ein statischer Webseitengenerator, welcher es ermöglicht einfache Texte bspw. via Markdown in durchdachten Designs einzubetten.
[9]GitHub-Pages ist eine Funktion von GitHub, welche das Hosting einer eigenen Website für eine Repository ermöglicht.

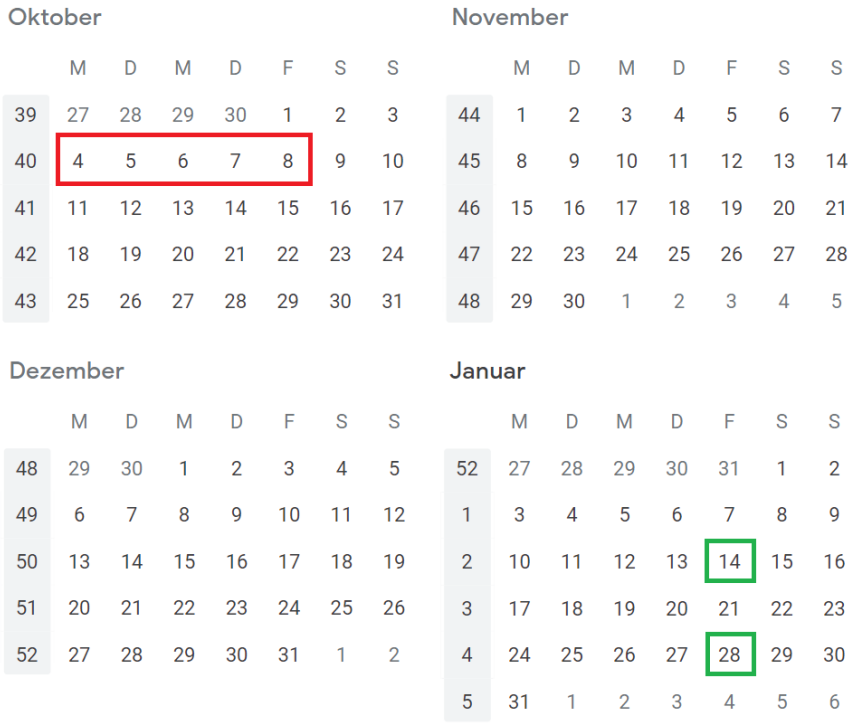


ABBILDUNG 3.4: Kalenderauszug des Projektzeitraums

Das Event-Storming fand vom 04. bis zum 08.10.2021 (im Abbildung 3.4 rot markiert) statt und fiel damit in die erste Blockwoche des Project Launch Modul, wodurch die Teilnehmer:innen aller Module ausgenommen MSA teilnehmen konnten. Die Veranstaltung wurde dabei vor Ort am Campus Köln-Mühlheim in der Schanzenstraße durchgeführt.

Der größte Block war für die Entwicklung der Services selbst vorgesehen, vom 11. Oktober bis zum 22. Dezember 2021. In dieser Zeit fanden regelmäßige Treffen auf verschiedenen Ebenen statt. Jedes Serviceteam traf sich im zweiwöchentlichen Turnus zu einem Sprint-Planning/Review mit den Betreuern (Product Owner/Architekt:innen) von FAE/DDD, um aktuelle Probleme und die nächsten Aufgaben zu besprechen. Jeweils zusätzlich anwesend war einer der Lehrverantwortlichen.

Ebenfalls in einem zweiwöchentlichen Turnus fanden die Vorlesungen für das FAE/DDD-Modul statt. Hier hatten die Architekt:innen die Möglichkeit, gemeinsame Beschlüsse abzustimmen und den Fortschritt der Teams zu diskutieren. Neben diesen Diskussionspunkten wurden weitere Inhaltsimpulse zu ausgewählten Thematiken vermittelt. Die Inhaltsimpulse wurden vereinzelnd auch durch Gastvorträge gegeben.

Zuletzt war ein Hackathon geplant, welcher Anfang Januar 2022 stattfinden sollte. In Präsenz und unter Einbeziehung des letzten Moduls MSA. Nach mehrtägiger Entwicklungszeit sollten die Roboterkonfigurationen gegeneinander in verschiedenen »Kampf-Sessions« antreten.

Aufgrund längerfristigem krankheitsbedingten Ausfalls der Hälfte der Simulationsbetreuer im November/Dezember 2022, ist jedoch die geplante Service-Testphase gegen Ende der Entwicklungszeit nicht ausreichend umsetzbar gewesen. Daraufhin wurde der Hackathon umgeplant. Die Roboterkonfigurationen konnten hierdurch im Zeitraum vom 03. bis zum 27. Januar durchgeführt werden. Dies ermöglichte den Teams, gefundene Bugs in den Services, beim Testen der eigenen Player zu beheben, ohne gleichzeitig dem Zeitdruck des Hackathons ausgesetzt zu sein.

Beendet wurde das Projekt mit einer Abschlusspräsentation aller Teilnehmer am 29. Januar 2022.

Kapitel 4

Auswertung

Im Folgendem wird das gesamte Projekt analysiert. Bei der Analyse werden vier Themenschwerpunkte gesetzt, beginnend mit dem Endergebnis des Projektes (siehe Kapitel 4.1) und einer Auswertung. In dieser Auswertung wird das Endergebnis mit der anfänglichen Vision verglichen. Weiterhin werden in Kapitel 4.2 die verschiedenen Probleme beleuchtet, welche im gesamten Projektverlauf aufgetreten sind. Daraus werden in Kapitel 4.3 konkrete, auf dieses Projekt beziehbare Optimierungen definiert. Durch diese Erkenntnisse werden abschließend in Kapitel 4.4 einige Vorschläge für Änderungen aufgezeigt und Möglichkeiten beschrieben, wie folgende Iterationen des Projektes oder verwandte Projekte aussehen könnten.

Als Grundlage für diese Analysen dienen besonders zwei Umfragen (siehe Anhänge D, E). Die erste Umfrage wurde zur Mitte des Projektes, in der letzten Novemberwoche durchgeführt. Die zweite Umfrage fand unmittelbar nach Abschluss des Projektes statt. Zusätzlich zu deren Ergebnissen dient die direkte Erfahrung der Lehrenden während des gesamten Projektes als weiterer Einfluss auf die Analyse. Weiterhin wird bei der Voraussicht in Kapitel 4.4 versucht, wo möglich, bisher nicht eingebundene Konzepte und Vorschläge aus Kapitel 2 einzuarbeiten.

4.1 Resultat

Die Vision des Projektes bietet den besten Vergleichswert, wenn man den Stand nach dem Event Storming nutzt. Zu diesem Zeitpunkt stehen die initialen Spielregeln. Die Einteilung der Services hat stattgefunden und der Entwicklungsauftrag für die Teams ist definiert.

Laut des Plans zum Zeitpunkt des Event Stormings sollte die Systemlandschaft mit fünf Services entwickelt werden. Namentlich Map, Robot, Gamelog, Trading und Game. Die Entwicklung dieser Services sollte inklusive einer Testphase gegen Mitte bis Ende Dezember abgeschlossen sein, sodass die Player-Entwicklung im Hackathon im Januar reibungslos stattfinden kann. Der Hackathon sollte in mehreren »Codefights« gipfeln, in welchem die Player und deren Robotersteuerungen gegeneinander antreten. Der

Hackathon ermöglicht dabei neue Teamkonstellationen oder ebenso ein Agieren als Einzelkämpfer. Der Hackathon sollte ebenso wie das Event Storming vor Ort stattfinden, um so den Austausch auf direktem Wege zu ermöglichen.

Die Systemlandschaft hat sich im Vergleich zur Vision nicht verändert. Es mussten keine Serviceschnitte angepasst oder neu definiert werden. Die Entwicklung hat sich jedoch geändert. Die Testphase konnte nicht im Dezember durchgeführt werden. Zum einen waren einzelne Services noch nicht ausgereift genug und wurden erst im Laufe des Dezembers so weit fertiggestellt und zum anderen entfiel die Testung aufgrund krankheitsbedingten Personalmangels.

Im Januar war klar, dass der Hackathon nicht in Präsenz stattfinden konnte, da durch die pandemische Coronaentwicklung mit der Verbreitung der Omikron-Variante eine solche Veranstaltung nicht hätte vertreten werden können. Aufgrund dieses Faktes und der fehlenden Testung der Services wurde die Entwicklung der Player über den gesamten Monat Januar gestreckt. So sollte den Serviceteams der Druck bei der gleichzeitigen Player-Entwicklung genommen werden, welcher ggf. durch aufkommende Bugs im eigenen Service entstehen könnte.

Es wurden zwei Codefights in zweiwöchigem Abstand angestoßen, um den Teilnehmenden zu ermöglichen, die eigenen Player mehrmals zu testen und Problemen auf den Grund zu gehen. Trotz der Verlängerung gelang es den meisten Teilnehmenden nicht, einen Player so weit zu entwickeln, um kompetitiv am Spiel teilnehmen zu können. Dadurch gewann der gleiche Spieler beide Codefights, er war als einziger dazu im imstande sowohl Ressourcen abzubauen als auch gegen anderer Roboter zu kämpfen.

Die meisten anderen Player konnten initial den ersten Roboter erwerben. Einige konnten ihn bereits bewegen, allerdings noch keine Ressourcen abbauen.

Trotz leichter zeitlicher Engpässe gegen Ende des Projektes, wurde das Projekt selbst vom überwiegenden Teil der Teilnehmer als Erfolg gewertet, wie Abbildung 4.1 aufzeigt.

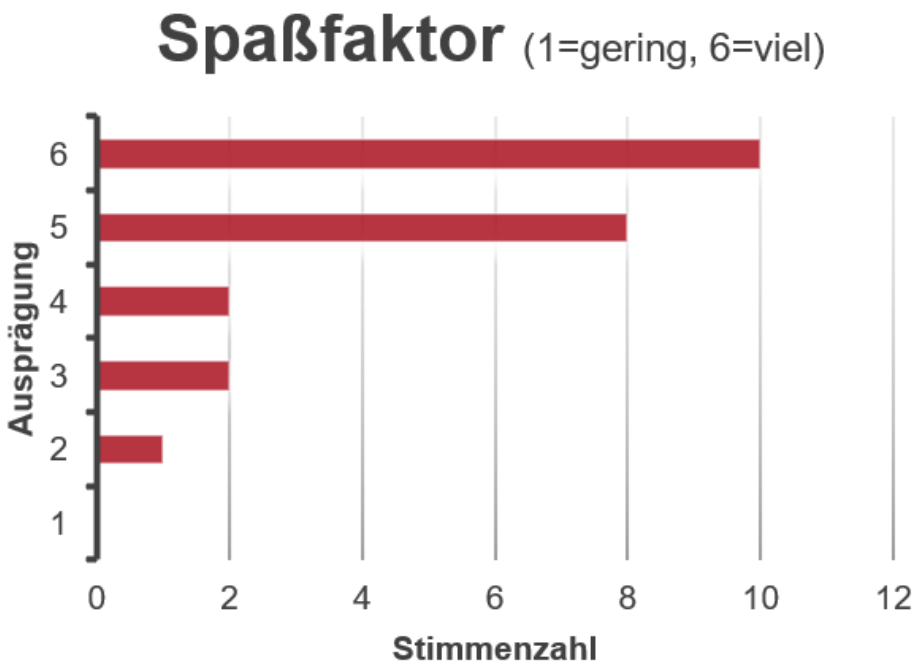


ABBILDUNG 4.1: Spaßfaktor der Teilnehmer des gesamten Projektes [Vgl. E.1.2]

Dies gilt weiterhin für die Punkte der Servicequalität, welche nach den ersten behobenen Bugs weitgehend reibungslos betrieben werden konnten.

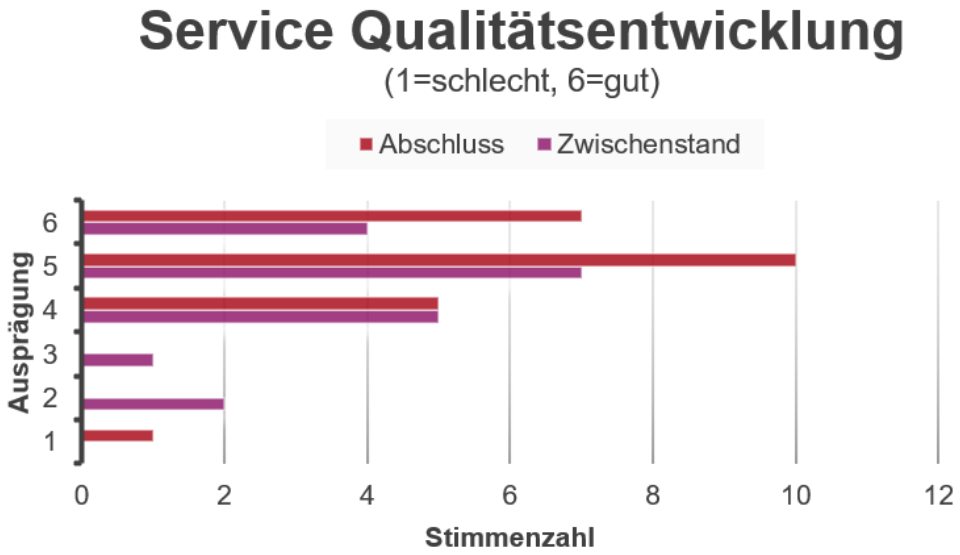


ABBILDUNG 4.2: Qualitätsempfinden der Services im Vergleich [Vgl. D.3.1, E.2.1]

Abbildung 4.2 zeigt die Einschätzung der Servicequalität im Vergleich von Zwischen- und Abschlussumfrage. Bis auf einen Ausreißer wird die Qualität

hoch eingeschätzt. Letztendlich wurden der Codefight und das Gesamtergebnis des Projektes positiv bewertet.

4.2 Probleme

In beiden Umfragen, in Sprintmeetings, sowie in Einzelgesprächen im Verlauf des Projektes, sind einige Problem-Thematiken vermehrt aufgetreten, welche in den folgenden Unterkapiteln beschrieben werden. Die Reihenfolge der Kapitel stellt dabei grob das erste zeitliche Auftreten der Problematiken dar.

4.2.1 Kommunikation

Kommunikation ist ein, wenn nicht das meist angesprochene Thema des Projektes. Sowohl in der Zwischen- als auch in der Abschlussumfrage wurde Kommunikation erwähnt. In offenen Fragen wurde das Thema Kommunikation ebenso positiv wie negativ behandelt. Dabei war es egal, ob der Kontext der Frage bereits »Kommunikation im Projekt« war [D.4.3, E.2.4, E.3.4, E.3.5] oder ob es sich um sonstige »offen gestellte Fragen« handelte [D.2.3, D.2.4, D.3.3, E.3.9, E.5.1, E.5.3, E.5.4]. Kommunikation war somit auch Thematik bei Herausforderungen [E.2.4] im Service oder besonders oft bei Fragen zur Verbesserung der Projektqualität in der Zwischenumfrage [D.3.3].

Eine hervorgehobene Problematik war, dass durch die verschiedenen möglichen Kommunikationswege wichtige Informationen zum Teil nicht alle relevanten Personen erreichten. Im Projekt wurden vier primäre Wege für den Informationsaustausch genutzt:

(1) Discord[1]: direkter Austausch via Instant Messaging für z. B. offene Diskussionen oder organisatorische Hinweise.

(2) Decision Log[2]: Definition und Begründung von Entscheidungen, sowohl für Architektur als auch andere projektrelevante Entscheidungen.

(3) GitHub Readme/Wiki je Service: Relevante Informationen zum Service, bspw. erweiterte Begründungen aus dem Decision Log, »How to run?« Anleitung, Dokumentationsverweise u. a.

(4) Gameplay-Wiki (GitHub Pages): Ursprünglich nur für die Definition des Gameplays durch Project Launch gedacht, später auch für API-Spezifikationen aller Teams genutzt.

Zusätzlich zu den vier genannten Wegen war es jedem Serviceteam freigestellt, für die eigene Aufgabenplanung/-verwaltung eine Technologie der eigenen Wahl zu nutzen. Dadurch entstand eine fünfte Informationsquelle je Team.

[1] Discord ist eine Kommunikationsplattform, welche ursprünglich aus der Gaming-Szene kommt und neben Textkanälen, Voice-Rooms und anderem auch Instant Messaging ermöglicht.

[2] Mehr Details in Kapitel 3.4.2

Während der Player-Entwicklung wurde die serviceeigene Wahl ignoriert und für alle aufkommenden Fehler GitHub-Issues verwendet. Durch diese Wahl war es ebenfalls allen Beteiligten möglich, auch bei fremden Services Fehler einfach zu melden.

Die Anzahl der unterschiedlichen Wege allein zeigt bereits auf, dass Studierende und Lehrende viele Informationsquellen auf einmal im Blick behalten mussten.

Neben der hausgemachten Problematik der unterschiedlichen Informationsquellen gab es auch zwischenmenschliche Kommunikationsschwierigkeiten in mindestens einem Team. Weder den zugeteilten POs/Architekt:innen noch den Lehrenden war dies vor der ersten Ansprache im Teammeeting bekannt, da in allen vorangegangenen Meetings keine Hinweise zu Problemen signalisiert wurden.

4.2.2 Motivation

Für Motivation bzw. die Abwesenheit derselben wurden in beiden Befragungen viele Gründe benannt. Als eine der größten »Spaßbremsen« wurde die Kommunikation beschrieben. [D.2.3, D.2.4] Dabei geht es besonders um die Kommunikation im gesamten Projekt und weniger um die interne Kommunikation in einzelnen Services. Die Kommunikation im Service wird überwiegend gut eingeschätzt [D.4.1]. Im gesamten Projekt wird die Kommunikation jedoch nur durchschnittlich bewertet [D.4.2]. Alle genannten Gründe befinden sich in einer ähnlichen Thematik. Genannt wird hier besonders der Wunsch nach einer größeren Transparenz. Das Silodenken in den einzelnen Services verhindert den Austausch über die Teamgrenzen. Der Wunsch nach direkten Absprachen und Problemaustausch konnte besonders bis zur Zwischenumfrage nicht umgesetzt werden.

Neben der Kommunikation wurde eine weitere Thematik mehrfach und über beide Befragungen hinweg genannt: Der zu große Projektumfang. [D.2.3, E.2.4, E.3.8] Der Wunsch liegt in der Reduktion des Umfangs, um den Fokus auf größere Programmierqualität und Umgang mit der Materie zu lenken.

4.2.3 Projektqualität

Die Servicesteams selbst sind großteils zufrieden mit der Qualität des eigenen Produkts. [D.3.1] Dennoch sind einige spezifische Probleme entstanden und genannt worden, welche im Anschluss aufgeführt werden.

Eine konstante Mehrfachnennung bezog sich auf Tests der Services und die fehlende Abdeckung. [D.3.3] Die fehlende Testabdeckung wurde durch die fehlende Kommunikation mit anderen Services verursacht. APIs verbundener Services unterlagen leichten Änderungen, wodurch die Entwicklung von Tests gegen eine Spezifikation lange nicht möglich war.

Aus diesem Hintergrund wurde mehrmals der Wunsch genannt, feste Spezifikationen bereits zu Beginn des Projekts zu definieren und nicht mehr zu

verändern. Ein verständlicher Wunsch, allerdings war die Änderung von Anforderungen während des Projektverlaufes angelehnt an entsprechende Bedingungen in reellen Softwareprojekten und wurde daher nicht als aktive Problematik aus Sicht der Lehrenden betrachtet, sondern als Herausforderung, die es zu meistern gilt. [C.2.7 Starke 2]

Ein Vortrag der Abschlusspräsentation erweiterte den Punkt der festen Spezifikationen mit dem Vorschlag einer festen Meilensteinplanung zu Beginn des Projektes. Einem Vorgang, welcher eher eine Projektplanung nach dem Wasserfallmodell widerspiegelt und keine agile Arbeitsweise, wie sie in diesem Projekt angewendet werden sollte.

4.2.4 Player Entwicklung

Eine Problematik, welche erst im letzten Monat des Projektes aufgekommen ist. Die Playerentwicklung wurde sowohl von den Studierenden als auch den Lehrenden, von ihrer Komplexität her unterschätzt.

Durch die verlängerte Entwicklungsphase der Player konnten sich zwar einige Roboter im letzten Codefight bewegen, aber ein kompetitives Spiel konnte noch nicht durchgeführt werden.

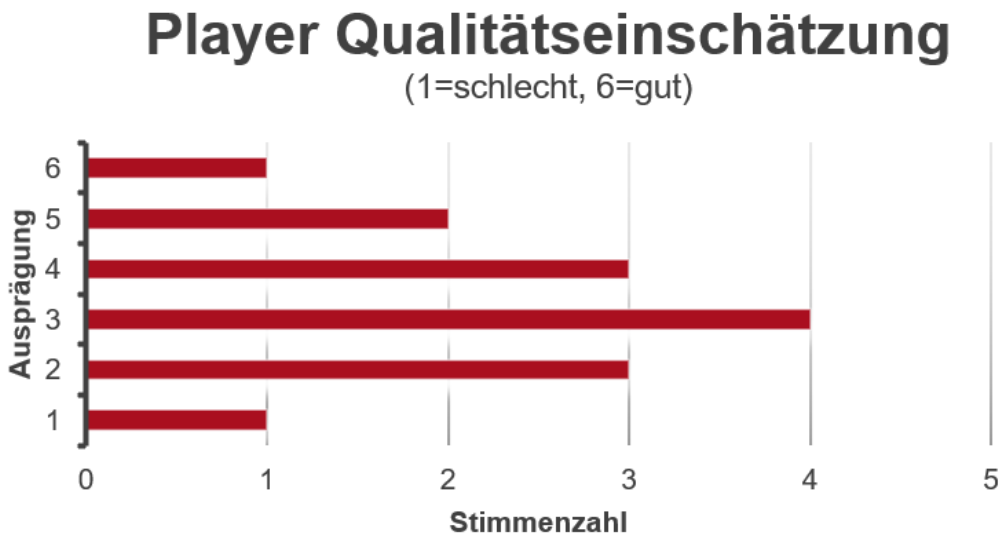


ABBILDUNG 4.3: Qualitätsempfinden der eigenen Player [Vgl. E.2.5]

Dies spiegelt sich auch in der Einschätzung der eigenen Playerqualität in der Abschlussumfrage wider, wie Abbildung 4.3 zeigt.

4.3 Lessons Learned

Die Problematiken, die während dieses Projektes entstanden, geben wertvolle Hinweise für künftige Planungen. Hieraus ergeben sich interessante Ansätze für Folgeprojekte.

Eine der Lehren, die aus dem Projekt geschlossen werden kann, ist die Reduktion der Komplexität der Thematik. Die Komplexität des zugrundeliegenden Spielkonzepts sollte so weit wie möglich reduziert werden, sodass die Services bereits in einer früheren Phase vollständig deployt und das System getestet werden kann. „[...] kompliziert wird es von ganz alleine, da muss man nicht am Anfang schon für sorgen." [C.2.6 Wörzberger (1)] Bei der Gameplay-Planung muss nicht jegliche Komplexität entfallen. Das Gameplay sollte jedoch aufeinander aufbauend gestaltet werden, sodass die Entwicklung eines MVP[3] ermöglicht wird, bevor weitere Komplexität durch weitere Gameplayfeatures eingebaut wird.

Ein konkretes Beispiel am Projekt könnte bedeuten, dass sich in der ersten Iteration die Roboter nur bewegen können sollen und alle komplexeren Systeme wie Upgrades, Items, Kampf und Mining erst folgend hinzukommen oder komplett wegfallen.

Ein Vorschlag, die Kommunikation zu verbessern, war „häufigere, dafür kürzere Meetings" [D.4.3] dafür zu nutzen, die Feedbackschleifen zu verkürzen und Nachfragen gezielter stellen zu können.

Eine Möglichkeit dies umzusetzen ist, die Einführung fester Arbeitstage für einen Teil der Projektteilnehmer. Dies wäre z. B. für die Teilnehmer des Informatikprojektes möglich. Mit festen Zeiten, in welchen an allen Services gearbeitet wird, könnten so unterschiedliche Stand-up[4]-Zeiten für alle Teams festgelegt werden. Weiterhin könnte so der agile Prozess weiter im Projekt etabliert werden. Durch Bekanntgabe der Stand-up-Zeiten aller Teams eröffnet sich die Möglichkeit, bei Gesprächsbedarf, dem Stand-up eines anderen Teams beizuwohnen oder im Anschluss daran Fragen zu stellen. Die Anwesenheit der kritischen Personen ist somit gewährleistet und die Öffentlichkeit sorgt für mehr Transparenz für aktuelle Probleme und Thematiken.

Die Transparenz sollte weiterhin geschaffen werden, indem die Technologiewahl der Teams auf organisatorischer Ebene leicht eingeschränkt wird. Zumindest für die Aufgabenplanung sollte das gewählte Tooling projektweit identisch sein. Somit werden allen Teams aktuelle Problematiken verdeutlicht. Außerdem können über diese Wege Probleme gemeldet werden, wie es in der abschließenden Phase des Projektes über die GitHub Issues bereits geschehen ist.

[3] Minimum Viable Product

[4] Regelmäßige, zumeist tägliches Meeting mit beschränkter Zeit, in welchem alle Teilnehmer kurz über ihre aktuelle Arbeit sprechen.

Als letzte große Erkenntnis kann das »local development environment« Skript bezeichnet werden. Das Skript wurde in der Endphase des Projektes entworfen und ermöglicht es den Studierenden, die Services auf der lokalen Entwicklungsmaschine zu starten. Die Services werden dabei von den aktuellen Images auf GitHub gebaut und lokal mittels Docker deployt. Eine Maßnahme, die nötig war, um die Playerentwicklung und Service-Debugging in der Endphase des Projektes voranzutreiben. In einem Greenfield-Projekt, in welchem die API-Spezifikationen auch nach einiger Entwicklungszeit leichten Änderungen unterliegen, ist die Nutzung eines solchen Toolings nötig. Fehlerquellen können sonst nicht valide nachverfolgt werden. Den Studierenden wurde so weiterhin vermittelt, wie wichtig es ist zu verinnerlichen: „Remote ist nicht Local“ und offen für die Fragen zu sein: „[...] Wo sind die Unterschiede zwischen Remote und Local? Was gibt es für zusätzliche Fehlerquellen?“. [C.2.6 Starke (3)]

Durch die Nutzung des Skripts konnte weiterhin die Interaktion zwischen den Studierenden gefördert werden. Gegenseitiges Feedback, Hinweise auf Fehler sowie Lösungen wurden vermehrt geteilt.

4.4 Projektiteration

Alle Erkenntnisse aus der Simulation ergeben viele Möglichkeiten, wie das Projekt weitergeführt werden kann bzw. neue Projekte um die Thematik entstehen können. Folgend wird damit die zweite Forschungsfrage »Wie muss eine Simulation gestaltet werden, um die festgestellten Kompetenzen erfolgreich vermitteln zu können?«, behandelt und beantwortet.

4.4.1 Weiterentwicklung

Mit einer Weiterentwicklung des aktuellen Stands lassen sich zwar nicht alle Lehren direkt auf Folgeprojekte übertragen, jedoch entstehen viele Vorteile, welche zur Lehre von lose gekoppelten Systemen und der Vermittlung von Qualifikationen von Softwarearchitekt:innen genutzt werden können.

Der größte und offensichtlichste Punkt ist der Gewinn einer Brownfield-Umgebung[5]. Unter den Teilnehmenden der Simulation wurde bereits Interesse bekundet, den Microservice Dungeon über weitere Projekte und Abschlussarbeiten zu begleiten. Es ist somit möglich, die Weiterentwicklung mit sowohl neuen als auch bekannten Entwicklern zu gestalten und so reelle Bedingungen noch besser darzustellen.

Als Motivationsfaktor kommt somit für alle Beteiligten hinzu, dass die entwickelte Software keine einfache Lehrabgabe ist, welche nach der Abgabe nicht mehr benötigt wird.

[5]Brownfield-Umgebung bedeutet die Entwickelung neuer Systeme und Features in einer bestehenden Systemlandschaft. Im Gegensatz zum Greenfield, der Neuentwicklung ohne strukturelle Rahmenbedingungen, müssen meist aktiv genutzte Services bedacht werden.

Bei der Weiterentwicklung wurden durch die Abschlussumfrage und weiterführenden Diskussionen bereits einige Schlüsselpunkte identifiziert, welche folgend beschrieben werden.

DevEnv

Das DevOps Team hat in seiner Rolle den Entwicklern der einzelnen Services viele Aufgaben in Deployment und Verwaltung des Services auf dem Hostsystem abgenommen. Eine Übergabe dieser Aufgaben an die Serviceteams selbst konnte aufgrund der Zeitknappheit während der Player-Entwicklung nicht stattfinden.

Aus Sicht der Lehrenden ist für das Betreiben der Infrastruktur die Verfolgung eines Zero-Admin-Ansatzes wünschenswert. Den Entwicklern soll es somit möglich sein, in einer Testumgebung ihren eigenen Service komplett zu verwalten. Dadurch wären manuelle Redeployments oder Datenbankwipes für Entwicklungszwecke ohne administrativen Aufwand Unbeteiligter möglich. Des Weiteren wurde während der Player-Entwicklung eine lokale Entwicklungsumgebung[6] bereitgestellt. Dabei handelt es sich um ein Pythonskript, welches es lokal ermöglicht, die Microserviceumgebung anhand der aktuellen Builds aller Services zu starten. Die Weiterentwicklung und Optimierung auf Erweiterbarkeit ist auch ein valides Ziel für die nächste Iteration.

Spielregeln Simplifizieren

Die Spielregeln erfordern teilweise komplexe technische Umsetzungen in den Services, welche die Erstellung von Playern erheblich erschwert. Eine Vereinfachung der Spielregeln und ein passender technischer Rückbau des Gesamtsystems, sollte eine der ersten Anpassungen darstellen.

Weiterhin wurden einige Spielregeln bereits während der Entwicklung vereinfach, aber noch nicht im Gameplay-Wiki angepasst. Der Rückbau muss einhergehen mit der Pflege der Gameplay-Regeln.

Das Feature, welches für die meiste Komplexität gesorgt hat, ist »Fog of War«. Die Spielmechanik beinhaltet, dass die Player nur Informationen erhalten dürfen, wenn eigene Roboter diese »sehen« können. Eine Mechanik, welche dem asynchronen Ansatz direkt widerspricht, in welchem Informationen breit zugänglich gemacht werden und die Services selbst nicht wissen, wer diese Informationen nutzt. Die Player selbst sind jedoch Teil der Microservicelandschaft, damit auch diese alle Aspekte der Servicekommunikation implementieren müssen.

Fog of War wurde daher künstlich über Verschleierung umgesetzt. Die initiale Anfrage der Spieler richtet sich synchron an den Game-Service, da dieser

[6] GitHub-Repository: https://github.com/The-Microservice-Dungeon/local-dev-environment

die Autorisierung übernimmt. Bei der Anfrage wird eine sogenannte Transaktion-ID erzeugt, welche nur der Spieler selbst kennt. Alle durch diese Anfrage erzeugten Events beziehen sich auf die Transaktion-ID und geben teilw. nicht die Spieler- oder Robot-ID zurück. So wird ermöglicht, dass nur der Spieler selbst das volle Bild kennt.

Die so entstehende Komplexität innerhalb des Players kann durch den Verzicht auf Fog of War entfallen.

4.4.2 Neubeginn

Eine weitere Möglichkeit der Gestaltung eines neuen Projektes ist es, neu in einer Greenfield-Umgebung anzufangen und nur die gewonnenen Erkenntnisse mit einzubeziehen. Ein großer Teil wird davon bereits in Kapitel 4.3 Lessons Learned aufgezeigt und daher folgend teilw. kürzer gefasst oder wie auch Kommunikation, nicht erwähnt.

Gameplay
Die Spielregeln sollten von Beginn an möglichst einfach aufgebaut werden. Komplexität kann über einen modularen Aufbau der Spielregeln mit eingeplant werden. Das Spiel muss allerdings bereits in der einfachsten Form spielbar sein. Dies ermöglicht in der Entwicklung den Fokus für die Umsetzung eines MVP.

Player-Entwicklung
Die Player-Entwicklung ohne tiefgreifende Grundlage erfordert einen großen Entwicklungsaufwand gegen Ende der Projektzeit. Mit der Einteilung eines eigenen Teams für die Entwicklung eines Player-Templates können die Grundmechaniken für einen Hackathon bereitgestellt werden. Die Entwicklung bietet weiterhin den Vorteil, dass die Services aktiv durch den Player getestet werden können. Der Fokus im Hackathon kann vermehrt auf der Implementierung von weiterreichenden Features und Taktiken liegen und nicht schlicht darauf, alles irgendwie lauffähig zu gestalten. Der Gamification-Aspekt kann dadurch gefördert werden.

Services
Die Einteilung der Services hat weitestgehend gut funktioniert. Ein Punkt, der selbst keine weitgehende Veränderung benötigt und sowieso vom studentischen Input beim Event Storming abhängig ist. Dieser Punkt kann nur indirekt von außen durch die Lehrenden manipuliert werden.

Weitere Probleme in den Services, wie unterschiedliches Niveau bspw. beim Programmieren, sollte kein Kriterium sein, wie es teilweise bereits beschrieben wurde. [E.5.1] Einen gewissen Unterschied im Niveau wird es immer geben und ihn zu überwinden bspw. durch Teamarbeit, zeigt Kompetenz und kann Softskills fördern.

4.4.3 Monolith

Eine weitere Möglichkeit, die Thematik des Projekts weiterzuführen, wäre den aktuellen Stand zu nutzen und in einen Monolithen zu überführen. Bei solch kleinen Anwendungsfällen ist es bspw. bei REWE Digital Praxis, für die Erstentwicklung mit einem Monolithen zu starten, da ein Start direkt mit Microservices aus Unternehmenssicht zu aufwendig und teuer ist. [C.2.7 Gauder (2)]

Die Monolith-Option bietet dabei die Möglichkeit der Entwicklung durch ein kleineres, ggf. neues Team. In diesem Schritt sollten die in Kapitel 4.3 Lessons Learned beschriebenen Aspekte des Gameplays ebenfalls angepasst werden. Nach der eigentlichen Entwicklung wird eine umfangreiche Analyse ermöglicht, mit deren Hilfe herausgefunden werden kann, welche Aspekte im konkreten Beispiel für und gegen Microservices sprechen.

Der größte Vorteil, welcher durch einen Monolithen hergestellt wird, ist die Möglichkeit weiterer Folgeprojekte. Microservices entstehen oft als Brownfield-Projekte und sollen alte Monolithe ablösen. Dies kann erst erfolgreich simuliert werden, wenn ein Monolith besteht.

Kapitel 5

Fazit

Das Ziel dieser Master Thesis war es, die beiden gestellten Forschungsfragen zu beantworten:

(1) Welche Erfahrungen müssen Lernende in der Softwarearchitektur machen, um die Notwendigkeit von modernen und lose gekoppelten Architekturen verstehen und umsetzen zu können?

(2) Wie muss eine Simulation gestaltet werden, um die festgestellten Kompetenzen erfolgreich vermitteln zu können?

Beide Fragen konnten im Verlauf der Arbeit beantwortet werden. Die erste Frage wurde in Kapitel 2 erfolgreich über die drei Säulen beantwortet:

> **Arbeitsmarktanalyse**: Welche Erfahrungen und Kompetenzen werden auf dem Arbeitsmarkt nachgefragt?
>
> **Analyse der Hochschullehre**: Welche Angebote werden an den Hochschulen in unterschiedlichen Modulen zum Erwerb notwendiger Kompetenzen gemacht?
>
> **Expertenbefragung**: Welche Erfahrungen haben Experten in ihrer beruflichen Tätigkeit gemacht? Welche grundlegenden Kenntnisse und Skills sollten Neueinsteiger ins Berufsleben mitbringen?

Die ersten beide Analysen brachten dabei weniger umfassende Erkenntnisse als erwartet. Der größte Teil der Erkenntnisse konnte aus den Expertenbefragungen gezogen werden. Besonders interessant dabei war das Ergebnis der enorm großen Bedeutung von Softskills im Architekt:innenberuf. Ein Fakt, welcher durch die Vorüberlegungen und Recherchen nicht so deutlich erwartet wurde.

Eine Fortsetzung dieser Arbeit könnte mit dem Fokus auf einer Verbreiterung des Recherchefundaments von Arbeitsmarkts- und Hochschullehrenanalyse durchgeführt werden.

Die zweite Forschungsfrage wurde anhand des begleitenden Projektes untersucht. Das Konzept wurde teilweise vor Beendigung aller Interviews und Analysen entworfen, wodurch nicht alle in dieser Thesis verfassten Erkenntnisse direkt mit einfließen konnten. Dennoch wurde der Großteil abgedeckt.

Die Simulation, welche unter der Bezeichnung des »Microservice Dungeon« Projektes durchgeführt wurde, musste kleine Rückschläge in Kauf nehmen. Zum einen führte eine mehrwöchige krankheitsbedingte Abwesenheit einer Lehrkraft dazu, dass wichtige Unterstützung der Studierenden nicht im geplanten Umfang geleistet werden konnte und zum anderen hat die »Omikron-Welle« zum Winter 2021/22 weitere geplante Präsenzveranstaltungen aller Teilnehmenden verhindert. Bedingt hierdurch ist bspw. der geplante Hackathon im Januar 2022 umgeplant worden. Diesen Widrigkeiten zum Trotz konnte das Projekt erfolgreich abgeschlossen werden. Trotz fehlender Zeit am Ende des Projektes konnten die ersten Player-Entwicklungen am Spielgeschehen teilnehmen. Besonders ein Teilnehmer konnte in jedem Durchgang durch seine fortgeschrittene Entwicklung gewinnen. Für eine Wiederholung einer Spielrunde mit dann mehreren kompetitiven Playern ist bereits ein Termin mit allen freiwilligen und begeisterten Projektteilnehmer:innen vereinbart.

Besonders im Projekt gelungen ist das Backend-lastige Konzept. Die starke Backend-Orientierung bringt eine große Arbeitslast für die Studierenden mit sich, birgt jedoch auch für die Lehrenden die Möglichkeit tiefgreifende Konzepte mit einzubinden und zu vermitteln.

Die Auswertung (siehe Kapitel 4) zeigt besonders durch die Umfragen, dass die Lehrziele erreicht werden konnten. Der Fokus auf Kommunikation und deren Problematik wurde oftmals aus Sicht der Studierenden positiv wie negativ diskutiert, ohne dass dies durch die Lehrenden angestoßen werden musste.

Mit den ermittelten Qualifikationen und Erkenntnissen der Simulation können zukünftig weitere, noch genauer zugeschnittene Projekte gestaltet werden, um den Studierenden die bestmöglichen Grundlagen für die spätere Berufswelt zu bieten.

Literatur

[1] Jeremy Hillpot. *4 Microservices Examples: Amazon, Netflix, Uber, and Etsy*. Letzte Sichtung 25. Januar 2022. Juli 2021. URL: https://blog.dreamfactory.com/microservices-examples/.

[2] Matthias Naab. *Was macht ein Software Architekt? @ONLINE*. Letzte Sichtung 27. Dezember 2021. Juli 2019. URL: https://www.iese.fraunhofer.de/blog/was-ist-eigentlich-ein-softwarearchitekt/.

[3] Christiane Schäfter und Markus Ley. „Manche Stellenausschreibungen lesen sich wie die berühmte Suche nach der eierlegenden Wollmilchsau“. In: *Wirtschaftsinformatik & Management* (2018), S. 8–11. URL: https://doi.org/10.1007/s35764-018-0001-5.

[4] Olaf Zukunft. *Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016)*. 2016.

[5] Renato Cordeiro, Thatiane Rosa, Alfredo Goldman und Eduardo Guerra. „Teaching Complex Systems based on Microservices“. In: *GROUP* 1 (), S. 1.

[6] ItS Initiative für transparente Studienförderung gemeinnützige UG. *Uni Ranking Informatik @ONLINE*. Letzte Sichtung 07. Februar 2022. 2022. URL: https://www.mystipendium.de/hochschulen/rankings/uni-ranking-informatik.

[7] René Wörzberger. *Biography @ONLINE*. Letzte Sichtung 13. Januar 2022. 2022. URL: https://rene.woerzberger.de/.

[8] Thoughtworks. *Unsere Geschichte @ONLINE*. Letzte Sichtung 13. Januar 2022. 2022. URL: https://www.thoughtworks.com/de-de/about-us.

[9] Stefan Tilkov. *Biography @ONLINE*. Letzte Sichtung 13. Januar 2022. 2022. URL: https://www.innoq.com/de/blog/rollen-bei-innoq/.

[10] iSAQB e. V. *Über Uns. Der Verein @ONLINE*. Letzte Sichtung 13. Februar 2022. 2022. URL: https://www.isaqb.org/de/ueber-uns/ueber-uns/#about-isaqb.

[11] Gernot Starke. *Dr. Gernot Starke @ONLINE*. Letzte Sichtung 13. Januar 2022. 2022. URL: https://www.gernotstarke.de/about/.

[12] Sam Newman. *Microservices: Konzeption und design*. MITP-Verlags GmbH & Co. KG, 2015.

[13] Oona Hämäläinen. „API-First Design with Modern Tools“. In: (2019).

[14] Alexis Henry und Youssef Ridene. „Migrating to microservices“. In: *Microservices*. Springer, 2020, S. 45–72.

[15] Wikipedia. *Didaktik @ONLINE*. Letzte Sichtung 13. Februar 2022. 2021. URL: https://de.wikipedia.org/wiki/Didaktik.

[16] Peter Fenske. *Das kleine Buch vom Lernen*. 18. Aufl. ISBN 3-89312-411-X. AOL-Verlag, 2006.

[17] Falko Rheinberg und Regina Vollmeyer. *Motivation*. Kohlhammer Verlag, 2018.

[18] Edward L Deci und Richard M Ryan. „Die Selbstbestimmungstheorie der Motivation und ihre Bedeutung für die Pädagogik“. In: *Zeitschrift für Pädagogik* 39.2 (1993), S. 223–238.

[19] Falko Rheinberg. „Intrinsische Motivation und Flow-Erleben“. In: *J. Heckhausen & H. Heckhausen (Hrsg.), Motivation und Handeln* 3 (2004). URL: http://www.psych.uni-potsdam.de/people/rheinberg/files/Intrinsische-Motivation.pdf.

[20] Walter Edelmann. „Intrinsische und extrinsische Motivation“. In: *Grundschule* 4 (2003), S. 30–32.

[21] Team Asana. *Was ist intrinsische Motivation und wie funktioniert sie? @ONLINE*. Letzte Sichtung 10. Februar 2022. 2021. URL: https://asana.com/de/resources/intrinsic-motivation.

[22] Bernhard Schlag. „Intrinsische und extrinsische Motivation“. In: *Lern- und Leistungsmotivation*. Springer, 2013, S. 21–26.

[23] Sebastian Deterding, Rilla Khaled, Lennart E Nacke, Dan Dixon u. a. „Gamification: Toward a definition“. In: *CHI 2011 gamification workshop proceedings*. Bd. 12. Vancouver, Canadá. 2011, S. 12–15.

[24] Yu-kai Chou. *Actionable gamification: Beyond points, badges, and leaderboards*. Packt Publishing Ltd, 2019.

[25] Isabel John. „Gamification in der Software Engineering Lehre–ein Erfahrungsbericht“. In: (2020).

[26] Bitkom. *Anteil der Computer- und Videospieler in verschiedenen Altersgruppen in Deutschland im Jahr 2021 [Graph]. In Statista. Statista GmbH.* Letzte Sichtung 04. Januar 2022. 2021. URL: https://de.statista.com/statistik/daten/studie/315924/umfrage/anteil-der-computerspieler-in-deutschland-nach-alter/.

[27] Alberto Brandolini. *Introducing EventStorming: An act of Deliberate Collective Learning*. Leanpub, 2021.

[28] Melvin E. Conway. „How Do Committees Invent?“ In: *Datamation magazine* (1968). URL: https://www.melconway.com/Home/Committees_Paper.html.

[29] TH Köln. *Code & Context (B.Sc) @ONLINE*. Letzte Sichtung 02. Januar 2022. 2022. URL: https://www.th-koeln.de/studium/code--context-bachelor_62103.php.

[30] TH Köln. *Modul »Project Launch« (PL1) @ONLINE*. Letzte Sichtung 02. Januar 2022. 2022. URL: https://coco.study/module/530-project-launch-1/.

[31] TH Köln. *Modul: Informatikprojekt @ONLINE*. Letzte Sichtung 02. Januar 2022. 2022. URL: https://fhpwww.gm.fh-koeln.de/hops/modules/modulelisting/module.php?mkz=1623.

[32] Stefan Bente. *Modul »Domain-Driven Design of Large Software Systems« (DDD) @ONLINE*. Letzte Sichtung 02. Januar 2022. 2022. URL: https://digital-sciences.de/modules/domain-driven-design/.

[33] TH Köln. *Kurs »Microservice Architectures« (DT11) @ONLINE*. Letzte Sichtung 02. Januar 2022. 2022. URL: https://coco.study/kurse/310-developing-things-1/microservice-architectures/.

Abbildungsverzeichnis

Anhang A

Stellenanzeigen

Für die Analyse des Arbeitsmarktes wurden 59 Stellenanzeigen von 49 unterschiedlichen Unternehmen aus über 20 verschiedenen Branchen analysiert.

Die Stellenanzeigen wurden dabei aus acht unterschiedlichen Job-Portalen sowie direkt von Unternehmensanzeigen bezogen. Aufgrund der Schnelllebigkeit auf den genannten Portalen sind bei der Recherche nicht die direkten Verlinkungen gesichert worden. Stattdessen wurden zum einen die eigentlichen Inhaltstexte der Aufgaben und Profilbeschreibungen gesichert sowie die Quellen mittels web.archive.org persistiert.

Die folgende Liste enthält die Meta-Informationen aller gesammelten und analysierten Stellenanzeigen.

Sick AG (Elektrotechnik, Feinmechanik & Optik)
Titel: Softwarearchitekt* / Software Engineer* Embedded
Anzeige bei Stepstone am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220204090615/https://www.stepstone.de/stellenangebote--Softwarearchitekt-Software-Engineer-Embedded-Waldkirch-bei-Freiburg-im-Breisgau-SICK-AG--7922792-inline.html?rltr=3_3_25_seorl_m_0_0_0_0_0_0

Volkswagen AG (Fahrzeugbau/-zulieferer)
Titel: Softwarearchitekt (m/w/d) – MAGIC Communication Middleware
Anzeige bei Stepstone am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220204090714/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-MAGIC-Communication-Middleware-Bochum-Wolfsburg-Volkswagen-Infotainment-GmbH--7774220-inline.html?rltr=1_1_25_seorl_m_0_0_0_0_0_0

Bosch Rexroth (Hydraulik Zulieferer)
Titel: Softwarearchitekt (m/w/div.)
Anzeige bei Stepstone am 03.02.2022
Archivierte Quelle: https://web.archive.org/web/20220205195522/https://www.stepstone.de/stellenangebote--Software-Architekt-w-m-div-Lohr-am-Main-Bosch-Rexroth--7448545-inline.html

---

KARL STORZ SE & Co. KG (Medizintechnik)
Titel: Softwarearchitekt (m/w/div.)
Anzeige bei Stepstone am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220204091352/https://www.stepstone.de/stellenangebote--Software-Architekt-m-w-d-Schaffhausen-CH-oder-Tuttlingen-KARL-STORZ-SE-Co-KG--7640912-inline.html?rltr=4_4_25_seorl_m_0_0_0_0_1_0

SSI Schäfer IT Solutions GmbH (Lager- und Logistiksystemen)
Titel: Softwarearchitekt (m/w/div.)
Anzeige bei Stepstone am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205113656/https://www.stepstone.de/stellenangebote--Software-Architekt-w-m-d-Dortmund-Oberviechtach-Regensburg-SSI-Schaefer-IT-Solutions-GmbH--7786951-inline.html

---

Dataport AöR (IT & Internet)
Titel: Software-Architekt digitale Bildungslösungen (w/m/d)
Anzeige bei Stepstone am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205111902/https://www.stepstone.de/stellenangebote--Software-Architekt-digitale-Bildungsloesungen-w-m-d-Altenholz-Bremen-Halle-Saale-Hamburg-Magdeburg-Rostock-Potsdam-Dataport-AoeR--7924123-inline.html

---

Bosch Gruppe (Fahrzeugbau/-zulieferer IT & Internet Konsumgüter/ Gebrauchsgüter)
Titel: Softwarearchitekt (m/w/div.)
Anzeige bei Stepstone am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205114600/https://www.stepstone.de/stellenangebote--Software-Architekt-w-m-div-Hildesheim-Bosch-Gruppe--7441066-inline.html

---

Bosch Gruppe (Fahrzeugbau/-zulieferer IT & Internet Konsumgüter/ Gebrauchsgüter)
Titel: Software-Architekt virtuelle Steuergeräte im Bereich Automatisiertes Fahren (w/m/div.)
Anzeige bei Stepstone am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205114913/https://www.stepstone.de/stellenangebote--Software-Architekt-virtuelle-Steuergeraete-im-Bereich-Automatisiertes-Fahren-w-m-div-Abstatt-Bosch-Gruppe--7924749-inline.html

---

ARRK Engineering GmbH (Fahrzeugzulieferer)
Titel: Softwarearchitekt (m/w/div.)
Anzeige bei Stepstone am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205115512/https://www.stepstone.de/stellenangebote--Software-Architekt-m-w-d-Muenchen-ARRK-Engineering-GmbH--7786519-inline.html

---

TQ-Systems GmbH (Elektrotechnik, Feinmechanik & Optik Medizintechnik Luft- und Raumfahrt)
Titel: Softwarearchitekt E-Mobility (m/w/d)
Anzeige bei Stepstone am 27.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205115910/https://www.stepstone.de/stellenangebote--Softwarearchitekt-E-Mobility-m-w-d-Inning-bei-Muenchen-TQ-Systems-GmbH--7864283-inline.html

---

Thüga SmartService GmbH (IT & Internet Sonstige Dienstleistungen Telekommunikation)
Titel: Informatiker als IT Architekt / Software Architekt Energiebranche (m/w/d)
Anzeige bei Stepstone am 27.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205120435/https://www.stepstone.de/stellenangebote--Informatiker-als-IT-Architekt-Software-Architekt-Energiebranche-m-w-d-Muenchen-Freiburg-Naila-Thuega-SmartService-GmbH--7781554-inline.html

---

Zeiss (Elektrotechnik, Feinmechanik & Optik)
Titel: Software Architekt (m/w/x)
Anzeige bei Stepstone am 27.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205120853/https://www.stepstone.de/stellenangebote--Software-Architekt-m-w-x-Oberkochen-Baden-Wuerttemberg-ZEISS--7556251-inline.html

---

Dataport AöR (IT & Internet)
Titel: Softwarearchitekt Web, .NET Core, Containerumfeld (w/m/d)
Anzeige bei Stepstone am 27.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205122659/https://www.stepstone.de/stellenangebote--Softwarearchitekt-Web-NET-Core-Containerumfeld-w-m-d-Halle-Saale-Magdeburg-Kiel-Bremen-Hamburg-Rostock-Dataport-AoeR--7862954-inline.html

---

Optica Abrechnungszentrum Dr. Güldener GmbH (Finanzdienstleister)
Titel: Softwarearchitekt / Senior Software-Entwickler (m/w/d)
Anzeige bei Stepstone am 26.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205123010/https://www.stepstone.de/stellenangebote--Softwarearchitekt-Senior-Software-Entwickler-m-w-d-Stuttgart-Optica-Abrechnungszentrum-Dr-Gueldener-GmbH--7551069-inline.html

---

Porsche AG (Fahrzeugbau/-zulieferer)
Titel: Softwarearchitekt (m/w/d) Porsche Conneced Car
Anzeige bei Stepstone am 26.01.2022
Archivierte Quelle: https://web.archive.org/web/20220204093407/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-Porsche-Conneced-Car-Weissach-Porsche-AG--7916508-inline.html?rltr=22_22_25_seorl_m_0_0_0_0_0_0

Deutsche Bahn AG (Sonstige Branchen Transport & Logistik)
Titel: Softwarearchitekt / Lead Softwareentwickler im DevOps Team (w/-m/d)
Anzeige bei Stepstone am 26.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205123323/https://www.stepstone.de/stellenangebote--Softwarearchitekt-Lead-Softwareentwickler-im-DevOps-Team-w-m-d-Berlin-Frankfurt-Main-Deutsche-Bahn-AG--7858071-inline.html

---

Hensoldt (Elektrotechnik, Feinmechanik & Optik Luft- und Raumfahrt Metallindustrie)
Titel: Softwarearchitekt*in FCAS Simulationsanwendungen (w/m/d)
Anzeige bei Stepstone am 26.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205123525/https://www.stepstone.de/stellenangebote--Softwarearchitektin-FCAS-Simulationsanwendungen-w-m-d-Immenstaad-HENSOLDT--7857653-inline.html

---

paragon GmbH & Co. KGaA (Fahrzeugzulieferer)
Titel: Softwarearchitekt (m/w/d) für Automotive Embedded Entwicklung
Anzeige bei Stepstone am 25.01.2022
Archivierte Quelle: https://web.archive.org/web/20220204091903/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-fuer-Automotive-Embedded-Entwicklung-Limbach-paragon-GmbH-Co-KGaA--7902526-inline.html?rltr=37_12_25_seorl_m_0_0_0_0_0_0

Leopold Kostal GmbH & Co. KG (Fahrzeugbau/-zulieferer Elektrotechnik, Feinmechanik & Optik Sonstiges produzierendes Gewerbe)
Titel: Software Architekt (m/w/d) für Elektromobilität
Anzeige bei Stepstone am 25.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205123817/https://www.stepstone.de/stellenangebote--Software-Architekt-m-w-d-fuer-Elektromobilitaet-Dortmund-Leopold-Kostal-GmbH-Co-KG--7854692-inline.html

---

EDEKA DIGITAL GmbH (IT & Internet)
Titel: Cloud Software Architekt (m/w/d)
Anzeige bei Stepstone am 25.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205124108/https://www.stepstone.de/stellenangebote--Cloud-Software-Architekt-m-w-d-Hamburg-EDEKA-DIGITAL-GmbH--7909453-inline.html

---

R+V Allgemeine Versicherung AG (Versicherungen)
Titel: Softwarearchitekt (m/w/d) IAM / Cloud-Native
Anzeige bei Stepstone am 25.01.2022
Archivierte Quelle: https://web.archive.org/save/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-IAM-Cloud-Native-Wiesbaden-R-V-Allgemeine-Versicherung-AG--7051753-inline.html?rltr=44_19_25_seorl_m_0_0_0_0_0_0

Zeiss (Elektrotechnik, Feinmechanik)
Titel: Softwarearchitekt | C++ | Industrielle Messtechnik | Informatik / Naturwissenschaft / Ingenieurwesen (m/w/x)
Anzeige bei Stepstone am 24.01.2022
Archivierte Quelle: https://web.archive.org/web/20220204092547/https://www.stepstone.de/stellenangebote--Softwarearchitekt-C-Industrielle-Messtechnik-Informatik-Naturwissenschaft-Ingenieurwesen-m-w-x-Oberkochen-ZEISS--7772474-inline.html?rltr=49_24_25_seorl_m_0_0_0_0_0_0

Schaeffler Engineering GmbH (Fahrzeugzulieferer)
Titel: Software-Architekt für Embedded Systems im Antriebsstrang (m/w/d)
Anzeige bei Stepstone am 24.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205124324/https://www.stepstone.de/stellenangebote--Software-Architekt-fuer-Embedded-Systems-im-Antriebsstrang-m-w-d-Werdohl-Schaeffler-Engineering-GmbH--7730040-inline.html

---

BAYOONET AG (IT & Internet)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Stepstone am 22.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205124742/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-Darmstadt-Muenchen-Berlin-Koeln-BAYOONET-AG--7823792-inline.html

---

Deutsche Bahn AG (Sonstige Branchen Transport & Logistik)
Titel: Mobile Softwarearchitekt Android (w/m/d)
Anzeige bei Stepstone am 22.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205125015/https://www.stepstone.de/stellenangebote--Mobile-Softwarearchitekt-Android-w-m-d-Frankfurt-am-Main-Deutsche-Bahn-AG--7606574-inline.html

---

aixigo AG (Finanzen)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Stepstone am 22.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205125212/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-Aachen-Koeln-Karlsruhe-Berlin-Duesseldorf-Remote-aixigo-AG--7821296-inline.html

---

TESAT-Spacecom GmbH & Co. KG (Luft- und Raumfahrt)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Stepstone am 22.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205125531/https://www.stepstone.de/stellenangebote--Software-Architekt-d-m-w-Backnang-TESAT-Spacecom-GmbH-Co-KG--7900522-inline.html?rltr=1_1_25_crl_m_0_0_0_0_0&cs=true

---

EDEKA Handelsgesellschaft Hessenring mbH (Groß- & Einzelhandel)
Titel: Softwareentwickler - Softwarearchitekt (m/w/d)
Anzeige bei Stepstone am 22.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205125721/https://www.stepstone.de/stellenangebote--Softwareentwickler-Softwarearchitekt-m-w-d-Melsungen-EDEKA-Handelsgesellschaft-Hessenring-mbH--7816072-inline.html

---

Paigo GmbH (Finanzdienstleister)
Titel: Software Architekt (m/w/d)
Anzeige bei Stepstone am 22.01.2022

Meierhofer AG (IT & Internet Gesundheit & soziale Dienste)
Titel: Software Architekt / Senior Entwickler (m/w/d) Healthcare IT
Anzeige bei Stepstone am 22.01.2022

SEAL Systems AG (Industrie)
Titel: Software Architekt (m/w/d)
Anzeige bei Indeed am 14.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205132538/https://t3n.de/jobs/job/seal-systems-ag/software-architekt-m-w-d/

IVU Traffic Technologies AG (Logistiksystemen)
Titel: Softwareingenieure und -architekten (m/w/d) für ÖPNV-Software
Anzeige bei Indeed am 29.12.2021
Archivierte Quelle: https://web.archive.org/web/20220205133445/https://www.ivu.de/bewerben/stellenangebote/softwareingenieure-und-architekten-mwd-fuer-oepnv-software-1

KALORIMETA GmbH (Imobilien)
Titel: Software Architekt (w/m/d)
Anzeige bei Stepstone am 04.02.2022
Archivierte Quelle: https://web.archive.org/web/20220205130726/https://www.stepstone.de/stellenangebote--Software-Architekt-w-m-d-Hamburg-KALORIMETA-GMBH--7895517-inline.html

---

PROFI Engineering Systems Bochum (IT & Internet)
Titel: Software-Architekt (m/w/d)
Anzeige bei Indeed am 29.12.2021
Archivierte Quelle: https://web.archive.org/web/20220205144051/https://www.profi-ag.de/stellenangebote/software-architekt-m-w-d/

Porsche AG (Fahrzeugbau/-zulieferer)
Titel: Softwarearchitekt (m/w/d) Porsche Conneced Car
Anzeige bei Stepstone am 25.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205144743/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-Porsche-Conneced-Car-Weissach-Porsche-AG--7916508-inline.html

---

Volkswagen Infotainment GmbH (Fahrzeugbau/-zulieferer)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Stepstone am 04.02.2022
Archivierte Quelle: https://web.archive.org/web/20220205145424/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-Bochum-Volkswagen-Infotainment-GmbH--4037259-inline.html

---

Diehl Group (Logistiksystemen)
Titel: Senior Software Entwickler & Architekt für Echtzeit AI Bildverarbeitungssysteme (m/w/d)
Anzeige bei Firmenseite am ?
Archivierte Quelle: https://web.archive.org/web/20220205145757/https://www.diehl.com/career/de/stellenboerse/stellenboerse/senior-software-entwickler-architekt-fuer-echtzeit-ai-bildverarbeitungssysteme-mwd/

Airbus (Luft- und Raumfahrt)
Titel: GALILEO Software Architekt (d/m/w)
Anzeige bei Talentify am ?
Archivierte Quelle: https://web.archive.org/web/20220205173048/https://airbus.talentify.io/job/software-architekt-mwd-stuttgart-baden-wurttemberg-airbus-jr10043984

CodeWrights GmbH (Automatisierung)
Titel: Software Architekt (m/w/d)
Anzeige bei Kimeta am 12.12.2021
Archivierte Quelle: https://web.archive.org/web/20220205173533/https://www.kimeta.de/search?q=Software+Architekt&loc=Rheinstetten&r=0&selected=671150086&mvc=1&coe=source%40mvcTrefferseite&coe=pagePosition%408&coe=logoShown%400&coe=topplacedShown%400&coe=page%401&slp=%2Fsoftware-architekt-stellenangebote-rheinstetten&back=%2Fsearch%3Fq%3DSoftware%2520Architekt%26loc%3DRheinstetten%26r%3D0%26mvcselected%3D671150086

Airbus (Luft- und Raumfahrt)
Titel: Software Architekt (d/m/w)
Anzeige bei Arbeitsplätze.net am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205174740/https://www.arbeitsplatze.net/job/de5df169479af81b167e45d6a81422b6ee

---

OHB Digital Services GmbH (Luft- und Raumfahrt)
Titel: Software Architekt (d/m/w)
Anzeige bei Firmenseite
Archivierte Quelle: https://web.archive.org/web/20220205175709/https://www.ohb-ds.de/karriere/softwarearchitekt-w-m-d

---

One Inside Germany (IT & Internet)
Titel: Software Architect
Anzeige bei Firmenseite
Archivierte Quelle: https://web.archive.org/web/20220205180821/https://one-inside.com/de/jobs/softwarearchitekt-m-w-d-80-100-d-de/

---

nedyx software GmbH (IT & Internet)
Titel: Software-Architekt (m/w/d)
Anzeige bei jobs.meinestadt.de am 26.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205181431/https://jobs.meinestadt.de/meerbusch/standard?id=245466260

BBR Verkehrstechnik GmbH (Verkehr)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei job 38.de am 25.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205182027/https://job38.de/jobs/softwarearchitekt-braunschweig?jobid=1531715

FIS (Finanzen)
Titel: Software Architekt - Java (m/w/d)
Anzeige bei Kimeta am 04.02.2022
Archivierte Quelle: https://web.archive.org/web/20220205183738/https://www.kimeta.de/search?q=Softwarearchitekt&r=30&_r=3645&page=1&searchId=9c9a7258-0ed2-4369-aaec-e979e4bc2df3&selected=680279067&coe=source%40searchPage&coe=page%402&coe=pagePosition%4011&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%40lggMITVoWuLOuwDh4YRUM&coe=searchId%409c9a7258-0ed2-4369-aaec-e979e4bc2df3

J.P. Sauer & Sohn Maschinenbau GmbH (Maschinenbau)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Kimeta am 06.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205183212/https://www.kimeta.de/search?q=Softwaretechniker&r=30&page=0&searchId=3b96b035-dbc7-4656-9b04-991f3223953b&selected=674881789&coe=source%40searchPage&coe=page%401&coe=pagePosition%402&coe=topplacedShown%401&coe=logoShown%401&coe=searchGroupId%40J9KPqxqwTbwoHqbOGQmDf&coe=searchId%403b96b035-dbc7-4656-9b04-991f3223953b

---

Bosch Rexroth (Maschinenbau)
Titel: Softwarearchitekt (m/w/div.)
Anzeige bei Kimeta am 03.02.2022
Archivierte Quelle: https://web.archive.org/web/20220205184744/https://www.kimeta.de/search?q=Softwarearchitekt&r=30&_r=3645&page=4&searchId=ad71d694-13d1-45bc-a92a-b790d38edbad&selected=679951492&coe=autoDelivered%400&coe=jobId%40679951492&coe=clientSource%40searchPage&coe=logoShown%400&coe=page%403&coe=pagePosition%407&coe=searchGroupId%40lggMITVoWuLOuwDh4YRUM&coe=searchId%40ad71d694-13d1-45bc-a92a-b790d38edbad&coe=topplacedShown%400&coe=source%40partner-back

---

Athlon Germany GmbH (Finanzdienstleister)
Titel: Lead Developer .NET / Software-Architekt (m/w/d)
Anzeige bei Stepstone am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205185226/https://www.stepstone.de/stellenangebote--Lead-Developer-NET-Software-Architekt-m-w-d-Duesseldorf-Athlon-Germany-GmbH--7614637-inline.html

---

ZF Friedrichshafen AG (Fahrzeugbau/-zulieferer)
Titel: Software Architekt (m/w/d)
Anzeige bei Kimeta am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205190101/https://www.kimeta.de/search?q=Zf&loc=Koblenz&r=0&page=0&searchId=a5162d6e-c1c4-41ce-b503-0c75c5f9ef61&selected=679114938&coe=source%40searchPage&coe=page%401&coe=pagePosition%4013&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%40J1TZMqT-kPc4-Zx1qvInb&coe=searchId%40a5162d6e-c1c4-41ce-b503-0c75c5f9ef61

Viessmann Group (DE) (Haustechnik)
Titel: Embedded Software Architekt (m/w/d)
Anzeige bei Kimeta am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205190829/https://www.kimeta.de/search?q=Softwarearchitekt&r=30&_r=3645&page=9&searchId=f3c00ba3-0aa9-42ec-b205-919e27a114c2&selected=679532261&coe=source%40searchPage&coe=page%4010&coe=pagePosition%409&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%402z3sDI6Z_1IxUpC76_FQ3&coe=searchId%40f3c00ba3-0aa9-42ec-b205-919e27a114c2

---

adesso SE (IT & Internet)
Titel: Software Architekt (all genders) Java in Leipzig
Anzeige bei Kimeta am 05.02.2022
Archivierte Quelle: https://web.archive.org/web/20220205191506/https://www.kimeta.de/search?q=Softwarearchitekt&r=30&_r=3645&page=9&searchId=f3c00ba3-0aa9-42ec-b205-919e27a114c2&selected=680143679&coe=source%40searchPage&coe=page%403&coe=pagePosition%406&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%402z3sDI6Z_1IxUpC76_FQ3&coe=searchId%4010989944-b449-4d9f-8beb-dc1f072968ac

---

Professional Scientists GmbH & Co. KG (Personaldienstleistungen)
Titel: Softwarearchitekt / Softwareentwickler (m/w/d) Embedded Software
Anzeige bei Stepstone am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205191957/https://www.stepstone.de/stellenangebote--Softwarearchitekt-Softwareentwickler-m-w-d-Embedded-Software-Grossraum-Stuttgart-Konstanz-Jena-Professional-Scientists-GmbH-Co-KG--7869658-inline.html

---

Secunet Security Networks AG (IT & Internet)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Stepstone am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205192236/https://www.stepstone.de/stellenangebote--Softwarearchitekt-m-w-d-Berlin-Essen-Siegen-Secunet-Security-Networks-AG--7857458-inline.html

---

KARL STORZ SE & Co. KG (Medizintechnik)
Titel: Software Architekt (m/w/d)
Anzeige bei Kimeta am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205192715/https://www.kimeta.de/search?q=Karl+Storz+&r=30&_r=4133&page=2&searchId=6cef44c9-3428-4622-9c63-fd7a77f30d38&selected=679950698&coe=autoDelivered%400&coe=jobId%40679950698&coe=clientSource%40searchPage&coe=logoShown%401&coe=page%403&coe=pagePosition%4015&coe=searchGroupId%40Akm3kTjWWzdQetbWCQZo3&coe=searchId%406cef44c9-3428-4622-9c63-fd7a77f30d38&coe=topplacedShown%401&coe=source%40partner-back

---

ZF Friedrichshafen AG (Fahrzeugbau/-zulieferer)
Titel: Softwarearchitekt für Radarsteuergeräte (m/w/d)
Anzeige bei Kimeta am 29.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205193040/https://www.kimeta.de/search?q=ZF+softwarearchitekt+&r=30&_r=7465&searchId=4b56f4ea-4f42-4127-ba6e-3c44c0073124&selected=678922160&coe=source%40searchPage&coe=page%401&coe=pagePosition%402&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%409s1Dy3VxOWwQbtoBTBsTW&coe=searchId%404b56f4ea-4f42-4127-ba6e-3c44c0073124

FERCHAU GmbH (IT & Internet)
Titel: Softwarearchitekt (m/w/d)
Anzeige bei Kimeta am 17.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205193401/https://www.kimeta.de/search?q=Ferchau+softwarearchitekt+&r=30&_r=5025&searchId=de10e454-6b02-416b-8081-45892492f353&selected=67694122 7&coe=source%40searchPage&coe=page%401&coe=pagePosition%407&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%40kbieHYVzR_WDlgu7DuO0-&coe=searchId%40de10e454-6b02-416b-8081-45892492f353

NORD/LB Norddeutsche Landesbank Girozentrale (Finanzen)
Titel: Software Architekt / Enterprise Architekt (m/w/d)
Anzeige bei Kimeta am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205193705/https://www.kimeta.de/search?q=Nord%2FLB+Software&r=30&_r=4231&searchId=2df3fe04-16a7-4c5b-920c-8e8945085713&selected=678895208&coe=source%40searchPage&coe=page%401&coe=pagePosition%403&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%40bCxq22BZQY1xcu6MEcJXh&coe=searchId%402df3fe04-16a7-4c5b-920c-8e8945085713

Brunel GmbH (Fahrzeugbau/-zulieferer)
Titel: Embedded Software Architekt für den Bereich (w/m/d)
Anzeige bei Kimeta am 28.01.2022
Archivierte Quelle: https://web.archive.org/web/20220205194037/https://www.kimeta.de/search?q=Brunel+Software&r=30&_r=2729&searchId=76ad9d23-900d-4520-bafd-62bb4400221d&selected=678904144&coe=source%40searchPage&coe=page%402&coe=pagePosition%407&coe=topplacedShown%400&coe=logoShown%400&coe=searchGroupId%40rv5J3gbSiOo6yc6ENJNUS&coe=searchId%4076ad9d23-900d-4520-bafd-62bb4400221d&page=1

Anhang B

Hochschulangebot

Für die Analyse der Lehre wurden insgesamt 57 Module von 15 verschiedenen Hochschulen herangezogen, welche einen Bezug zur Softwarearchitektur aufweisen.

Die folgende Liste enthält die Meta-Informationen aller gesammelten und analysierten Module.

Muster TU
Modul: Mustermodultitel
Quelle: muster-einer-aufgeloesten-ip.net
Abgerufen am 01.01.1900

RWTH Aachen
Modul: Software Architectures
Quelle: https://www.se-rwth.de/teaching/ws1819/pig/
Abgerufen am 20.01.2022

RWTH Aachen
Modul: Softwaretechnik
Quelle: http://www.tk.rwth-aachen.de/bachelor/232_module22.html
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Modellbasierte Softwareentwicklung
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=1768&pDocNr=6358856
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Software-Architekturen
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=1768&pDocNr=6358856
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Generative Softwareentwicklung
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=583&pDocNr=6357792
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Modellbasierte Softwareentwicklung
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=583&pDocNr=6357792
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Innovationen im Software Engineering
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=583&pDocNr=6357792
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Model-based Systems Engineering
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=583&pDocNr=6357792
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Objektorientierte Softwarekonstruktion
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=586&pDocNr=6358420
Abgerufen am 30.01.2022

RWTH Aachen
Modul: Model-based Systems Engineering
Quelle: https://online.rwth-aachen.de/RWTHonline/wbModhbReport.downloadPublicMHBVersion?pOrgNr=14194&pStpStpNr=586&pDocNr=6358420
Abgerufen am 30.01.2022

TU Braunschweig
Modul: Softwarearchitektur
Quelle: https://www.tu-braunschweig.de/isf/teaching/2018w/swa
Abgerufen am 20.01.2022

TU München
Modul: Softwarearchitekturen
Quelle: https://campus.tum.de/tumonline/wbModHb.wbShowMHBReadOnly?pKnotenNr=476726
Abgerufen am 20.01.2022

TU München
Modul: Software-Engineering in der industriellen Praxis
Quelle: https://campus.tum.de/tumonline/WBMODHB.wbShowMHBReadOnly?pKnotenNr=608628
Abgerufen am 30.01.2022

TU München
Modul: Softwarearchitekturen
Quelle: https://campus.tum.de/tumonline/wbModHb.wbShowMHBReadOnly?pKnotenNr=476725
Abgerufen am 30.01.2022

TU München
Modul: Software Engineering für betriebliche Anwendungen
Quelle: https://campus.tum.de/tumonline/WBMODHB.wbShowMHBReadOnly?pKnotenNr=460528
Abgerufen am 30.01.2022

TU München
Modul: Advanced Topics of Software Engineering
Quelle: https://campus.tum.de/tumonline/WBMODHB.wbShowMHBReadOnly?pKnotenNr=1060130
Abgerufen am 30.01.2022

TU München
Modul: Software-Engineering in der industriellen Praxis
Quelle: https://campus.tum.de/tumonline/WBMODHB.wbShowMHBReadOnly?pKnotenNr=608628
Abgerufen am 20.01.2022

TU München
Modul: Einführung in die Softwaretechnik
Quelle: https://campus.tum.de/tumonline/wbModHb.wbShowMHBReadOnly?pKnotenNr=452816
Abgerufen am 20.01.2022

TU München
Modul: Muster in der Softwaretechnik
Quelle: https://campus.tum.de/tumonline/WBMODHB.wbShowMHBReadOnly?pKnotenNr=473855
Abgerufen am 20.01.2022

BTU Cottbus
Modul: Softwaresystemtechnik
Quelle: https://www.b-tu.de/modul/12209
Abgerufen am 30.01.2022

BTU Cottbus
Modul: Entwicklung von Softwaresystemen
Quelle: https://www.b-tu.de/modul/12104
Abgerufen am 20.01.2022

BTU Cottbus
Modul: Softwaretechnik
Quelle: https://www.b-tu.de/modul/11289
Abgerufen am 20.01.2022

BTU Cottbus
Modul: Grundzüge der Softwaretechnik
Quelle: https://www.b-tu.de/modul/11904
Abgerufen am 20.01.2022

FernUniversität Hagen
Modul: Software Engineering
Quelle: https://www.fernuni-hagen.de/mi/studium/module/se_eins.shtml
Abgerufen am 20.01.2022

Universität Heidelberg
Modul: Einführung in Software Engineering
Quelle: https://hgs.iwr.uni-heidelberg.de/Portfolio_Modulhandbuch/OUTPUT/web_page_display_course.php?code=ISW
Abgerufen am 20.01.2022

TU Berlin
Modul: Softwaretechnik
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40715&version=1&sprache=1
Abgerufen am 30.01.2022

TU Berlin
Modul: Softwaretechnik und Programmierparadigmen
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40029&version=5&sprache=1
Abgerufen am 30.01.2022

TU Berlin
Modul: Architektur von Anwendungssystemen
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html;jsessionid=UAbwsfXjbcJYfmbBh9Yqbo5xvefsveb9EUG3FVJa.moseskonto?nummer=40333&version=10&sprache=1
Abgerufen am 20.01.2022

TU Berlin
Modul: Concepts of Software Engineering
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40865&version=2&sprache=2
Abgerufen am 30.01.2022

TU Berlin
Modul: Continuous Software Engineering
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40962&version=4&sprache=2
Abgerufen am 30.01.2022

TU Berlin
Modul: Current Topics in Software and Business Engineering
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40989&version=1&sprache=2
Abgerufen am 30.01.2022

TU Berlin
Modul: Hot Topics in Software and Business Engineering
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40975&version=2&sprache=2
Abgerufen am 30.01.2022

TU Berlin
Modul: Master Project and Seminar Software Engineering of Embedded Systems
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40832&version=3&sprache=1
Abgerufen am 30.01.2022

TU Berlin
Modul: Modellgetriebe-ne Software-Entwicklung
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40796&version=6&sprache=1
Abgerufen am 30.01.2022

TU Berlin
Modul: Software Architecture for Blockchain Applications
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40971&version=3&sprache=2
Abgerufen am 30.01.2022

TU Berlin
Modul: Software Engineering - A Practical Approach (big)
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40868&version=2&sprache=2
Abgerufen am 30.01.2022

TU Berlin
Modul: Software Engineering eingebetteter Systeme
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40710&version=8&sprache=1
Abgerufen am 30.01.2022

TU Berlin
Modul: Softwaretechnik
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40715&version=1&sprache=1
Abgerufen am 30.01.2022

TU Berlin
Modul: Softwaretechnik und Programmierparadigmen
Quelle: https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?nummer=40029&version=5&sprache=1
Abgerufen am 30.01.2022

KIT Karlsruhe
Modul: Softwaretechnik II
Quelle: https://sdqweb.ipd.kit.edu/wiki/Vorlesung_Softwaretechnik_II_WS_2021/22
Abgerufen am 20.01.2022

KIT Karlsruhe
Modul: Softwaretechnik I
Quelle: https://ps.ipd.kit.edu/405_1859.php
Abgerufen am 20.01.2022

FernUniversität Hagen
Modul: Software-Architektur
Quelle: https://www.fernuni-hagen.de/mi/studium/module/pdf/Leseprobe-komplett_01798.pdf
Abgerufen am 20.01.2022

TU Dresden
Modul: Softwaretechnologie II
Quelle: https://tu-dresden.de/ing/informatik/smt/st/studium/lehrveranstaltungen?leaf=1&lang=de&subject=439&embedding_id=47eddfa7c5a54ed5be49042aff35a31b
Abgerufen am 20.01.2022

HU Berlin
Modul: Requirements Engineering und Software-Architektur - Detailseite
Quelle: https://agnes.hu-berlin.de/lupo/rds;jsessionid=74DBAA40D3929E87A62388E61086E93B.angua_root?state=verpublish&status=init&vmfile=no&moduleCall=webInfo&publishConfFile=webInfo&publishSubDir=veranstaltung&veranstaltung.veranstid=137784&expand=1130900#contentinformation
Abgerufen am 20.01.2022

HU Berlin
Modul: Software Engineering
Quelle: https://www.informatik.hu-berlin.de/de/forschung/gebiete/se/teaching/ws2021/se/se
Abgerufen am 20.01.2022

Uni Freiburg
Modul: Softwaretechnik / Software Engineering (Lecture)
Quelle: https://swt.informatik.uni-freiburg.de/teaching/SS2016/swtvl
Abgerufen am 20.01.2022

Uni Bonn
Modul: Softwaretechnologie
Quelle: https://sewiki.iai.uni-bonn.de/teaching/lectures/se/2020/vorlesungsinhalte
Abgerufen am 20.01.2022

LMU München
Modul: Methoden des Software Engineering
Quelle: https://cms-cdn.lmu.de/media/contenthub/studiengangsfinder/downloads/16_mhdb_msc_informatik_120_ects_psto_08_09_2010_stand_18_12_2014_de.pdf
Abgerufen am 30.01.2022

LMU München
Modul: Software Engineering für spezielle Anwendungsgebiete (INF-SEspA)
Quelle: https://cms-cdn.lmu.de/media/contenthub/studiengangsfinder/downloads/16_mhdb_msc_informatik_120_ects_psto_08_09_2010_stand_18_12_2014_de.pdf
Abgerufen am 30.01.2022

Uni Heidelberg
Modul: Einführung in Software Engineering
Quelle: https://se.ifi.uni-heidelberg.de/teaching/ws_1314/isw.html
Abgerufen am 30.01.2022

Uni Heidelberg
Modul: Requirements Engineering
Quelle: https://se.ifi.uni-heidelberg.de/teaching/previous_semesters/sose_2020/iswre.html
Abgerufen am 30.01.2022

Hochschule Reutlingen
Modul: Software Engineering
Quelle: https://www.inf.reutlingen-university.de/fileadmin/user_upload/Fakultaet_INF/Downloads/Studium/Master/PSE/PSE_Modulhandbuch_2019-v1.0.pdf
Abgerufen am 30.01.2022

Hochschule Reutlingen
Modul: Softwarearchitektur
Quelle: https://www.inf.reutlingen-university.de/fileadmin/user_upload/Fakultaet_INF/Downloads/Studium/Master/PSE/PSE_Modulhandbuch_2019-v1.0.pdf
Abgerufen am 30.01.2022

Hochschule Reutlingen
Modul: Architekturmanagement
Quelle: https://www.inf.reutlingen-university.de/fileadmin/user_upload/Fakultaet_INF/Downloads/Studium/Master/WI/Modulhandbuch_03-2021.pdf
Abgerufen am 30.01.2022

Hochschule Reutlingen
Modul: Advanced Software Engineering
Quelle: https://www.inf.reutlingen-university.de/fileadmin/_migrated/media/Module_Handbook_Master_of_Business_Informatics__wiM_.pdf
Abgerufen am 30.01.2022

Hochschule Reutlingen
Modul: Architecture Management
Quelle: https://www.inf.reutlingen-university.de/fileadmin/_migrated/media/Module_Handbook_Master_of_Business_Informatics__wiM_.pdf
Abgerufen am 30.01.2022

UNI Freiburg
Modul: Software Engineering
Quelle: https://campus.uni-freiburg.de/qisserver/pages/cm/exa/curricula/genericRailsSearchUnitsSimple.xhtml?_flowId=searchCourseOfStudyForModuleDescription-flow&_flowExecutionKey=e1s3
Abgerufen am 30.01.2022

Anhang C

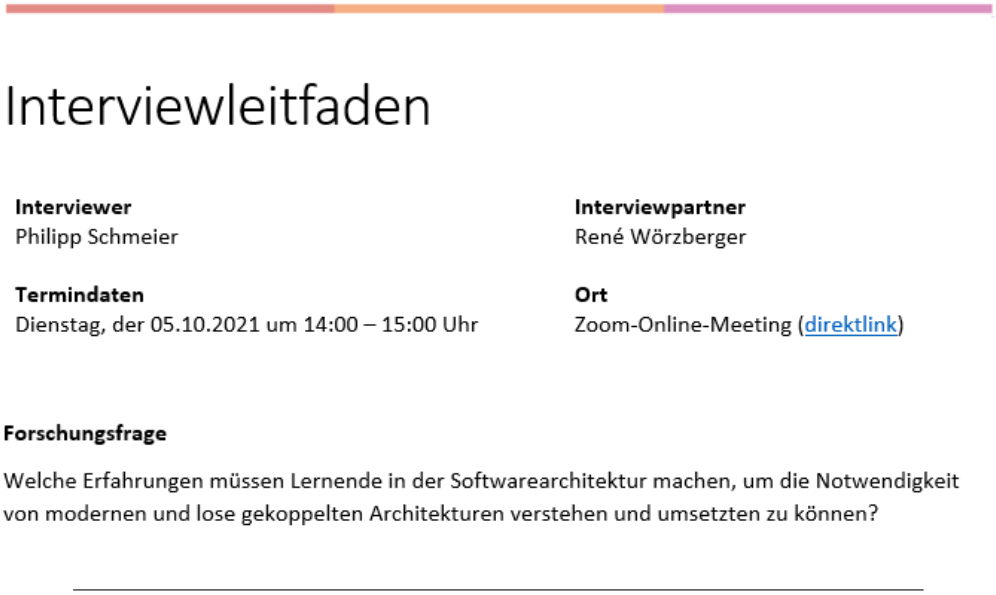
Experteninterviews

Für die Experteninterviews wurden zwei Leitfäden erstellt. Einmal mit Fokus auf Fragen für die Lehre und einmal mit Unternehmensfokus. Die Ergebnisse wurden transkribiert, das Transkript auf Lesbarkeit überarbeitet und ausgewertet. Die wichtigsten Zitate, welche auch innerhalb der Arbeit referenziert wurden, sind in Kapitel C.2 nach Themenbereichen sortiert.

C.1 Leitfäden

Die Interviewleitfäden sind in drei Rubriken unterteilt.

1. Header mit organisatorischen Informationen (Bsp. Abbildung C.1)
2. Die für alle Interviews relevante Forschungsfrage
3. Der Themenleitfaden (Fragenkatalog)



Interviewleitfaden

Interviewer
Philipp Schmeier

Interviewpartner
René Wörzberger

Termindaten
Dienstag, der 05.10.2021 um 14:00 – 15:00 Uhr

Ort
Zoom-Online-Meeting (direktlink)

Forschungsfrage

Welche Erfahrungen müssen Lernende in der Softwarearchitektur machen, um die Notwendigkeit von modernen und lose gekoppelten Architekturen verstehen und umsetzten zu können?

ABBILDUNG C.1: Beispiel: Header und Forschungsfrage der Interviewleitfäden

Der Themenleitfaden ist wiederum in drei Bereiche eingeteilt: Intro, Kernfragen und Outro. Die Kernfragen unterscheiden sich dabei zwischen den Interviews leicht. Zum einen gibt es Kernfragen, welche besonders auf diejenigen Personen abzielen, die selbst in der Lehre tätig sind (siehe Kapitel C.1.1) und andere Kernfragen für diejenigen Personen, die in Unternehmen tätig sind (siehe Kapitel C.1.2).

Das Intro beinhaltet dabei folgende vier Punkte:

- Vorstellung des Interviewenden und der jeweiligen Themen
- Zeitplan, Rahmen und Ziel des Interviews
- Einwilligung für die Aufnahme des Interviews
- Kurze Vorstellung seitens des Interviewpartners

Abschließend lässt das Outro noch Platz für sonstige Anmerkungen, Fragen, Wünsche oder Hinweise des Interviewten sowie für die Verabschiedung.

C.1.1 Kernfragen mit Lehrfokus

1. Wie bringen Sie Ihren Studierenden Softwarearchitektur näher?
    - Welche Module gibt es bei Ihnen, die Softwarearchitektur behandeln?
    - Auf welche Kernaspekte legen Sie dabei den Fokus?
    - Gibt es dabei Methodiken oder Skills (auch Softskills), welche Sie Studierenden vermitteln möchten? (Bspw. Clean Code, Test Driven Development, Event Storming)
    - Gehen Sie dabei von einem Brown- oder Greenfield Umfeld aus?
2. Welche besonderen Herausforderungen existieren bei der Vermittlung von Softwarearchitekturen?
3. Welche größeren Wissenslücken zeichnen sich bei Ihren Studierenden in Bezug auf Softwarearchitektur ab?
4. Wenn Sie Studierenden lose gekoppelte Softwarearchitektur näherbringen möchten, was wäre/ist ihr erster Schritt?
5. Wo werden bei Ihnen lose gekoppelte Softwarearchitekturen genutzt?

C.1.2 Kernfragen mit Unternehmensfokus

1. Wie sahen Ihre ersten Berührungspunkte mit Softwarearchitektur aus?
2. Welche Eigenschaften schätzen Sie besonders bei der Arbeit mit jungen bzw. unerfahrenen Softwarearchitekt:innen?
    - Was sind die größten bestehenden Wissenslücken dieser?

- Gibt es Methodiken oder Erfahrungswerte, welche für den Architekt:innen Beruf erforderlich oder besonders hilfreich sind?
- Sind Softskills Teil dieser Erfahrungswerte, falls ja, welche beachten Sie hierbei als besonders wichtig?

3. Wie arbeiten Sie sich selbst in neue Systeme ein?
   - Welche Fragen stellen Sie, wenn Sie das erste Mal mit einer neuen Architekturlandschaft in Kontakt kommen?
   - Gibt es Problematiken, welche Sie in fast jeder Architektur wiedererkennen?
4. Welche Technologien sollte jede/r angehende Architekt:in kennen?
5. Wenn Sie angehenden Architekt:innen lose gekoppelte Softwarearchitektur näherbringen möchten, was wäre Ihr erster Schritt?
6. Wo werden bei Ihnen lose gekoppelte Softwarearchitekturen genutzt?

C.2 Themen

C.2.1 Einstieg als Softwarearchitekt:in

Jann Deterling

(1) *„Das heißt du gehst sechs Wochen, damals noch nach Indien oder nach China, da gibt es eine Thoughtworks University. Dort wird dann vermittelt, wie ThoughtWorks gewisse Projekte angeht. Wir sind eine Consultant-Firma, das heißt, wir fahren zum Kunden raus bzw. arbeiten mit den Kunden sehr eng daran, dass ihre Projekte umgesetzt werden und sind dann natürlich auch auf allen Ebenen wie Architektur, Programmierung und auch Methoden beizubringen, dabei.“*

(2) *„[...] Ein virtuelles Projekt, was von anderen Thoutworkern vorbereitet wird. Du hast [...] dann pro Woche eine Iteration. Das Ganze läuft agil ab. [...] Du hast einen PO, du hast Lehrende sozusagen, die einem dabei helfen, das Projekt-Pensum aufzuarbeiten. Und dann gibt es verschiedene Storys, die einfach abgearbeitet werden, nach und nach. Man muss alles selbst organisiert machen. Das heißt: erst einmal anfangen. Also ein Team ohne alles. Das heißt, man hat nur das Wissen: Okay, das ist das Projekt. Und dann muss man sich komplett einarbeiten. Das ist so ein bisschen, wie das Setting losgeht. Hat leider, muss man auch sagen, relativ wenig mit der realen Welt zu tun, weil das natürlich schon sehr isoliert ist und sehr in diesem Thoughtworks Kosmos abspielt und darauf aufgebaut ist, wie unsere Projekte im Idealfall ablaufen sollten. Was auch gut ist, das so zu lernen.“*

Eberhard Wolff

(1) *„[...] da ich eben nicht eine strukturierte Ausbildung oder so etwas hatte, sondern in dem Projekt Ad hoc dann versucht habe die Herausforderungen zu lösen.“*

(2) *„Softwarearchitekt, mindestens als Titel, ist etwas was ja eine „seniorige" Position ist. Und eine Person, die frisch von der Uni kommt, wird typischerweise nicht in einer Softwarearchitekturrolle offiziell landen."*

Sebastian Gauder

„Warum haben wir diese Entscheidung getroffen? [...] wenn Leute verstehen, warum man gewisse Dinge gemacht hat, dann glaube ich, ist es auch einfacher für die Leute, das auch zu leben und anzuwenden."

C.2.2 Einarbeitung in neue Systeme

Jann Deterling

(1) *„Generell fange ich erst einmal an, mit den Leuten zu reden, also erstmal zu verstehen: Okay, worum geht es hier überhaupt? Also versuchen grob die Domäne schonmal zu umfassen, was ist das Ziel von dem Projekt? Was soll das Produkt am Ende können? Was ist die Vision des Produkts? Das ist jetzt natürlich noch sehr fachlich und hat eigentlich technisch noch gar nichts zu sagen. Ich glaube, das ist erst einmal das Wichtigste, was man machen muss. Erst mal zu verstehen: Was soll ich hier tun?"*

(2) *„Wie wurde das System entwickelt? Und warum ist es so entwickelt worden?"*

(3) *„Habe ich Zugang zum Code? Wenn ja, würde ich mir einen Pair nehmen, der vielleicht schon ein bisschen mehr Erfahrung auf dem Projekt hat und einfach mal versuchen, an einer Story mitzuarbeiten, zu verstehen: Wie sind die Abläufe hier im Team? Wie arbeitet das Team zusammen? Wie werden neue Anforderungen entwickelt? Und dann auch zu verstehen? Okay, wie ist die Architektur in der Anwendung selbst aufgebaut? Und dann geht es meistens wirklich, finde ich, ins Code lesen. Zu verstehen, wie ist der Code aufgebaut?"*

Eberhard Wolff

(1) *„Eigentlich ist die Art und Weise, wie mir das vorgestellt wird, schon relevant und interessant."*

(2) *„Und da sind Fragetechniken eigentlich ein Thema für mich. Zum einen ist das für mich so, dass ich versuche, sozusagen Inkonsistenzen zu finden. Du hast aber vorher das gesagt, und jetzt sagst du das, warum ist das so? Weil das halt etwas ist, wo man dann noch einmal ein tieferes Verständnis für hinbekommen kann. Und ich versuche tatsächlich, die Informationen aufzunehmen, und mir nicht schon gleich eine Meinung zu bilden."*

(3) *„Extrem schwierig wird es an der Stelle, wo man nach der Semantik der Boxen fragt. Das ist vielleicht ein Modell, mit dem sich diese Personen unterhalten und dann sagen okay, so sieht es irgendwie aus oder so könnte es aussehen. Aber eigentlich möchte ich eine Abstraktion über den konkreten Code haben [...] und dafür müsste ich eine klare Semantik der Box haben. Also ich müsste jetzt sagen, jede*

Box ist etwas, was ich im Source Code irgendwo lokalisieren kann. [...] Das bedeutet, wenn ich über die Box rede, rede ich über einen Teil des Sources Codes und ich kann darüber über die Struktur reden. Das ist leider nicht immer gegeben. Und es ist dann auch noch so, dass [...] die Semantik der Box nicht klar ist. Und dann kommt manchmal noch hinzu, dass es nicht klar ist, ob man über Ist oder Soll spricht! Dann wird es endgültig kompliziert.“

Sebastian Gauder

„Also so richtig stellt sich die Frage bei mir nicht. [...] Das läuft so ab, dass ich eigentlich, von vornherein bei jeder Idee, auf einem sehr groben Level, mit dabei bin. Das heißt, wir haben einen Prozess, wie überhaupt neue Ideen in das System reinkommen.“

Gernot Starke

(1) *„Ich versuche, das im ersten Schritt zu verstehen: Was soll das Ding überhaupt tun? Also was ist Ziel des Systems, was ist so eine Art Vision, warum ist das Ding überhaupt gebaut worden? Also bevor ich verstehe, wie es gebaut war, muss ich erst mal wissen, warum, und was sind so grundsätzliche Anforderungen?“*

(2) *„[...] was ist eure Subsystem-Struktur? Wie ist euer System im Inneren, im Großen organisiert? Habt ihr ein paar Schichten? Habt ihr ein paar Microservices, die halt irgendwie miteinander interagieren? Kannst du mir die großen Bestandteile deines Systems mal zeigen, ohne dass wir in den einzelnen Code gucken, sondern kannst du mir so die großen Bestandteile zeigen?“*

(3) *„Gib mir zumindest die Möglichkeit, dass ich mal validieren kann, also mich selbst überzeugen kann, ob die Kästchen, die ihr da gezeichnet habt, irgendwie einer Realität im Code entsprechen.“*

C.2.3 Wiederkehrende Problematiken

Jann Deterling

„Ich glaube vieles geht in so eine Vorurteil-Richtung rein, dass man sich nicht genug informiert hat über gewisse Sachen. Also, ich habe jetzt schon mal schnell auf die Schnelle die Top fünf Benefiz von einem Microsoft-System nachgelesen. Oder, ich habe nur die Nachteile durchgelesen. [...] Und da kommt wieder dieses Domänenwissen mit rein, das ich so wichtig finde. Dass man erst mal guckt, okay, was ist überhaupt das Problem in der Domäne das wir lösen wollen?“

Eberhard Wolff

(1) *„[...] wenn ich jetzt Architektur-Dokumentation bekomme, wo nicht vorne drinsteht, das sind übrigens die Sachen, die relevant sind, dann ist das eigentlich schon mal ein Problem. [...] Und wenn dann aber vorne zum Beispiel stehen würde, das soll irgendwie benutzerfreundlich sein, ist die nächste Frage: Wie kann ich das denn*

objektivieren oder überhaupt feststellen? Und das ist das nächste Thema woran die Leute dann scheitern. Und dass ist das dritte Thema: Wie setze ich das denn um?“

(2) *„Dann wird es deswegen schief, weil sie es entweder nicht besser wissen oder weil sie eben rücksichtslos vorgehen [...] durch Termindruck, und es dadurch nicht auf die Reihe bekommen es eben sauber aufzubauen.“*

Sebastian Gauder

(1) *„Das Klassische ist, [...] dass man, während man das Coded, schon weiß, dass das technische Schuld werden wird. [...] Aber häufig gibt es dann doch Gründe, die sagen [...] wir müssen jetzt doch lieber schneller in den Markt mit diesem neuen Feature.“*

(2) *„Das wird auch eine Sache, die wir rückblickend hätten besser machen können. Das ist nämlich: Entscheidungen besser dokumentieren. Im Endeffekt ist es dann doch so, dass jemand reinkommt und sich einfach mit den Gegebenheiten abfinden muss. Es gibt gute Möglichkeiten, solche Entscheidungen zu dokumentieren. Es gibt zum Beispiel ADR.“*

Gernot Starke

(1) *„Ich habe in einem wirklich relevanten Projekt in einem größeren Unternehmen in Deutschland ein ganz grobes Review gemacht und da hatte ich halt diesen Klassiker, ich dachte: Wollt ihr mich auf den Arm nehmen? Ist das eure zentrale Architekturdarstellung? Also es gab eine Person, die das gemacht hat, die hat mir das zum Angucken gegeben: Vier verschiedene Arten von Pfeilen. Und in der Mitte, also der Kern war ein Fabriksymbol, also eine gezeichnete Fabrik. Ein Schema, rote Pfeile ausgefüllt, blaue Pfeile ausgefüllt, blaue Pfeile mit Spitze aber nicht ausgefüllt und noch schwarze Pfeile und gestrichelte. Ohne Legende.“*

(2) *„Ja, wenn ich nur drei Services habe, [...] die kann ich mir irgendwie zusammen packen in ein Linux, kann mir mit einem Tail oder Head das angucken, das kriege ich noch hin. Aber wenn ich 50 hab oder ich habe 20 Services mit jeweils 20 Instanzen, das kann ich nicht mehr überblicken. Da brauche ich schon wirklich schlaue Mittel, um da überhaupt zu merken: Oh, einer von den 20 hat gerade irgendwie „Schluckauf“. Um rauszufinden, welche Clients benutzen denn gerade diese eine Instanz? Da sind so ganz normale, triviale, simple Mittel wie auf eine Konsole gucken, nicht mehr in der Lage, die notwendige Antworten zu geben.“*

C.2.4 Technologien

Jann Deterling

(1) *„Ich glaube an technischen Skills ist wirklich so etwas wichtig, wie diese gängigen Best Practices. Ich mache am besten TDD, Clean Code hattest du ja, glaube ich, auch in deinen Fragen genannt. Ich bin sehr überzeugt von Pair Programming, weil man einfach vier Augen permanent auf dem Code hat. Und man muss auch Pair Programming gar nicht nur auf programmieren sehen, sondern auch auf Architecture-Entscheidung. Wenn ich immer jemanden an meiner Seite habe, der mit mir bereit*

ist, darüber zu diskutieren, wie wir vielleicht gerade eine Änderungen unserer Architektur machen oder wie wir eine Kommunikation zwischen zwei Services aufbauen, oder wie wir auch unsere Architektur innerhalb einer Anwendung also innerhalb eines Services aufbauen, dann kriege ich immer wieder Feedback. Das heißt mein Ziel wäre es immer, kleine Feedback Cycles zu bauen.“

(2) *„Ich glaube womit man heutzutage in fast jedem Projekt konfrontiert wird: Java Spring Boot. Also das sollte man, einfach kennen oder schon einmal gehört haben. [...] Wenn es nicht Java ist, dann ist es jetzt Gottseidank Kotlin und Spring.“*

(3) *„Was finde ich auch immer wieder kommt ist diese Sichtweise: Ich bin nur ein Backend-Dev und will kein Frontend machen. Das kommt auch irgendwie oft. Ich finde, davon sollte man sich auch heutzutage ein bisschen distanzieren, weil das Frontend einen immer größeren Teil in der Architektur und in der gesamten Anwendung einnimmt.“*

Eberhard Wolff

„Es gibt Themen, die halt irgendwie modern sind, Kubernetes, Microservices, ... Das sind Sachen, wo man, sozusagen, grob eine Einordnung machen können sollte. Aber ich bin mir nicht sicher, ob man da jetzt sagen sollte: Jeder Architekt sollte das kennen.“

Sebastian Gauder

(1) *„Da Achten wir eigentlich weniger drauf. Also wir haben bisher nur Architekten aus unserem eigenen Kreis gecastet. [...] Deswegen wissen wir eigentlich schon, dass die schon in dem Umfeld gearbeitet haben, was wir vorgegeben haben.“*

(2) *„Also wir haben unsere Architektur-Prinzipien und Guides. Die müssen sich halt in den Bahnen bewegen, die irgendwie bei uns vorgegeben sind. Deswegen wissen wir schon, dass sie mit Microservices gearbeitet haben, dass sie wissen wie die funktionieren, worauf es ankommt, wie man Eventing macht, wie Schnittstellen gemacht werden und so etwas. Deswegen, von den Hard Skills haben wir das [...] den Leuten selber schon antrainiert. Deswegen achten wir wenig darauf [...] was für konkretes Architektur-Skills die Leute haben, sondern mehr: Wie „ticken“ die? Wie motiviert sind die? Was ist der Grund dafür, dass sie die Rolle annehmen wollen?“*

(3) *„[...] Ich glaube, da ist auch eine Gefahr drin, von vornherein sich auf irgendwelche Technologien zu fixieren.“*

(4) *„Neun Module. Es geht los mit Big Picture, dann gibt eine Einführung in Domain Driven Design. Em, dann gibt es Kommunikation, also synchrone Kommunikation, asynchrone Kommunikation, Security.“*

C.2.5 Softskills

René Wörzberger

„Eben dem Aspekt das so ein Softwarearchitekt, ja so dieses Elevator-Spiel gut können muss, also hoch fahren können muss, mit dem Management sprechen können

muss, mit dem Fachbereich sprechen können muss oder dem Product Ownern, die halt nicht so technisch drauf sind. Aber er sollte natürlich schon aus einer technischen Richtung herausgewachsen sein“

Jann Deterling

(1) *„Zum Einstieg, so generell in Softwarearchitektur oder auch generell als Entwickler. Ich glaube, was wichtig ist, ist, dass man so eine gewisse Neugier hat, eine gewisse Neugier und Probleme lösen und verstehen zu wollen.*“

(2) *„Eine Sache, die ich auch noch relativ gut finde ist, wenn man gut darin ist Feedback anzunehmen und geben zu können, also so eine Art Feedback-Kultur im Team zu haben. Das [...] basiert natürlich alles auf Vertrauen im Team. [...] wenn man offen Feedback in einem Team teilen kann.*“

Eberhard Wolff

(1) *„Also erst einmal, das ist halt ein People Business, [...] wir versuchen, mit vielen Personen zusammen, Systeme zu bauen. Und das impliziert, dass wir da miteinander reden müssen.*“

(2) *„[...] ein Fehler, den man machen kann, ist, sich hinzustellen und zu sagen okay, dass ist also mein Team und ich sage diesem Team jetzt: Das ist übrigens die Lösung, macht mal, weil offensichtlich ist das die Lösung, ist das die Art und Weise wie ich das Problem lösen würde. Eigentlich empfinde ich das mittlerweile als nicht mehr so schlau. Ich glaube, es ist wichtiger zu sagen: Was machen wir? Okay, habt ihr an diese Dinge gedacht? Und [...] so Leitplanken aufzustellen.*“

Sebastian Gauder

„Und ich finde immer, die wichtigste Eigenschaft ist, über den Tellerrand zu gucken, eine Ebene höher, was kann ich hier verändern? Was kann ich hier besser machen? Damit es quasi in der Mikroarchitektur besser funktioniert, also auch einmal große Dinge anfassen. Nicht zu denken: Okay, das ist hier etwas, schon außerhalb meiner Grenzen, da kann ich eh nichts dran ändern, sondern Mut, auch an den großen Sachen etwas zu machen.“

Gernot Starke

„Das ist etwas, was ich unglaublich schätze, wenn Leute auf Basis einer robusten, soliden technischen Skill-Sammlung, auch über entsprechende zugehörige, notwendige Kommunikationsfähigkeiten verfügen.“

C.2.6 Einstieg für Studierende

René Wörzberger

(1) *„Ja eben, das sage ich auch jedem Studierenden: Also kompliziert wird es von ganz alleine, da muss man nicht am Anfang schon für sorgen.*“

(2) *„Das hat mit Architektur eigentlich wenig zu tun. Es sei denn, man betrachtet jetzt Infrastrukturen aus architektonischer Sicht. Also wo gehört der Load Balancer hin? [...] wo gehört der Application Server hin, wo gehört die Datenbank hin, ist aber relativ stark im Hintergrund also, da habe ich in den Modulen wenig zu tun, mit Architekturen.“*

(3) *„Jetzt nutze ich das in der DevOps-Vorlesungen noch insoweit, dass ich einmal sagen okay, hier ist der Monolith, und jetzt haben wir den auch zerlegt. Das habe ich [...] denen vorgegeben. [...] Eure Aufgabe ist es jetzt, das zu containerresieren und zu deployen in die Cloud. Und dann müssen die in einem Container, [...] eine Änderung vornehmen, deversonieren, neue Container Version erstellen und die redeployen [...]. Da geht es gar nicht darum, jetzt irgendwie eine Riesenanforderung umzusetzen, in einem Brownfield Projekt, sondern macht irgendeine Änderung und seht aber zu, dass ihr es mit Docker und Cloud, ordentlich umgesetzt bekommt, möglichst auch so, dass das Ganze ausfallfrei redeployed werden kann.“*

(4) *„Das Problem, was ich jetzt habe beispielsweise, im 5. Semester, den Leuten Kubernetes beizubringen, ist schon didaktisch relativ schwer. Also nicht, denen die Technik zu erklären, sondern denen zu erklären, warum muss man sich jetzt damit beschäftigen? Warum muss man das skallieren, warum muss man sich mit solchen yaml Files rumschlagen, und zum Schluss tut dieses [...] Programm dann doch nur das, was es tut. [...] Das muss man den Leuten halt auch sagen, und das sage ich denen auch immer explizit: Ihr müsst jetzt von dieser Einfachheit des Anwendungsbeispiels, ein Stück weit abstrahieren. Also, ob das jetzt ein Guestbook ist, wo man dann immer nur eine Zeile eintippt [...] oder ob es jetzt Facebook ist [...] eigentlich egal.“*

(5) *„Das Problem ist, Brownfield wäre halt realistischer und große Beispiele wären halt realistischer. Da hat man an einer Hochschule immer das Problem, dass die Beispiele, damit sie irgendwie in der Kürze der Zeit aufnehmbar sind, immer relativ klein sind gegenüber dem, was die Studierenden ja eigentlich erwartet im späteren Berufsleben und die Methoden, [...] mit denen man darauf schießt, viel zu groß sind.“*

Jann Deterling

„Wahrscheinlich erst mal ganz klassisch, zu erklären: Was gibt es für verschiedene Architekturen? Also aufzeigen: [...] Es gibt, monolithisches Design, es gibt Microservices, es gibt [...] alles Mögliche dazwischen, [...] Und dann wahrscheinlich erst einmal wirklich deren Vor-und Nachteile durchgehen.“

Eberhard Wolff

„Ich bin ja, Mitglied im iSAQB. Der iSAQB hat sich als Aufgabe gesetzt, Wissen rund um Softwarearchitektur zu verbreitern. Und da gibt es ein definiertes Ausbildungsprogramm, mit Foundation-Level und Advanced-Level.“

Sebastian Gauder

(1) *„Was ich immer wichtig finde, ist, dass man irgendwie Spaß und Interesse daran hat, an Gesamtlösungen.“*

(2) *„Aber grundsätzlich [...] das Wichtigste ist, den Leuten zu sagen, warum habe ich was wie gemacht?“*

Gernot Starke

(1) *„Was ich mir vorstellen könnte ist, dass ihr so eine Art Retrospektive zwischendrin macht. Also sagt: so wir legen jetzt mal los. Aber bevor sich Leute verrennen dann schon nach relativ kurzer Zeit, einem Monat, mal sagen: So, lasst uns einen Break machen. Lasst uns mal systematisch von den Strukturentscheidungen, Technologieauswahl, Code angucken, sonstwie, bis hin zu konkret Arbeitsweise, reflektieren. Hat jemand zu viel Workload? Ist das halbwegs vernünftig verteilt? Wie ist unser planning? Konnten wir die selbstgesteckten Ziele einhalten? Oder müssen wir vielleicht an der Stelle noch mal was tun, um auf die Weise Leuten klarzumachen ja, ihr habt zwar sehr cool Sourcecode produziert, aber ihr habt halt einen in den vier Wochen schon total verschlissen, weil der die ganze Arbeit machen musste und ihr habt euch nicht genug ausgetauscht. Und ihr habt jetzt schon, nach vier Wochen, widersprüchliche Vorstellungen, wo es überhaupt hingehen soll.“*

(2) *„Ich muss ja erst mal dazu verstehen: Was bedeutet Kopplung? Kopplung wird manchmal in Architekturbüchern als etwas Negatives dargestellt. Aber andererseits ist das die Grundlage der Zusammenarbeit. Das heißt: Kopplung erst mal an sich, ist überhaupt gar nicht schlecht, sondern ist notwendig, als so eine Art Lebenselixier, dass Systeme überhaupt zusammenarbeiten können. [...] Dann könnte ich drüber reden: Ja, wodurch kann Kopplung entstehen und was gewinne ich, wenn ich die verschärfe oder löse? [...] Lose gekoppelte Systeme in Richtung Microservices vielleicht gedacht, machen mir einen Haufen Stress, den ich an vielen Stellen gar nicht will. [...] im Betrieb dieser Systeme. Ich brauche so unglaublich viel Infrastruktur, damit ich den Überblick behalten kann. Es können so komische Fehlersituationen auftreten nur dadurch, dass ich entkoppeln wollte.“*

(3) *„Remote ist nicht Local. Oder wo sind die Unterschiede zwischen Remote und Local? Was gibt es für zusätzliche Fehlerquellen?“*

(4) *„Da ist es eigentlich fast egal, mit welcher solcher Technologie-Mixen du dich beschäftigt hast. Wenn man mal eine mittelgroße, ein bisschen größere Anwendung ans Laufen gekriegt hat, dann ist das super. Dann hat man da schon ganz viel an tollen Voraussetzungen.“*

C.2.7 Sonstiges

René Wörzberger

„Also generell ist meine Lehre von der Situation zur Abstraktion und dann erst in der Abstraktion selber rum sinnieren, dann aber auch wieder runter zum eigentlichen Code. Das ist dieser Roundtripp sozusagen meine generelle Lehr-Philosophie.“

Jann Deterling

„Bei uns gibt es keinen wirklichen Software-Architekten.“

Eberhard Wolff

„Ich teile ein System in Module auf, so, und ein Grund warum ich das in Module aufteile, das ist zumindest die klassische Argumentation würde ich behaupten ist, dass ich sonst nicht dazu in der Lage bin, dieses System überhaupt zu verstehen. Also das Hauptproblem, das wir eigentlich haben ist, dass eine größere Gruppe von Menschen, ein Stück Software baut, und keine dieser Personen kann das gesamte System verstehen.“

Sebastian Gauder

(1) *„In meinem Fall, also ich kümmere mich um E-Commerce-Plattformen. Da muss ich jetzt mal schauen, aber ich denke, wir haben so um die 200 bis 250 Microservices, das wird halt irgendwann unübersichtlich.“*

(2) *„Also wir haben jetzt mehrere Ausgründungen gemacht, also noch mal neue, ganz neue Produkte auf der grünen Wiese entwickelt, wo wir gesagt haben, dass ist Overkill mit Microservices. Das machen wir an der Stelle nicht. Wir bauen jetzt wieder schön Monolithen, vielleicht schön modularisiert, sodass wir uns das irgendwie offen halten können. Und bauen später wieder einzelne Teile raus. Aber von vornherein auf der grünen Wiese Microservices anfangen [...] halte ich für ein Antipattern, das ist von vornherein viel zu aufwendig und zu teuer.“*

(3) *„[...] ich habe quasi den Beginn der Firma etwas verpasst, bin etwas später eingestiegen, zum Zeitpunkt, als wir im Prinzip noch einen großen Monolithen hatten und gerade angefangen wurde, den Monolithen auszuschlachten und in kleinere Microservices aufzuteilen.“*

Gernot Starke

(1) *„Es gibt halt so viele andere Dinge, die Leute ja auch lernen müssen. Soft Skills vermitteln, ohne dass die Erkenntnis da ist, dass Leute die brauchen, also so jemand zu zwingen: Du musst dir jetzt eine Kostenrechnungsvorlesung anhören, da kannst du die halt ein Jahr später mal fragen, was davon dann noch übrig geblieben ist.“*

(2) *„Und in der Wirklichkeit machst du nie Dinge alleine. Aufgaben sind nie klar umrissen, und sie ändern sich ständig.“*

(3) *„Ich bin kein so ganz ausgeprägter UML Freund. Das gäbe mir die Möglichkeit mit einer standardisierten Syntax, standardisierten Notation zu arbeiten. In Informationssystemen ist das wirklich nicht mehr so ganz hipp. In Embedded-Systemen, Real Time-Systemen, Safety-Critical, also wo es um Leib und Leben geht, ist das immer noch sehr, sehr verbreitet und wird auch recht formal gemacht, zum Glück. Aber ansonsten verwenden Leute halt alle Arten von Visualisierungen. Wenn ich Leute coache, dann sage ich Ihnen ja, stelle sicher, dass es eine klare Bedeutung dieser Symbolik gibt, also so eine Art Legende. Wenn es sie nicht gibt, ist das ein Fehler.“*

Anhang D

Zwischenumfragen

Die erste Umfrage wurde in der letzten Novemberwoche 2021 durchgeführt und somit nach mehr als der Hälfte der Entwicklungszeit der Services. Ziel der Umfrage war, die aktuelle Stimmung der Teilnehmenden zu ermitteln und direktes Feedback anzuregen.

Die Teilnahme an der Umfrage war freiwillig und anonym. Die Texte der Studierenden in den Umfrageergebnissen werden folgend unverfälscht dargestellt.

D.1 Einleitung

D.1.1 Verteilung der teilnehmenden Personen

An der Umfrage haben nur die Teilnehmer:innen der zu dieser Zeit aktiven Module FAE/DDD und IP teilgenommen. FAE/DDD wird folgend nur als FAE dargestellt.

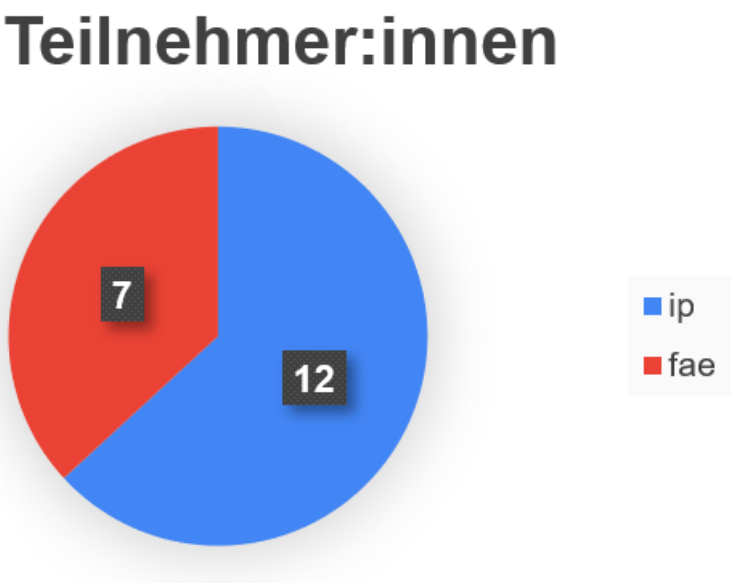


ABBILDUNG D.1: Verteilung der Teilnehmer:innen

D.1.2 Coding Skills

Die Selbsteinschätzung der Coding Skills der Teilnehmer:innen.

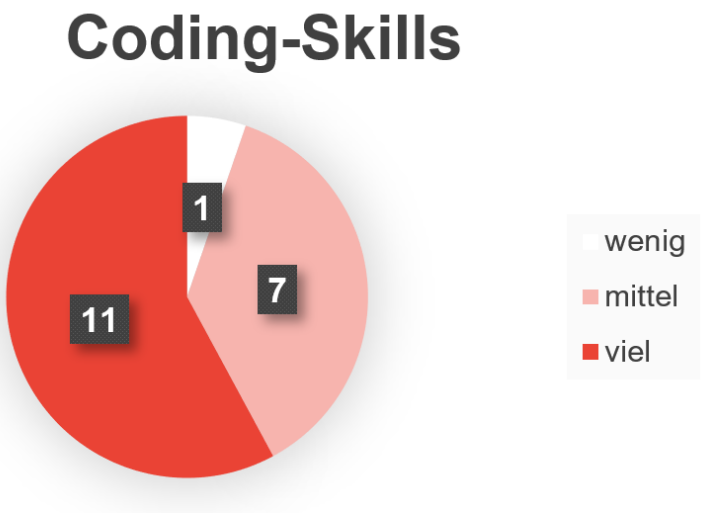


ABBILDUNG D.2: Selbsteinschätzung der Coding Skills

D.2 Motivation

D.2.1 Spaßfaktor des Projektes

Wie viel Spaß macht Ihnen das Projekt bisher?

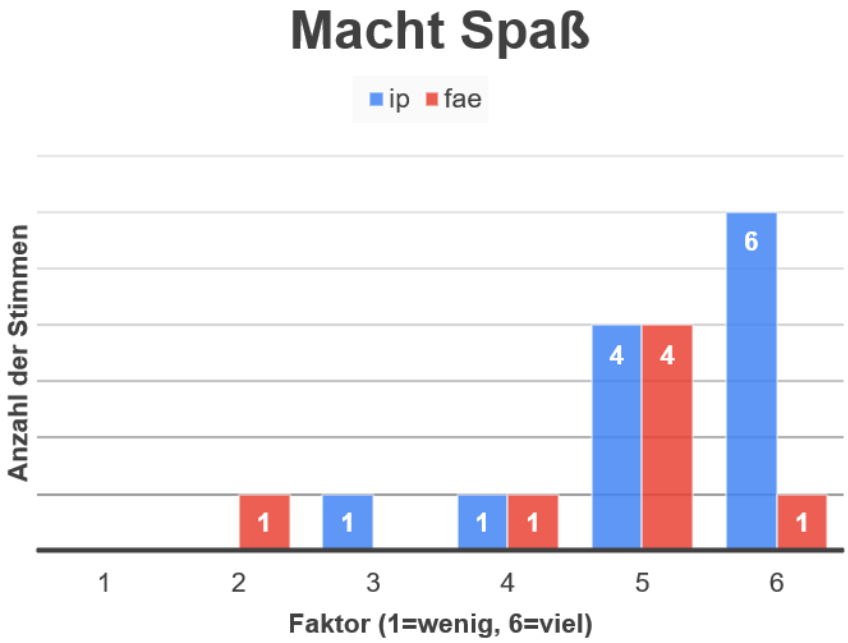


ABBILDUNG D.3: Spaßfaktor

D.2.2 Neues Lernen

Wie viele neue Dinge konnten Sie bisher im Projekt kennenlernen?

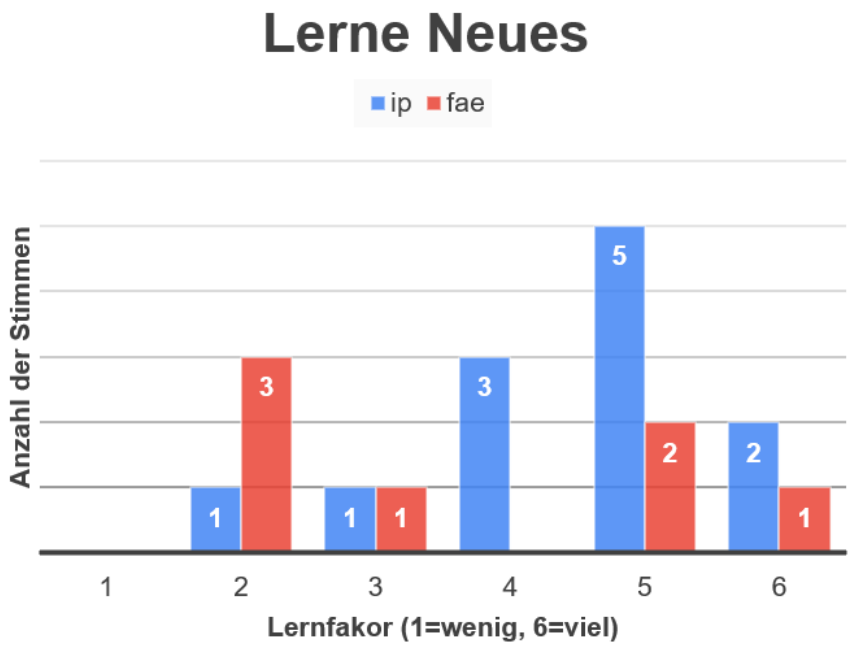


ABBILDUNG D.4: Einschätzung des neu Gelernten

D.2.3 Spaßbremsen (Generell)

FAE

- Motivation anderer Teilnehmer, kein „gemeinsames“ Ziel vor Augen
- Leute, die den Arbeitsfluss aufhalten + nicht ordentlich mitdiskutieren + Absprachen verwerfen ohne ordentlich Bescheid zu geben + problematische Charaktere auf persönlicher Ebene (von denen wir hier im Projekt zum Glück keine haben soweit ich weiß)
- Das die Decisions nicht gelesen werden. Ich denke, die könnten noch übersichtlicher dargestellt werden.
- Vor allem, dass ich einfach durch Arbeit, Stress und Zeitdruck den Kopf nicht so richtig dafür frei habe, mich stärker auf das Projekt einzulassen. Die Idee ist sehr cool, aber leider sind die Rahmenbedingungen für viele so, dass das Projekt nur noch mehr Druck erzeugt durch den damit verbundenen Zeitaufwand. Ich glaube, mehr Zeit würde da gut tun (also das Projekt über einen längeren Zeitraum strecken, um den Zeitdruck zu reduzieren).
- Fehlende Zeit zum vollständigen einsteigen; Zu groß
- Aufgabenverteilung am Wochenende

IP

- Sich ändernde Anforderungen
- Es gibt nicht wirklich Spaßbremsen. Evtl die doch recht hohe Komplexität, die teilweise für Verwirrung sorgt, aber das ist ja eigentlich auch das Interessante an der ganzen Sache.
- Eigene Demotivation, und auch oftmals Ablenkungen durch andere Fächer und Nebenjob
- Etwas zu wenig Koordination zwischen den Teams. Ein kleineres „System“ hätte die Möglichkeit gegeben, auch mal die anderen Services tiefer anzuschauen(vllt größer Lernerfolg auch mal zur Seite zu schauen), so versucht man nur fertig zu werden und das bestmöglich.
- Es gibt viele verschiedene Stellen an denen informationen liegen, und es gibt kein Benachrichtigungssystem bei neuen Informationen. Dadurch ist es sehr schwer aktuelle Informationen zu haben.
- Unklarheit und Chaos wie Sachen funktionieren sollen.
- Fehlende Kommunikation zwischen den Teams und daraus entstehende Änderungen an bereits implementierten Features.
- Die Komplexität und die damit entstehenden Fragen (hält sich in Grenzen muss nur geklärt werden und entsteht durch die Rollentrennung zwischen Developer und Gamedesignern)
- Zeitlich wurde es im CoCo Project Launch etwas knapp.
- Gibt nicht so wirklich welche, am ehesten wären es Diskussionen die etwas zu lang werden.
- Auf den Fortschritt anderer Services warten müssen

D.2.4 Spaßbremsen (Persönlich)

FAE

- Coder sind zu langsam
- Manche Entscheidungen sollten von vorneherein stehen, sodass wir als Architekten auch vorarbeiten können. z.B. wurde die Struktur der Events zwar frühzeitig festgelegt, dann jedoch infrage gestellt.
- Missverständnisse, Umwerfen von Plänen
- Schwierige Kommunikation, obwohl es an sich genug Kanäle dafür gäbe.
- Persönlicher Zeitmangel. Streckenweise könnte die Kommunikation aber auch etwas besser sein, vielleicht durch ein kurzes Meeting in den Nicht-Workshop-Wochen.

- Extreme Spontanität, 24/7 in Discord erreichbar sein
- Absprachen mit anderen Team (unklare API Specs)

IP

- Sich ändernde Anforderungen
- Am meisten behindern mich andere Externe Projekte, die meine Zeit fressen. Im Projekt selber ist es maximal die Kommunikation, die das Informationen bündeln etwas erschwert, aber selbst die läuft eigentlich sehr gut. Hätte ich jetzt keinen Vorschlag, wie man die noch weiter verbessern könnte.
- Siehe Spaßbremsen
- Zu wenig Erfahrung mit Spring.
- Die Kommunikation zwischen IP und FAE und der IP Gruppe untereinander.
- Kafka aufsetzen. Die Events sind teilweise noch nicht definiert, das letzte Video ist etwas chaotisch, da Philipp krank wurde
- Zeit
- Immer wieder ein bisschen eine Art von „Ich weiß nicht genau wo ich anfangen soll“, aber es wird langsam besser je mehr ich lerne und mache.
- Fehlende Erfahrung mit Kafka

D.2.5 Anregungen für die Dozenten

FAE

- Ich finde, die Inhalte werden gut gezeigt. Die Impulse sind da und die Videos auf dem Archi-Lab Kanal sind sehr informativ.
- Wie gesagt, mehr Zeit wäre gut. Wenn man z.B. den Hackathon erst Ende Januar oder in der Woche nach den Prüfungen macht, kann das Projekt besser reifen und man kann mehr Workshops abhalten. Für den bestehenden Rahmen finde ich das schon gut gelöst.
- Schwierig, weil man das meiste aus dem Bachelor kennt.
- Als FAE Teilnehmer ist es schwer zu beurteilen, da für mich das allermeiste schon bekannt ist bzw. ich sogar schon damit gearbeitet habe (mit der Ausnahme Kafka). Obwohl ich schon Erfahrung mit chaotischen Projekten habe und im kleinen Rahmen sowohl die PO als auch die Architektenrolle schon ausgeführt habe ($\tilde{6}$ Leute), liegt für mich in der Organisation/ dem Schaffen von Transparenz der größte Lerneffekt. Gewünscht hätte ich mir mehr Fokus auf Möglichkeiten, wie man Decisions verwalten kann/welche Tools es da gibt/ wie das in großen

Projekten durchgeführt wird (bei denen es tatsächlich mal richtig funktioniert) - Gitpages sind eine gute Option und es ist interessant sich dazu die Implementierung anzuschauen, allerdings fehlen da einige sehr wichtige Features (z.B. Markierung von Änderungen), von daher wäre es interessant gewesen mehr über diese Seite der Architektentätigkeit zu erfahren

- Konkret klarmachen, was überhaupt gelernt werden soll
- Mehr Wissens-Inputs, wie es bei den letzten beiden Terminen der Fall war.

IP

- Vorstellung relevanter Techniken (Designpatterns etc.)
- Schwierig zu sagen. Das Ganze ist eigentlich schon genauso perfekt organisiert und aufgebaut, wie es sein kann. Ich bin jetzt natürlich ein wenig in der Position das ich von mir selber behaupten würde recht „erfahren“ zu sein was Teamarbeit und Coding in einem solchen Kontext betrifft, deswegen kann ich das schlecht beurteilen. Das, was ich jedoch in diesem Projekt mache, habe ich so in der Form selber auch noch nicht gemacht, was natürlich bedeutet, dass ich Neues lerne. Insofern läuft also alles richtig!
- Vielleicht live coding events für alle Interessierten, alle zwei Wochen in Präsenz, ein paar Stunden zu einem bestimmten Thema z.B. für dieses Projekt wie Testing mit Spring oder Token Authentifizierung etc.
- Den Einsatz - für den Studenten neuer - Technologien fordern. Das bedarf es aber gar nicht. Wenn alles neu ist, macht es gar nicht so viel Spaß, wie wenn man einiges schon kennt und einiges neu dazu lernt.
- Verschiedene Themen als Video vorproduzieren, beispielsweise Testing.
- Ich finde, viele neue Sachen muss man in diesem Projekt nicht lernen, da (für mich) Kafka z.B. schon neu ist und es zeitlich gut ist Sachen, die man schon kennt zu benutzen.
- Out of scope - Realtime communication bei einem einzelnen Service mit einbringen?
- Insgesamt finde ich die Themenwahl super, würde mich aber über etwas mehr Stoff zum drum herum wünschen (Datenbank, Konfiguration o.ä). Wobei das schwierig ist mit Technologie Freiheit.
- Der Dozent bzw. die Veranstalter geben sich große Mühe. Eventuell die Komplexität etwas heruntersetzen.

D.3 Service-/Projektstand

D.3.1 Qualitätsempfinden des Services

Wie empfinden Sie die aktuelle Qualität Ihres Services?

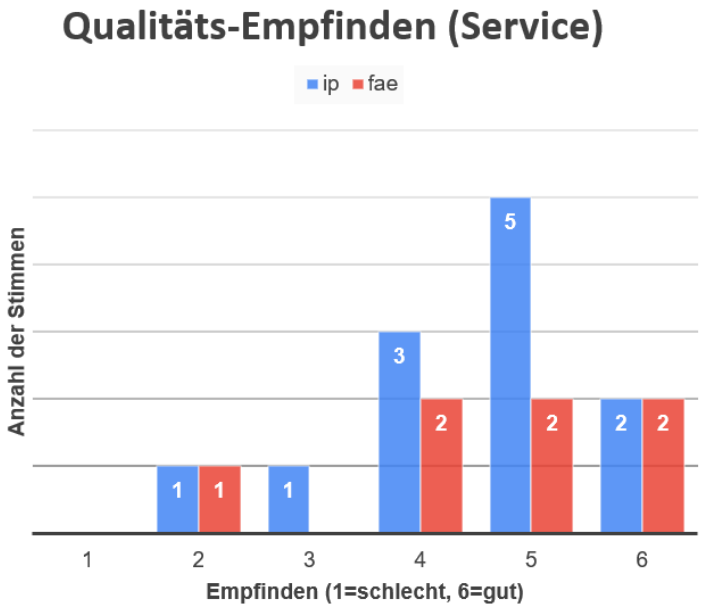


ABBILDUNG D.5: Qualität des eigenen Services

D.3.2 Qualitätsempfinden des Projektes

Wie empfinden Sie die aktuelle Qualität des Projektes?

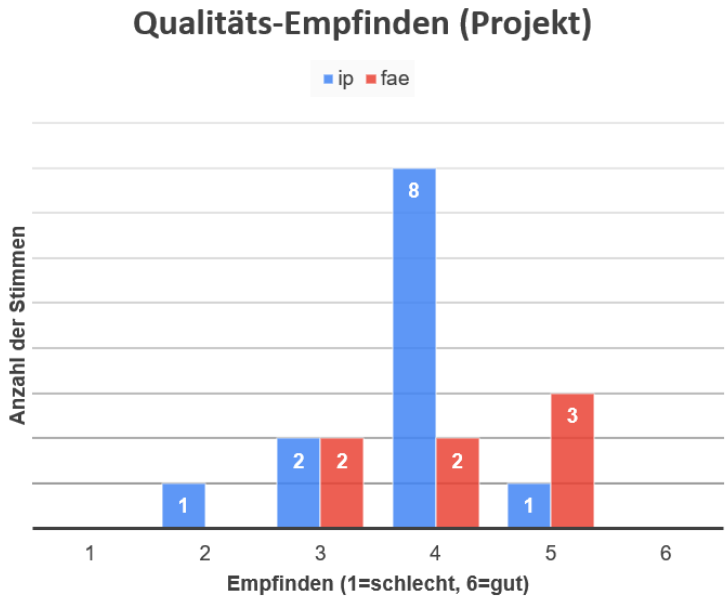


ABBILDUNG D.6: Qualität des gesamten Projektes

D.3.3 Verbesserung der Servicequalität

Was genau müsste in „Ihrem“ Service passieren, damit die Qualität besser wird?

FAE

- Tests, technische Konzepte des Frameworks und der Sprache müssen den Codern bekannt sein,
- Schnellere Klärung von Schnittstellen + weniger Umschmeißen von Plänen (oft müssen wir Entscheidungen komplett umbauen, je später man das macht, desto chaotischer wird es)
- Der Aufbau des Projekts könnte sich strenger an die Konventionen des DDD, die im Vorfeld bestimmt wurden, halten.
- Ich glaub, die Jungs machen das schon ganz gut. Für bessere Qualität müsste man mehr Reviews durchführen. Das ist dann wieder eine Zeitfrage.
- Nichts

IP

- Läuft schon alles super!
- Ich finde es selber schwierig, die Qualität einzuschätzen. Zudem fehlt ein wenig das „Testing“ der Service-Schnittstellen. Ich mag ungern etwas Coden, was dann erstmal nicht wirklich getestet werden kann. Und damit verringert sich für mich auch die Code-Qualität.
- (Ich muss als Programmierer bessere handwerkliche Fähigkeiten entwickeln.) Ich gehe davon aus, dass alle anderen im Team sich auch bestmöglich anstrengen.
- Die Umsetzung von Tickets sollte vorher in einer kleinen Runde besprochen werden und Ideen für die Umsetzung gesammelt werden. Ich denke, das würde zu besseren Implementierungen führen.
- Teammitglieder müssten aktiver sein und sich mehr einbringen, bzw. die Sache als solche ernster nehmen.
- Müsste weiter entwickelt werden, da das Konzept noch nicht ganz steht
- Integrationstests
- Einfach mehr Zeit reinschütten. Ich lerne eigentlich viele neue Wege die Sachen besser zu machen, aber das Recherchieren dazu ist an Stellen mühselig.
- Mehr Erfahrung mit dem Framework und Kafka.

D.3.4 Verbesserung der Projektqualität

Was genau müsste im Gesamtprojekt passieren, damit die Qualität besser wird?

FAE

- Erst MVP, dann Features hinzufügen
- - Deadlines für APIs und Events - APIs und Events müssen zur Deadline zentral verlinkt sein - mehr, zeitnähere und bessere Dokumentation mit klar definierten Sammelpunkten
- Teams sollten sich öfter außerhalb der Termine absprechen, um große Entscheidungen frühestmöglich treffen zu können
- Mehr Struktur vorgeben, damit nicht jede Kleinigkeit in endlose Diskussionen...
- Wir hätten ein Meeting mit allen gebraucht, evtl. Tag 3 des Event Stormings wo die Features nicht nur anhand dessen geplant/besprochen wurden, was cool für das Game wäre, sondern auch was technisch möglich und in unserem Scope ist. Außerdem haben nicht alle mitbekommen, wenn irgendwelche Features aufgrund der Umsetzbarkeit / des Scopes rausgestrichen wurden. Im Gespräch mit z.B. den Gamemastern kam heraus, dass diese einige Konzepte immer noch fest geplant haben, obwohl das nicht implementiert wird (z.B. Adminzugriff auf Gamevalues) und bei anderen Features (z.B. Fog od War) sich keine Gedanken um die technische Umsetzbarkeit mit unserem Scope gemacht haben. Außerdem bräuchte es eine klare Kommunikation, was genau vom originalen Gameplan abweicht (uns fehlen viele Decisions, da das nicht sehr gut kommuniziert wurde, dass wir da für jede noch so kleine Änderung was brauchen), außerdem müsste man sich überlegen, wie die Leute Änderungen gut sehen können (z.B. frisch bearbeitete Decisions etc. flashy markieren)

IP

- Besser definierte (moderierte) Anforderungen der Kommunikation zwischen Services, um zu verhindern, dass einzelne Teams ihre Implementierung zu oft anpassen müssen, um Anforderungen anderer zu erfüllen.
- Zurzeit ist alles ein wenig unübersichtlich. Die Wikis und Dokus der Services sind alle noch nicht richtig gepflegt, was es schwierig macht schnell die eine Information zu finden die man gerade sucht.
- Ich habe mich bislang weniger mit den anderen Services beschäftigt. Nur das Nötigste, um die API-Punkte herauszufinden, etc. Diese Dinge sind dafür aber alle gut strukturiert und ordentlich.

- Decision Log und Team Wiki's teilweise uneinheitlich, unübersichtlich und nicht aktuell, dazu kommt suboptimale Kommunikation durch POs. Beim Projektstart festeres Grundgerüst festlegen, das unumstößlich ist. (Auch wenn es weniger realitätsnah ist, ermöglicht es einen besseren Lernerfolg im Ganzen). Und darauf die must have und optional features der Services aufbauen."
- APIs sind häufig nicht mehr ganz aktuell und die Begründungen für den Aufbau der APIs müssen häufig nachgefragt werden. Eine zentrale Stelle für die Ground-Truth mit Begründung würde hier helfen.
- Die Kommunikation zwischen den verschiedenen Teams erhöhen.
- Die Spezifikation müssen früher fertig sein
- Integrationstests - Vielleicht SDKs für die Player entwickeln und testen
- Kann ich leider nicht viel zu sagen, aber insgesamt ist es mit der API doku + dem Decision Log schon ziemlich aufgeräumt.
- Schnellere Erstellung der API und Event Spezifikationen

D.4 Kommunikation

D.4.1 Kommunikation im Service

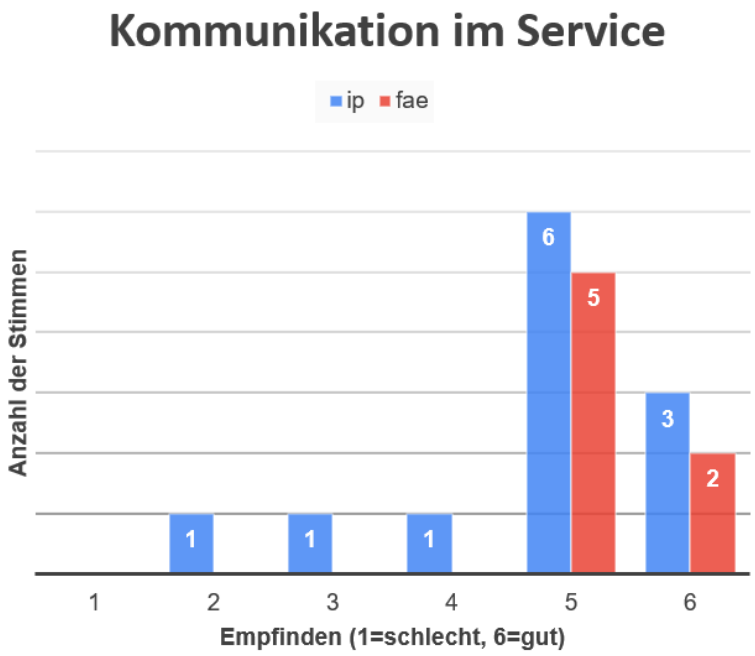


ABBILDUNG D.7: Kommunikation im Service

D.4.2 Kommunikation im Projekt

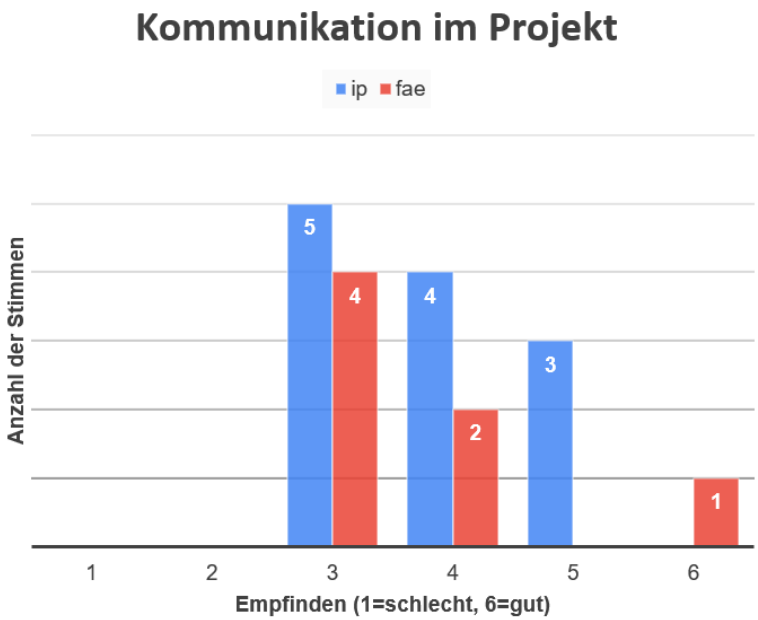


ABBILDUNG D.8: Kommunikation im gesamten Projekt

D.4.3 Verbesserungsvorschläge der Kommunikation

Was könnten wir tun, damit die Kommunikation besser wird (sowohl im Team wie auch im Gesamtprojekt)?

FAE

- Nichts, das liegt eher an der natürlichen Silobildung. Aktiv Fallstricke aufstellen, die zur Kommunikation zwingen, wäre eine Option - aber nicht sinnvoll.
- Mehr Transparenz durch mehr Decisions und evtl. in den FAE-Meetings eine halbe Stunde, bei der nicht nur die Motivation des Teams angesprochen wird, sondern auch, was genau inhaltlich (aka an Features) schon fertiggestellt wurde bzw. in Arbeit ist / für den aktuellen Sprint geplant ist
- Mehr Gespräche mit der ganzen Gruppe, da die Projektowner noch nicht gut genug geübt sind darin.
- Wie gesagt, häufigere, dafür kürzere Meetings. Es sollte jede Woche die Möglichkeit geben, Dinge anzusprechen und die Todos für die Woche bis zum nächsten Meeting zu klären.

IP

- Wie oben schon erwähnt, ich habe keine Idee wie das Ganze mit Blick auf die Kommunikation verbessert werden könnte. Läuft so wie es läuft, schon recht gut.
- Im Team, die Kommunikation läuft recht reibungslos. Über das Gesamtprojekt hinweg würde ich behaupten ebenfalls. Die Kommunikation erfolgt eher über die FAE Leute, was aber super funktioniert.
- Team: FAE mehr in den Entwicklungsprozess integrieren, z.B. Code Review durch FAE. Dadurch sind alle im Team ungefähr auf demselben Stand und FAE kann früher gegensteuern bei fehlerhaften Entwicklungen. (Bei uns sind einzelne aus FAE sehr wenig involviert) Gesamtprojekt: Siehe Qualität Projekt. So etwas steht und fällt mit der Kommunikation.
- Regelmäßige (zwei-wöchentliche?) Treffen zwischen den Teams, falls es keine Fragen oder Probleme gibt kann man es ja immer noch nach 10 Minuten wieder beenden.
- Es müsste mehr Kontrolle geben. Die Arbeit im Team müsste transparenter nach außen sein und dürfte nicht als eine Art Blackbox für die Betreuer stattfinden. Beispielsweise sollte über den Discord Kanal des Servers kommuniziert werden, statt über einen eigenen Server. Das könnte ggf. schon den ein oder anderen aufwecken.
- Einen festen Termin festlegen in dem sich, zu mindestens ein Teil der verschiedenen Teams trifft, und sich über den aktuellen Stand austauscht. Eine Suchfunktion im Decision Log hinzufügen und diesen leichter zu navigieren machen, damit alle auf demselben Stand sind.
- Nicht viel, finde die Aufteilung des Discords aber sehr gut.
- Ausführliche Diskussionen in GitHub issues. Vielleicht auch regelmäßige Retrospektiven veranstalten.
- Bei Anforderungen an andere Teams, die essenziell für das Funktionieren des eigenen Service / Deployment sind, sollte irgend eine Rückmeldung kommen, ob es gelesen und verstanden wurde, oder ob es noch Fragen / Kritik gibt.

D.4.4 Direktes Feedback an die Dozenten

Was möchtest du den Dozenten noch mitgeben - Ideen, Kritik, Lob, Anregungen?

FAE

- Super Projektidee mit einer Menge Potenzial. Das Projekt selbst könnte aber länger laufen. Vielleicht auf zwei IPs aufteilen, das erste IP kann Greenfield machen und das zweite Brownfield?

- Ideen/Anregungen/Kritik sind ja schon meinen Antworten zu entnehmen, von daher möchte ich mich an dieser Stelle für das Interesse der Organisatoren (Hr. Bente / Philipp) am Projekt und die zeitliche Verfügbarkeit/Spontanität bedanken. Macht Spaß zu arbeiten, wenn man das Gefühl hat, dass jeder ein ehrliches Interesse daran hat und angeregte Diskussionen entstehen (dies gilt auch für die meisten Kursteilnehmer).
- Ich denke, es könnte sinnvoll sein, die Architekten noch mehr in die Entwicklung einzubinden. Generelle Architektur-Entscheidungen werden meistens von außerhalb (Vorgaben der anderen Services) bestimmt. Dabei bleiben dann nicht viele weitere Entscheidungen. Sodass hauptsächlich die Rolle eines Product Owners (was nicht unbedingt schlecht ist) bleibt.
- Ist ganz gut so. Wesentlich besser als Interaction Design.
- Grundsätzlich eine sehr gute Idee und ich glaube, dass ihr da in der Zukunft noch einiges an Potenzial durch Optimierung herausholen könnt. Auch wenn jetzt im ersten Versuch noch manches nicht ganz rund läuft, lernen wir alle schon sehr viel Nützliches. Weiter so (Daumen hoch Symbol)
- Sehr cooles Projekt / Modul!

IP

- Sehr gute Organisation, klare Aufgabenteilung, direkte Kommunikationswege auch „nach oben“, Fragen und Anliegen werden schnell und zuverlässig behandelt und gelöst. Ich bin sehr ein Teil dieses Projektes sein zu können, sehr interessant und relevant finde ich!
- Sehr cooles Projekt, wenn möglich beibehalten. Es gibt im Studium nichts Vergleichbares. Auch das übergreifende mit mehreren Studiengängen. Vllt heiße Phasen etwas anders planen oder noch früher im Semester anfangen, um mehr Fokus auf das Projekt zu haben. Aktuell: Die ersten Sprints bis die APIs standen, vergleichsweise wenig Arbeit für uns im IP. Und dann, wenn alle anderen Fächer Praktikumsabgaben haben, muss entwickelt werden. Wir haben so früh angefangen und könnten teilweise schon mit der Entwicklung fertig sein, bevor die anderen Praktika anfangen hätten, wenn die APIs früher gestanden hätten. Dann würde man auch mehr Zeit für die Gesamtprojekt-Integrationstests besitzen. Dies ist keine Kritik an der Planung, nur eine Anregung, sollte dieses Projekt nochmal angeboten werden.
- Ein Spiel fühlt sich leider nicht wirklich wie die beste Einsatzmöglichkeit für eine Microservice-Architektur an.
- Das Event-Storming zu Beginn des Projektes war eine sehr schöne Idee und kann man gerne wiederholen. Ansonsten, die Kommunikation zwischen den Teams verbessern.

- Freibier und ich finde das Projekt interessant, aber etwas zu komplex. Es ist immer jemand für Fragen da und die Kommunikation ist auch gut gestaltet(Discord). Die Wiki-Seite ist eine sehr gute Idee für alle zentralen Infos.
- Super Projekt, kommt der Realität sehr nah. Das selbstständige Arbeiten fördert das Organisationstalent, bringt aber je nach Typ Mensch viel Stress einher. Elastic Leadership ist ein Buch, welches diese Dinge sehr gut erklärt.
- Ich bin mega happy mit dem Projekt und dass ich die Chance habe, viel Zeit in ein Thema zu stecken, was mich wirklich interessiert. Fand das Event in Mülheim auch mega gut, in Präsenz über sowas zu diskutieren ist einfach super. Ich würde mir an manchen Stellen zwar ein bisschen mehr Leitung wünschen (DevOps-Sachen). Insgesamt lerne ich aber extrem viel, von dem ich den Eindruck habe, dass es mich tatsächlich ein Stück weit auf einen Beruf in der Software-Entwicklung vorbereitet. Deshalb finde ich es auch nicht schlimm, dass das Projekt manchmal aneckt / nicht alles perfekt läuft, das tut es in der Realität ja auch nicht. Ich bin auf jeden Fall super gespannt, was am Ende hierbei rauskommt und hoffe natürlich, dass es läuft :).
- Es macht viel Spaß und vieles wird ziemlich realitätsnah und sehr modern gehalten. Das kann man nicht immer an der TH sagen, und es ist schön, gegen Ende des Studiums noch einen etwas erfrischenden Eindruck der TH zu bekommen. Großes Lob an alle! :) Ideen: 01000110 01110010 01100101 01101001 01100010 01101001 01100101 01110010

Anhang E

Abschlussumfrage

Die Teilnahme an der Umfrage war freiwillig und anonym. Die Texte der Studierenden in den Umfrageergebnissen werden folgend unverfälscht dargestellt.

E.1 Einleitung

E.1.1 Über welches Modul haben Sie am Dungeonprojekt teilgenommen?

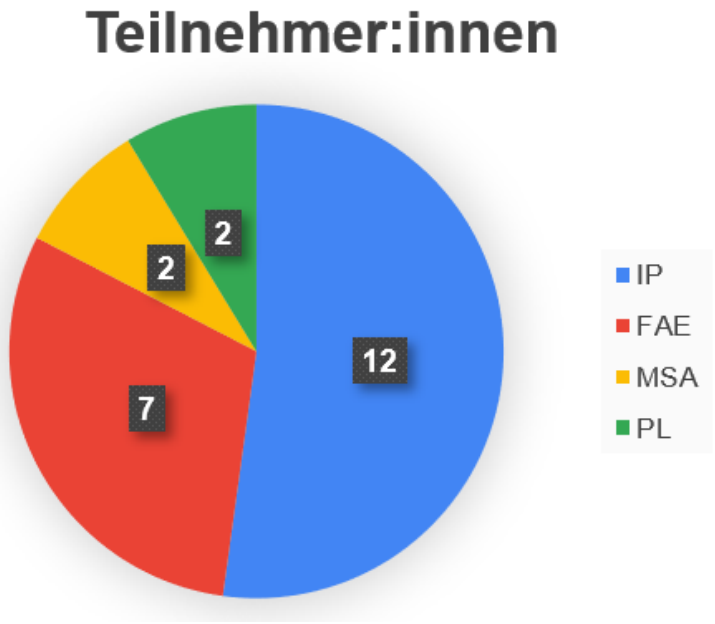


ABBILDUNG E.1: Teilnehmer:innenverteilung

E.1.2 Hat Ihnen das Projekt Spaß gemacht?

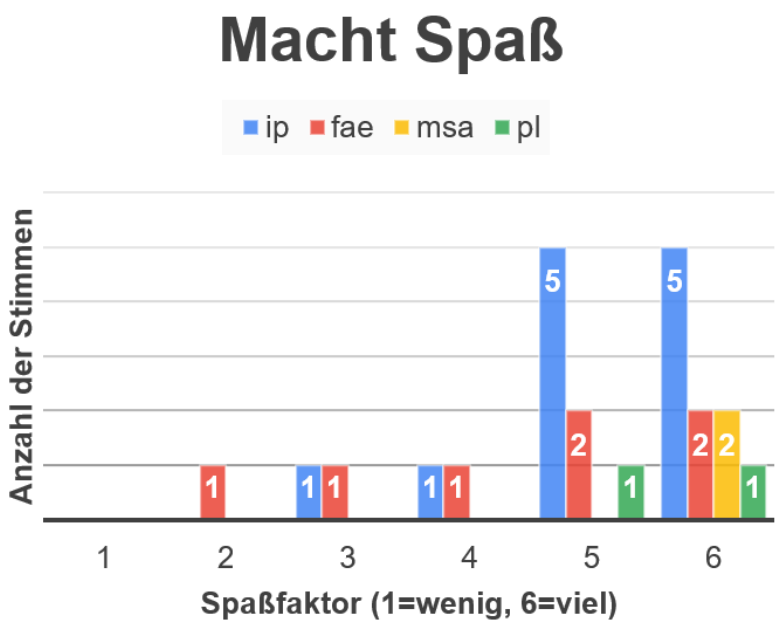


ABBILDUNG E.2: Spaßfaktor im Projekt

E.1.3 Haben Sie so viel gelernt wie erwartet?

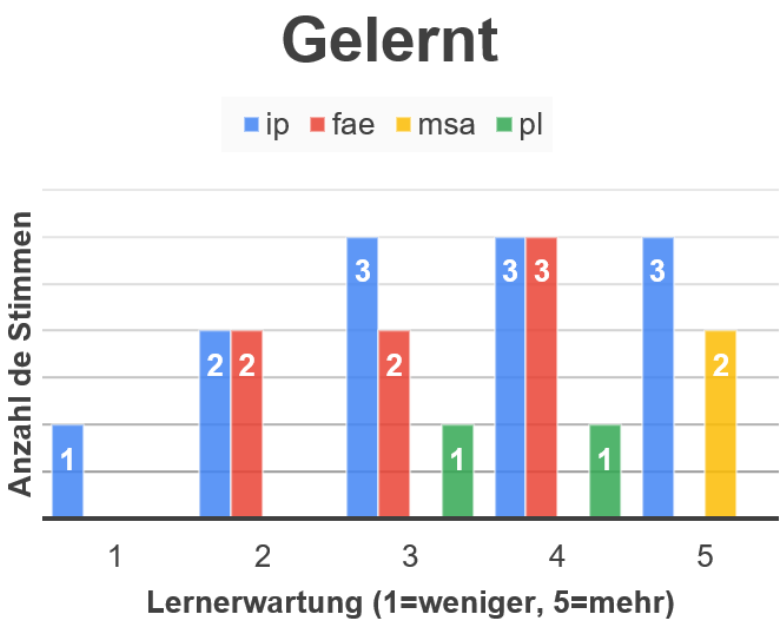


ABBILDUNG E.3: Lernerwartung

E.2 Fragen zum Service/Arbeit

E.2.1 Sind Sie mit der Qualität Ihres eigenen Services zufrieden?

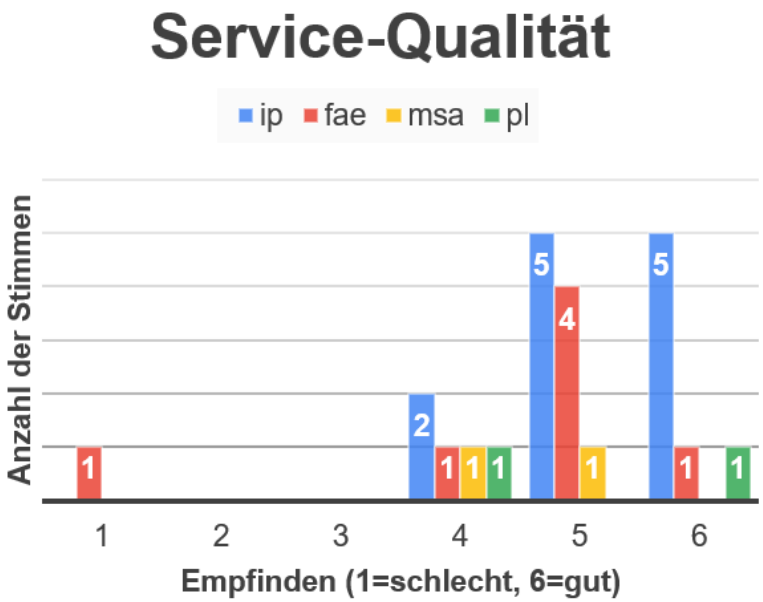


ABBILDUNG E.4: Qualitätsempfinden des Services

E.2.2 War der Umfang Ihres Services passend für die Gruppengröße?

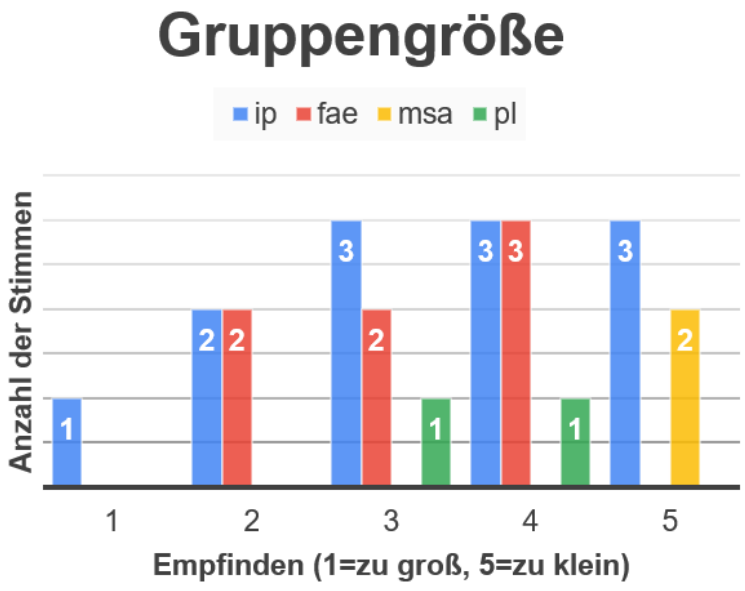


ABBILDUNG E.5: Gruppengröße des Serviceteams

E.2.3 Wie würden Sie die Kommunikation im Service-Team beschreiben?

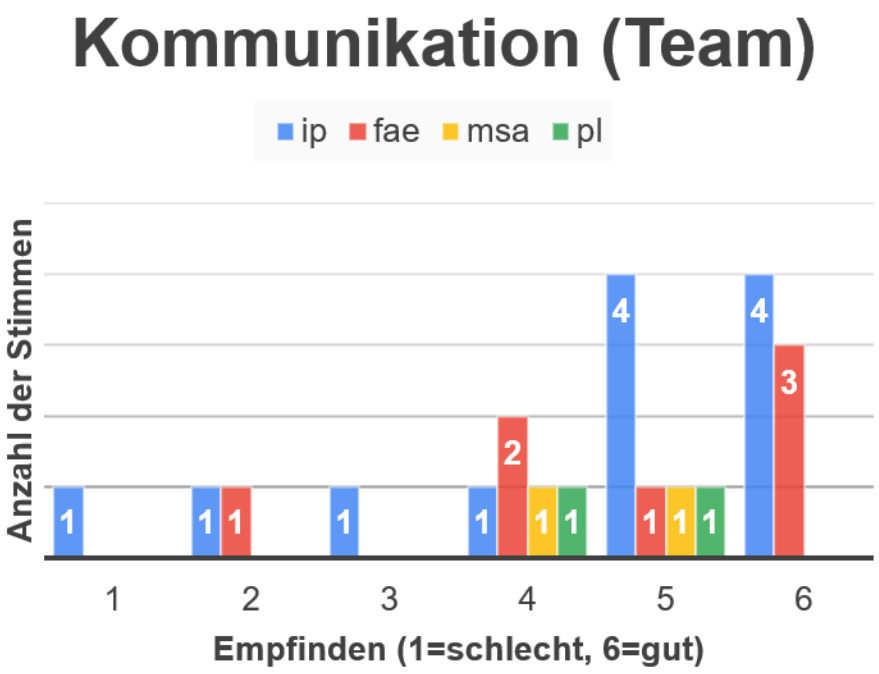


ABBILDUNG E.6: Kommunikation im Service

E.2.4 Was war die größte Herausforderung in Ihrem Service?

FAE

- Command Obfuscation, Zuordnung von Commands zu Spielern und Runden
- Absprachen mit vielen Personen
- Kommunikation mit den anderen Services
- Ein Gruppenmitglied hatte sich nicht an die Anforderungen gehalten und hatte dann keine Zeit es »geradezubiegen«.
- Team Robot hatte sehr viel zu tun, was zum Glück gut geklappt hat, da das Robot-Team sehr früh, sehr viel Zeit investiert hat (und schon pre finale Abgabe easy die Zeit für die Credits reingesteckt hat)
- Das Arbeiten ohne Testdaten

IP

- Wir als DevOps mussten viel hin und her und uns somit an vielen verschiedenen Stellen neu "reindenken".
- Zu viel Arbeit auf zu wenigen Leuten

- Sicherheit? Eigentlich alles so ein bisschen (DevOps). War halt komplett neu lernen.
- Fehlendes Testing durch nicht fertige, andere Services und somit Unwissenheit über eigene Richtigkeit
- Kommunikation mit anderen Services
- Das Team (Game). Schlechte Kommunikation, mangelnde Bereitschaft, mangelnde Kompetenz, fehlendes Verantwortungsgefühl, kein Ownership für seine eigenen Aufgaben.
- Economy Einschätzung und testen des Services
- Sinnvolle lokale Orchestrierung der anderen Services
- Die anfänglich Struktur.

MSA

- Sich im Gesamtkonstrukt zurechtzufinden
- Organisatorisch: plötzliches Wegfallen einer Person; Technisch: das Mocken für Kafka

PL

- Die Generierung der Gameworld

E.2.5 Sind Sie mit der Qualität des eigenen Players zufrieden? (bitte überspringen, falls kein Player entwickelt wurde)

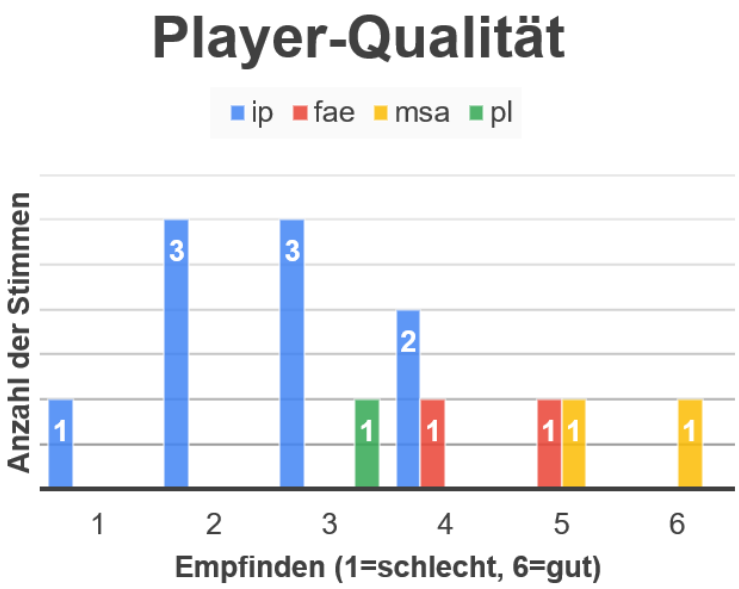


ABBILDUNG E.7: Qualitätsempfinden des eigenen Players

E.3 Fragen zum gesamten Projekt

E.3.1 Sind Sie mit dem Ergebnis des Projekts zufrieden?

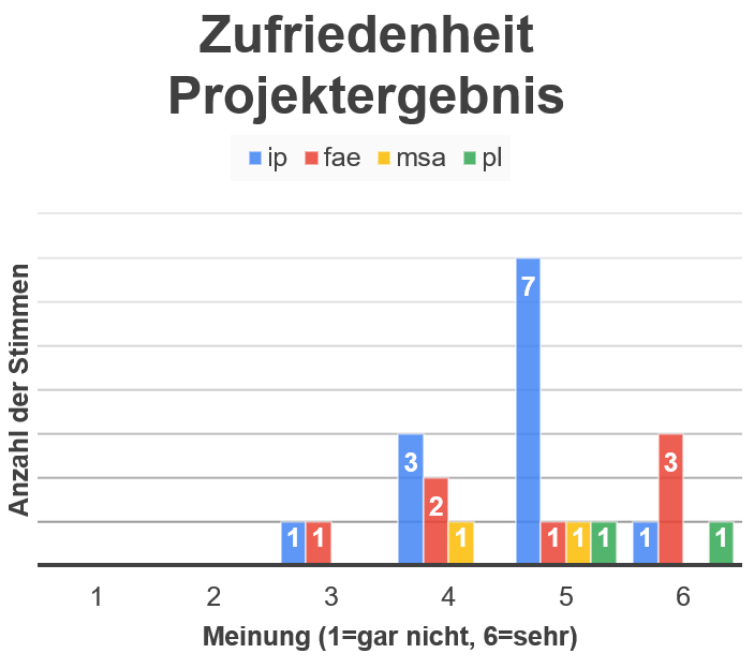


ABBILDUNG E.8: Akzeptanz des Projektergebnises

E.3.2 Wie passend war aus Ihrer Sicht der Projektumfang?

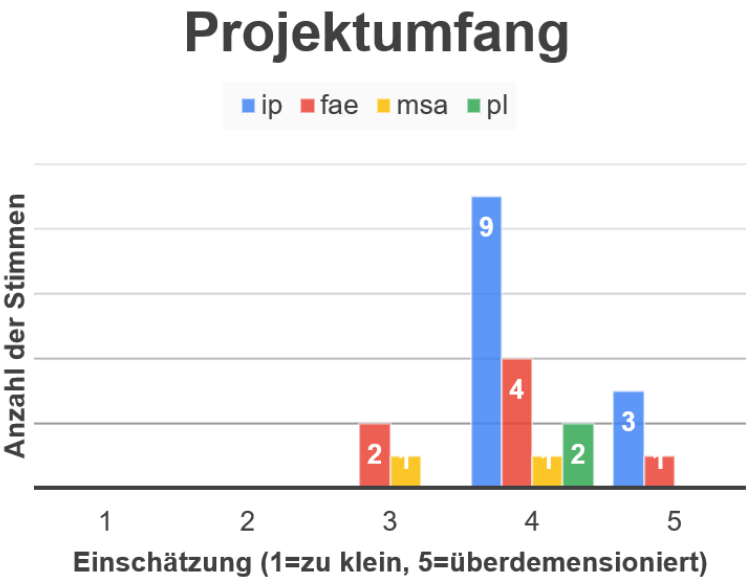


ABBILDUNG E.9: Einschätzung des Projektumfangs

E.3.3 Wie würden Sie die Kommunikation im gesamten Projekt beschreiben?

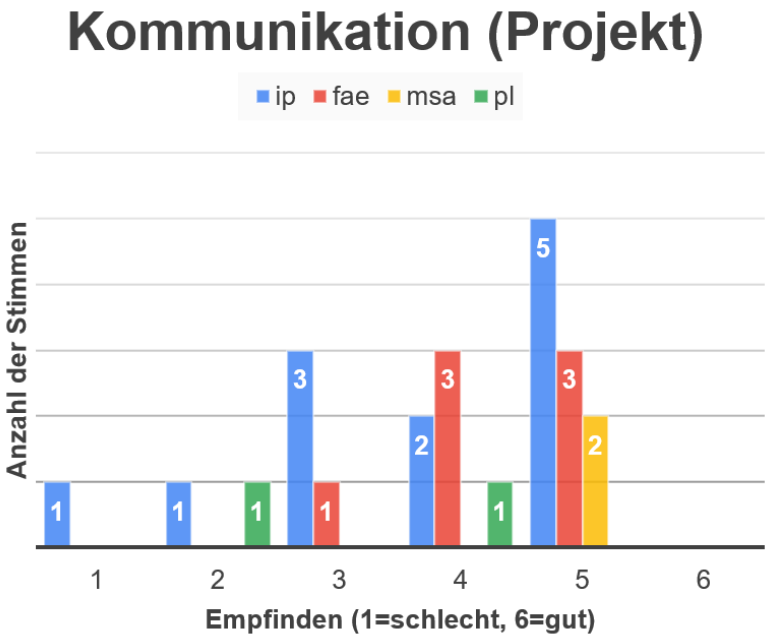


ABBILDUNG E.10: Einschätzung der Kommunikation im Projekt

E.3.4 Welche Punkte würden Sie bei der Kommunikation konkret verändern?

FAE

- Regelmäßige Team-übergreifende Status-Updates.
- Mehr Struktur von Anfang an vorgeben
- Teilweise wurden wichtige Details in Channels von einzelnen Services kommuniziert und gingen für viele verloren.
- Mehr Fokus auf den Entscheidungen. Zentrale Stelle, über die die Kommunikation läuft. Evtl. ein Projektweiter "Taskboard"
- Verlagerung der Absprachen noch mehr zu zwischen den Projektownern und keine vereinzelten Einzelabsprachen mehr
- Gameplay-Team muss in den Verlauf des Projekts eingebunden werden, Gamelog-Team hätte sich von Tag 1 an überlegen müssen, was sie eigentlich von den ganzen Services wollen und nicht erst im Januar ankommen und feststellen, dass ihnen die Info in den Events nicht reicht. TLDR: Jedes Team hätte von Anfang bis Ende mitdenken/mitarbeiten müssen.

- Stärkere Formalisierung des Feedback-Prozesses, vor allem bzgl. der APIs: häufigere, regelmäßige Meetings zur Rücksprache bzgl. Anforderungen an die anderen Services, Vereinheitlichung der Dokumentation, fester Ort für Issues der Schnittstellen (gerne auch aggregiert)

IP

- Es sollten mehr Standards und Spezifikationen explizit geklärt und festgelegt werden.
- Kein Discord, FAE/DDD haben als PO sehr wenig getan bei uns im Service
- Spezifikationen einheitlich definieren und breaking changes besser kommunizieren
- Synchronere Arbeitszeiten? Kernarbeitszeiten?
- Wöchentliche verpflichtende Meetings, in denen ein Ansprechpartner je Service da ist.
- Bessere Absprache bei API und Verpackung von Daten
- Mehr Meetings, eventuell das Gesamtprojekt agil managen, mehr Sprint Meetings usw., damit man sich besser abstimmen kann. So, dass jedes Team kurz den aktuellen Stand darlegt
- Wöchentliche Sprints in den Teams mit Anwesenheit eines Ädminsïn den ersten 5 Minuten.

MSA

- Bessere Dokumentation der Commands via GameService OpenAPI, allgemein mehr Ownership was die Doku anging

PL

- Noch mehr Ownership festlegen. Wer ist der Owner von Robots? etc.

E.3.5 In welcher Weise hat das Projekt Ihre Sicht auf die Bedeutung von Kommunikation in Projekten verändert?

FAE

- Schnelle Kommunikation durch Chats wie Discord ist gut. Wichtige Informationen und Entscheidungen sollten jedoch zentral dokumentiert werden.
- Komunikation is the key! (Aber das wusste ich vorher auch schon, also eigentlich gar nicht)
- Keine Veränderung. Ich wurde in dem Wissen, dass Kommunikation Gold wert ist, nur wieder einmal bestätigt ;)

- Hätte gedacht, dass das besser klappen würde. Braucht wohl mehr Vorgaben und Interaktion/Integration.

IP

- Man braucht klare Kommunikationssysteme, z.B. einheitliche Ticket Tools etc.
- Es ist wesentlich anstrengender als angenommen
- Es kann gar nicht zu viel Kommunikation geben
- Probleme und eigenes Empfinden schneller, früher und direkter ansprechen, bevor sich eine Situation anstaut oder bildet.
- Ich habe gemerkt, dass Planung von Schnittstellen sehr wichtig ist
- Die große Bedeutung von Kommunikation in "großen"komplexen Softwareprojekten ist wichtiger denn je.

MSA

- Hat mir wieder gezeigt wie wichtig sauberes Projektmanagement und häufige Check-ins sind

PL

- Daily Stand-ups und Retrospektiven sind sehr hilfreich!

E.3.6 War die Roboter-Spiele Idee passend für das Projekt und um Microservices zu vermitteln?

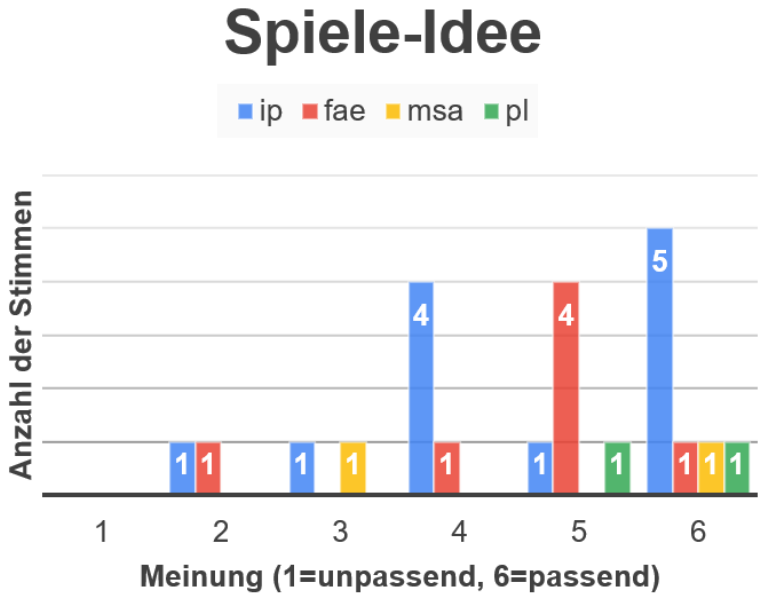


ABBILDUNG E.11: Akzeptanz der Spieleidee

E.3.7 Was wären weitere Ideen, welche Sie für ein Projekt solchen Umfangs und Themas behandeln würden?

FAE

- Websockets
- Milestones mit festen Kontrollen der Stände, mehr Talks, Workshops
- Die Idee ist an sich gut. Wie es verbessert werden könnte, kommt auf der letzten Seite,
- Sehr früh Awareness schaffen, dass jeder Service von Day one mitziehen muss (siehe Gamelog)

IP

- Load Balancing, Scaling der Services, Health-Monitoring
- Die Spiel-Idee lässt sich in beliebigen Szenarien verwenden. Von Cowboys im Wild West Setting bis zu einer Bürosimulation wo ein Spieler/Angestellter rumläuft und Aufgaben abarbeitet. Ich würde kein realistisches Thema wie online Shop nehmen. Spiele sind beliebig erweiterbar oder reduzierbar.
- Keine konkrete Idee, aber es braucht eine Idee, bei der die einzelnen Services deutlich unabhängiger voneinander sind.
- Spiele sind eine super Idee, da sich andere Projekte schlechter nutzen lassen. Agar.io könnte auch eine super Vorlage sein.
- M.M.n. wäre ein realitätsnäheres Projekt sinnvoller gewesen. Dazu hätte man sich die Unternehmen ansehen können, die Kafka produktiv im Einsatz haben. Das Konzept des selbst entworfenen Spiels hat deutlich mehr Komplexität in das Unterfangen gebracht.

MSA

- Für gemeinnützige Vereine Software entwickeln, die gebraucht wird. (Kostenlos - Dafür aber ein Product Owner, der das Produkt wirklich benutzen wird)
- Shop, Managementsysteme, Social-Media Plattform

PL

- Das Projekt so planen, dass im Endeffekt weniger Events published werden müssen. Also in unserem Fall, den Gedanken vom Fog of War weglassen.

E.3.8 Welche Nachteile von Microservices haben Sie in diesem Projekt wiedererkannt und wie bewerten Sie diese? (Beispiele für Nachteile: komplexere Infrastruktur nötig, höhere Netzlast, weniger Wiederverwendbarkeit von Quellcode, komplexes Testing)

FAE

- „Vertrauen“ auf die Korrektheit anderer Services
- Dieses Projekt könnte in diesem Umfang ohne Probleme von einem fünfköpfigen Team in absehbarer Zeit entwickelt werden. Durch die große Anzahl an Teilnehmern und die Teilung in Microservices ist die Architektur deutlich komplizierter als notwendig geworden.
- komplexere Infrastruktur, komplexes Testing, Verschleierung von Informationen sehr aufwendig
- Sehr viele Schnittstellen bringen viele Problemherde und, Fehler bei einem zentralen Service können das ganze Projekt zum Scheitern bringen.
- Komplexe Kommunikation zwischen den Services, es wurde sich nicht immer an definierte Naming Conventions gehalten (siehe Resources Naming), einige Teams haben dazu geneigt sich in ihrer eigenen Blackbox zu verkriechen (Game kam ziemlich spät, Gamelog suuuuper spät, Map war schlecht zu erreichen (wegen Orga Struktur klar, war trotzdem bisschen doof))
- Aus meiner Sicht hat die technische Seite recht gut geklappt. Die Absprachen waren eher das Problem.

IP

- Sehr komplexes hin und her von Events
- Das Deployment ist kompliziert und kann durch Kubernetes noch komplexer gestaltet werden.
- Komplexere Infrastruktur, ganz klar. Komplexere Testmethoden auch. Insgesamt braucht es deutlich mehr Koordinationsaufwand, um Microservices richtigzumachen.
- Testing lokal teilweise gar nicht möglich, man weiß halt nie wirklich, ob der Service, mit dem man kommunizieren möchte auch später genau so funktioniert, komplexes Umsetzen mit async. und sync. gleichzeitig
- Erheblich höherer Overhead in allen Belangen. Vorteil zeichnet sich eher bei wirklich großen Projekten ab, denn viele Probleme sind durch die Architektur an sich entstanden und das managen von states. Das ist aber ein generelles Problem der Spiel-Idee. Ein Runden-basiertes Spiel hat einen solid state und verlangt consistency, eine Event-Driven Microservice Architecture ist aber eventually consistent. Das ist aber

ein Mismatch und auch ein ganz erheblicher Grund dafür, warum hier viele Probleme entstanden sind. Rundenbasiert wäre meiner Meinung nach ein Modular Monolith passender. Für einen Microservice wäre ein Runden-loses-Spiel erheblich besser.

- Ungleichheiten bei apis
- Deutlich größeres Overhead. Es muss für jeden Services viel Boilerplait code wiederholt werden. Außerdem müssen ähnliche Konzepte gleich mehrfach implementiert werden, bspw. Event handling usw.
- Komplexität der Zusammenarbeit unter den Services und Endpunktänderungen, Langsamkeit durch die Containerisierung (bin ich nicht gewohnt aus Job, nimmt ein bisschen den Spaß am Programmieren)

MSA

- komplexes Testing
- Anscheinend hohe Last beim Betreiben aller Services auf demselben Host, Refactorings öffentlicher Schnittstellen können nicht automatisiert vorgenommen werden, wie bei einer gemeinsamen Codebase, sondern führen zu breaking API changes.

PL

- Komplexe Kommunikation zwischen den Services. Das Testing ist komplex, aber mit guten Mocks gut umsetzbar. Bei den Mocks müssen wir nur darauf achten, dass die OpenAPI Spezifikationen aktuell sind.

E.3.9 Welche Vorteile von Microservices haben Sie in diesem Projekt wiedererkannt und wie bewerten Sie diese? (Beispiele für Vorteile: leicht ersetzbar, leicht skalierbar, technische Unabhängigkeit, unabhängige Entwicklung einzelner Teams)

FAE

- Autarke Entwicklung, wenn Schnittstellen stabil sind
- Teams können etwas unabhängiger voneinander arbeiten.
- technische Unabhängigkeit, unabhängige Entwicklung einzelner Teams
- Die Gruppen können sich auf ihr Gebiet spezialisieren, was zu besseren Ergebnissen in den Details führen kann.
- eigener Service konnte gebaut und finalisiert werden, ohne dass man ewig auf andere warten musste (überwiegend, Schnittstellen wie z.B. Commands von Game entgegennehmen, waren natürlich trotzdem so ein Thema)

- Wir waren technisch sehr unabhängig und konnten unseren Stack selber wählen, das fand ich gut.

IP

- Leichtes Updaten einzelner Komponenten. Entwicklung der Services kann mit viel Parallelität laufen.
- Klarere Aufgabenverteilung durch die Services in der Software. Einfachere Wartbarkeit im Gegensatz zum Monolithen.
- Ohne den Lernerfolg infrage zu stellen: eigentlich habe ich keinen Vorteil in dem Projekt wiedererkannt. Einfach aus dem Grund, dass alle Services sich aufeinander verlassen haben. Wenn wir während einem Spiel einen Service (außer gamelog) abschießen würden, würden alle anderen Services einfach Fehler schmeißen und es nicht nach Neustart nochmal versuchen. Dafür fehlt uns einfach noch sehr viel Management Code. Unabhängige Entwicklung gab es deswegen auch nicht wirklich. Die technische Unabhängigkeit hat man in einzelnen Zügen sehen können, allerdings beschränkt dadurch, dass das DevOps-Team nur aus 3 Leuten bestand. Wenn jedes Team sein eigenes Build/Deployment selbst macht, haben wir tatsächliche Unabhängigkeit. Skalierung hatten wir gar nicht, gab nur einen Container pro Service, aber in der Theorie möglich.
- verschiedene Technologien, die miteinander kommunizieren und somit auch Unabhängigkeit der Teams
- Dezentrales Arbeiten in Teams, Information Hiding. Die Domain-Berührung mit anderen Teams ist sehr gering.
- Die Services funktionieren unabhängig voneinander und müssen entsprechend nur teilweise neu gestartet werden, im Fehlerfall
- Besonders die Modularität und der Schnitt von Services und die Aufteilung in die jeweiligen Teams fand ich gut. Außerdem hätte man ja theoretisch für jeden Services eine andere passendere Technologie verwenden können. Also bspw. andere Datenbanktypen, Suchmaschinen oder Programmiersprachen. Das gibt einem enorme Flexibilität
- unabhängige Entwicklung in den Teams bis zu einem gewissen Grad. An den wichtigen Stellen war die Kommunikation "mau", und es war teilweise schwer, verbindliche Zusagen zu bekommen. Skalierbarkeit fand ich hier nicht wichtig, da ich keine hohe Auslastung sehe. Die technische Unabhängigkeit ist gut.

MSA

- leicht ersetzbar, leicht skalierbar, technische Unabhängigkeit, unabhängige Entwicklung einzelner Teams
- Hohe Code Ownership der einzelnen Teams und daher besseres "Domänenwissen". Leichte Erweiterbarkeit

PL

- Entwicklung einzelner Teams. Vielleicht macht es bei anderen Projekten mehr Sinn, mit einem Monolith zu starten, um dann einzelne kritische Services zu extrahieren.

E.4 Fragen zum Gelernten

E.4.1 In welchem Bereich haben Sie am meisten im Projekt gelernt?

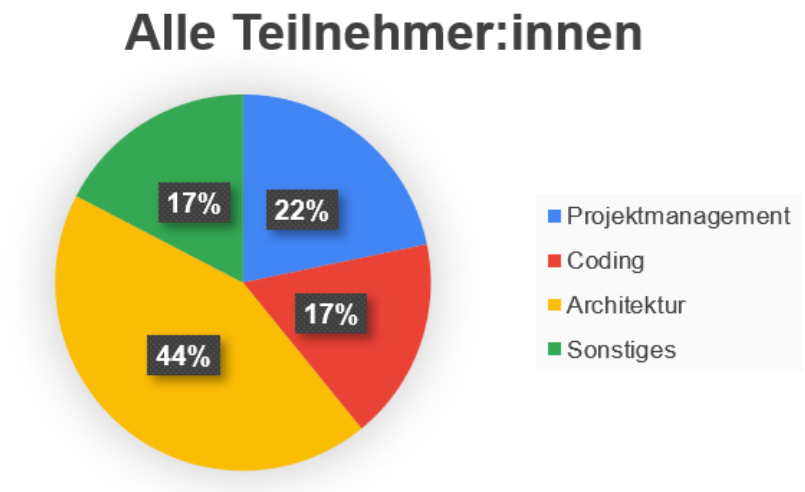


ABBILDUNG E.12: Größter Lernerfolg (Alle)

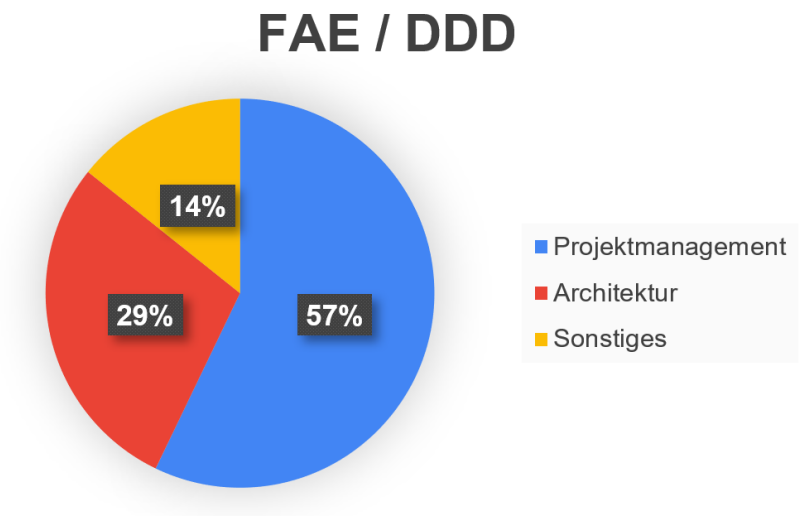


ABBILDUNG E.13: Größter Lernerfolg (FAE/DDD)

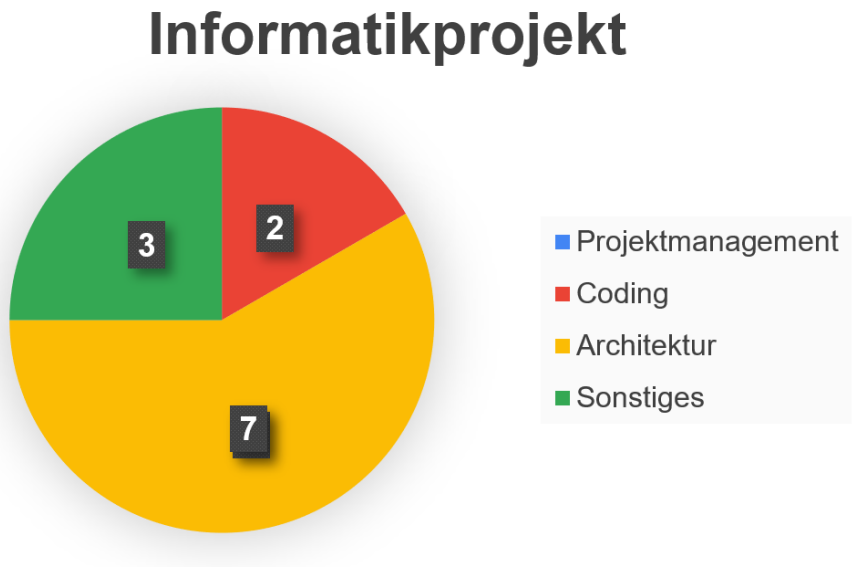


ABBILDUNG E.14: Größter Lernerfolg (Informatikprojekt)

E.4.2 Wie haben sich Ihre Coding Skills durch das Projekt verändert?

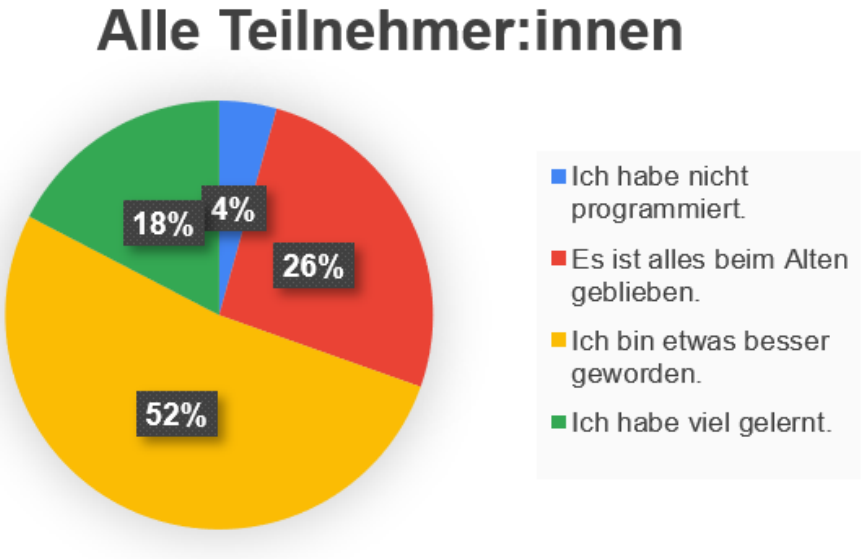


ABBILDUNG E.15: Entwicklung Codingskills (Alle)

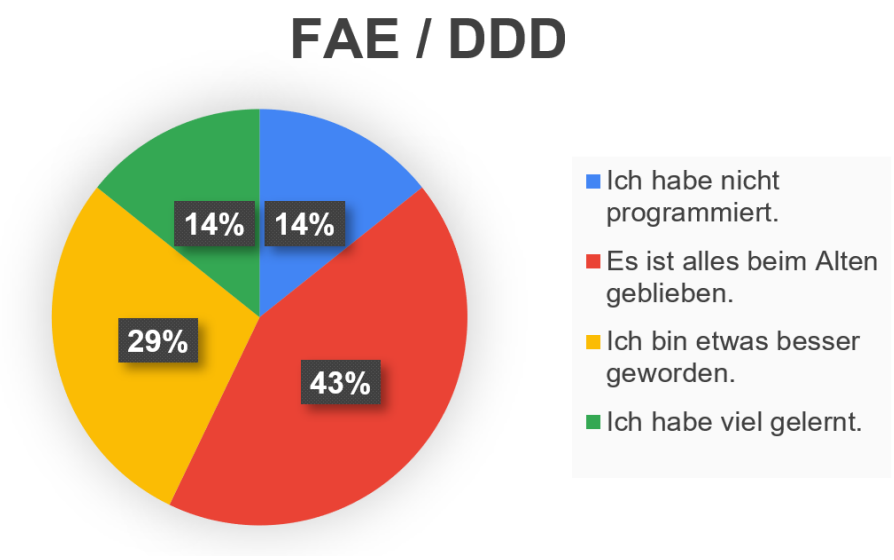


ABBILDUNG E.16: Entwicklung Codingskills (FAE / DDD)

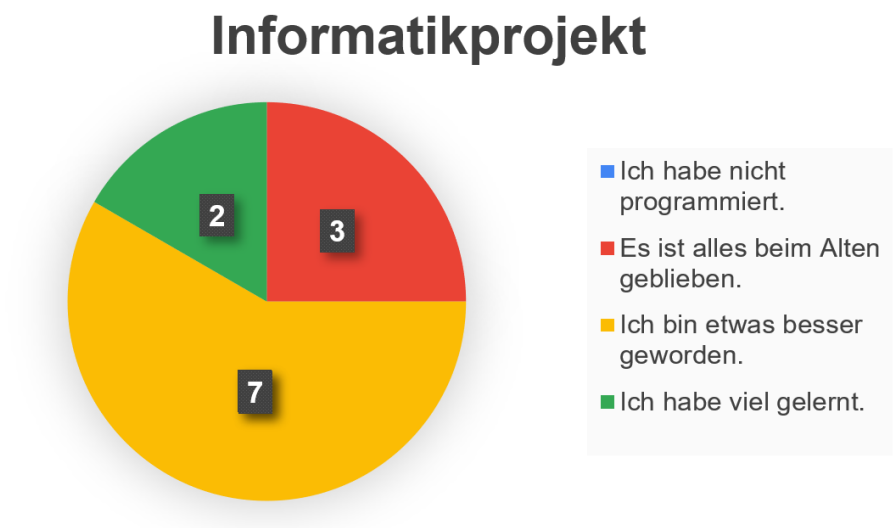


ABBILDUNG E.17: Entwicklung Codingskills (Informatikprojekt)

E.4.3 Hat Ihnen das Projekt geholfen, die Thematik der Microservice näherzubringen?

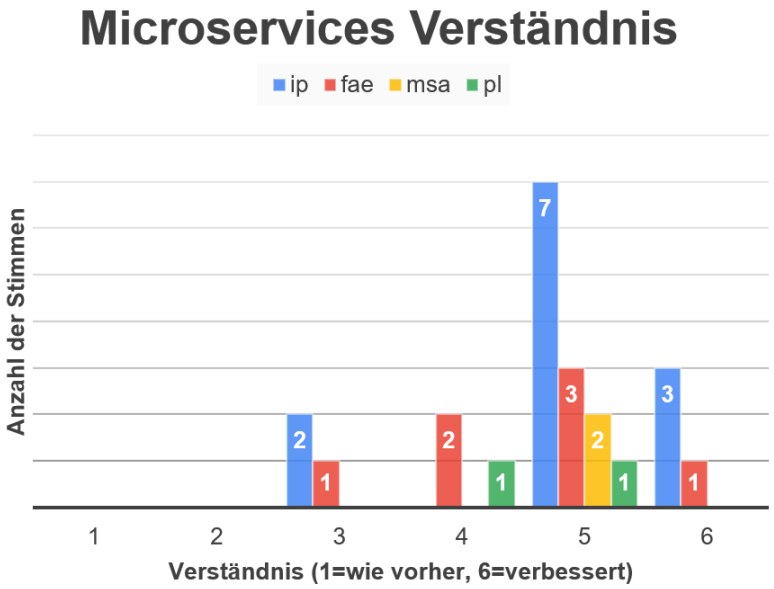


ABBILDUNG E.18: Projekterfolg bzgl. des Kennenlernens von Microservices

E.4.4 Haben Sie die Vorteile/Bedeutung von Microservices im Projekt nachempfinden können?

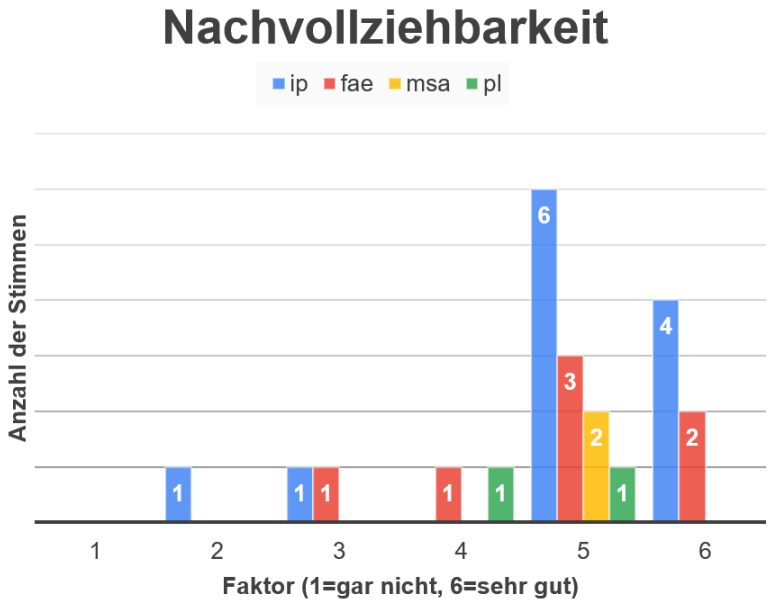


ABBILDUNG E.19: Nachvollziehbarkeit der Bedeutung von Microservices

E.5 Direktes Feedback

E.5.1 Was ist die größte Spaßbremse des Projekts für Sie gewesen?

FAE

- Viele Hacks und Workarounds nötig, Fehlentscheidungen im Tech-Stack -> Unzufriedenheit mit dem eigenen Service
- Ständige Verfügbarkeit in Discord
- Ungleiche Motivation
- Ausfälle bei einzelnen Mitgliedern wegen zu wenig Zeit oder Krankheit, sodass die Arbeit mehr auf den anderen lag.
- Leute, die zu spät mit Forderungen ankamen, weil sie sich vorher nicht drum gekümmert haben (aus Robot Perspektive waren das Game und Gamelog…)
- (Externer) Zeitdruck, Probleme bei der Kommunikation, fehlende Testdaten

IP

- Von der Spec abweichende Implementierung einiger Servicepunkte
- Kommunikation über Discord. Etwas zu offene Organisation.
- Widersprüchliche Spezifikationen. Gameplay-Entscheidungen zum Leiden der Implementation des Spielers.
- Fehlendes Engagement von Projekt-Mitgliedern, und mehrfaches Fehlschlagen von Fixes und Tests
- Unfähiges Team.
- Kafka-Probleme
- Kafka. Ich finde, man hätte im Vorfeld mehr darauf eingehen sollen.
- Das fehlende Gesamtbild, welches erst spät entstand.

MSA

- Manche Teamkollegen
- Nicht konsistente API (Doku)

PL

- Die Länge des Projekts hat sich leider über die eingeplante Zeit des PL gezogen.

E.5.2 Was hat am meisten Spaß gemacht?

FAE

- Die fachlichen Diskussionen
- Event Storming
- Spieler-Entwicklung
- Die Arbeit im Team, Event-Storming
- Meiste Zeit gute Stimmung im Team und das Erfolgserlebnis nach dem holprigen Ende
- Zu sehen wie gut Team Robot zusammen Gas gegeben hat und einfach delivered hat :D
- Coding mit TDD und Clean Code

IP

- Eindenken in viele verschiedene, teils neue Aspekte im Bereich Software-Entwicklung in einem „größeren“ Team bzw., mit mehreren Teams
- Wenn die Services angefangen haben miteinander zu interagieren.
- Das Durchlaufen der Build Pipeline zu sehen. Bzw., dass der Code vollständig automatisiert auf dem Server ausgeführt wird.
- Coding an sich und ausprobieren von Funktionalitäten (die dann auch nicht tausendmal fehlschlagen)
- Teamarbeit
- Allgemeiner und privater Austausch mit engagierten Teilnehmern, neue Leute kennengelernt, viel gelernt, besonders durch Fehler.
- Entwicklung des Players
- Konzeption und Diskussion mit like-minded-People
- Die Struktur des eigenen Players.

MSA

- Der Kreativität freien Lauf lassen
- Systemdesign & Frontend Client "basteln"

PL

- Der letzte Codefight und Show Off!

E.5.3 Was ist die größte Erkenntnis, welche Sie im Projekt erlangt haben?

FAE

- Kommunikation ist wichtig
- Funktionsweise von Events
- Wie hilfreich Logging in dieser Architektur sein kann.
- Eine Gruppe ist definitiv nur so stark, wie ihr schwächstes Glied.
- Ein solches Projekt braucht organisatorisch ein gutes Grundgerüst mit verbindlichen Kommunikationskanälen, Dokumentationsstandards (v.a. definierter Ort) und gemeinsamer Planung in regelmäßigen Treffen.

IP

- Specs müssen(!) bei solchen Projekten unbedingt eingehalten werden, sonst wird es gegen Ende stressig und unübersichtlich.
- Ohne DevOps läuft nichts.
- Große Projekte sind immer komplizierter, als man denkt.
- Eine gute Dokumentation der API ist essenziell. Ohne die Doku wäre z.B. eine Weiterentwicklung sehr schwierig bis unmöglich geworden. Gleiches gilt für den Player.
- Unterschätzung des Projektumfangs
- Dass die größte Hürde eines Projektes Personalmanagement ist, nicht Technik. Kompetente und engagierte Mitarbeiter, die sich untereinander vertrauen können, ehrlich und offen miteinander kommunizieren, machen 90 % aus.
- Ich habe mich mehr mit Architekturen auseinandergesetzt

MSA

- Spring Annotations sind super geil
- Verständnis für Pathfinding Algorithmen

PL

- Komplexe Systeme brauchen viel Zeit!

E.5.4 Was sind Ihre Vorschläge für ein Microservice Dungeon 2.0 Projekt?

FAE

- Komplexität entzerren, Babyschritte machen. Events unbedingt überarbeiten - was sind Events? Veränderung an Entities oder Aktionen? Testing unbedingt überarbeiten, eine Testumgebung

- Mehr Struktur vorgeben
- Für die Player Entwicklung eins der folgenden umsetzen:
  - deutliche Reduktion der Spielkomplexität
  - mehr Zeit für die Entwicklung
  - sehr ausgereifte Generic Player
- Das gleiche Projekt, aber schon mit dieser bestehenden Grundlage. Oder mit weniger Services, dafür mehr pro Team.
- Gameplay Team immer mit dabei haben und nicht nur am Anfang
- - Workshops im Vorhinein, um essentielle Dinge zu vermitteln:
  - Orga: gemeinsame Standards für Issues(&-pflege), Doku, Decisions, Kommunikation zwischen Teams; Regelmäßigkeit, Ablauf, Vor- und Nachbereitung der Team-Meetings; agiles Vorgehen im Team und im Gesamtprojekt
  - Techniken: TDD, Clean Code, DevOps, agiles Projektmanagement mit dem gewählten Tool

IP

- Api und Event Specs fest ausformulieren bevor entwickelt wird
- Straffere Organisation. Klare Regeln der Kommunikation. Mehr Präsenztermine, wo alle mal zusammen coden.
- Verbesserungen:
  Lehrmaterial zu: DevOps (speziell Build/Deploy), Testing, Asynchrone Entwicklung. DevOps sollten in jedem Team praktiziert werden. Weniger "kritischeSServices, bzw. mehr unabhängige, mit einer Aufgabe, die sie zur Not auch ohne (oder mit ein bis zwei weiteren) Services erledigen können. Gemeinsam Eventing-Guidelines erstellen (bspw.: keine Events mit variabler Payload Struktur im selben Event-Typ, immer transactionIds usw.). Weniger Umfang für das individuelle Abschluss-Produkt (in unserem Fall den Spieler-Service).

  Sonstige Ideen:
  Container-Management Software, SSL-Verschlüsselung, Secret-Vaults für Production-Server / Github, einheitliches frontend (queried alle servies oder so).
- Den Scope verkleinern und konkrete Deadlines setzen für die Services, damit nicht erst zwei Wochen vor Ende tatsächliche Fehler hereinkommen.
- Vor Ort!
- Lehr-Videos für die Veranstaltung bereitstellen, keiner hatte Ahnung, was ein Microservice ist, sollte aber einen bauen. Kleinere Teams, maximal 2-3 Personen. Bei 4 fremden Leuten ist der Overhead so groß, dass ein 2er-Team schneller wäre. FAE-Leuten mehr auf die Finger schauen

und notfalls eingreifen, wo das Design nicht funktioniert, denn wenige können das gesamte Projekt runterziehen.

- Realitätsnäheres Projekt, mehr Material zu Kafka, bessere lokale Entwicklungsumgebung,
- Mehr Kontrolle der Services und eine Terminplanung welche nicht so nah an die Klausuren kommt.

MSA

- Keins - Mag Spiele nicht so. Lieber echten Product Owner finden und ihm was programmieren
- AI / ML, mehr Visualisierung, Full Stack Services

PL

- Eine Weiterführung mit den aktuellen Services. Hier könnte man wirklich "Frameworks"für Player coden, mit denen man dann schnell eigene Strategien bauen kann.

E.5.5 Gib es weiteres Feedback, Ideen, Kritik, Lob oder Anregungen, welche Sie den Dozenten zum Abschluss mitgeben möchten?

FAE

- Super Projektidee
- Sehr schönes Projekt. Hat viel Spaß gemacht
- Zu Beginn wussten viele von FAE nicht genau, was sie tun mussten (zu wenig Input für Arbeit als Projectowner), aber insgesamt gut strukturiert und für die Services passende Vermittlung der inhaltlich relevanten Informationen.
- Wochenenden respektieren und bisschen weniger Spontanität fordern (spontane Meetings mind. 3 Tage vorher ankündigen, vor allem, wenn diese vorbereitet werden müssen). Die ständige Erreichbarkeit über Discord (da ich das viel privat nutze), war teilweise eine große Belastung
- Vielleicht das Projekt größer und interdisziplinärer denken:
  - als Grundgerüst für das ganze Semester nehmen
  - Module als Blockkurse um das Projekt herum anordnen
  - DevOps als Block vor dem Projekt
  - agiles Projektmanagement auch davor
  - Domain-Driven-Design und Qualitätsmanagement begleitend

  oder so ähnlich.
  Das Projekt hat echt viel Potenzial, praktische Erfahrung schon fast auf Berufs-Niveau zu bieten, wenn es nicht so sehr in das normale Semester-Gerüst gepresst werden muss. Erfordert aber auch Rücksicht

darauf, dass die Studenten neben dem Studium im Master faktisch Arbeiten gehen müssen, um ihren Lebensunterhalt zu sichern - Unterhaltspflicht gibt es nach dem ersten Abschluss nicht mehr und BAföG bekommt quasi niemand.
Geht also wahrscheinlich zeitlich nicht innerhalb der Vorlesungszeit, sondern braucht auch einen Teil der Semesterferien.
Das alles nur als Anregungen - das ist so wahrscheinlich nicht umsetzbar, aber vielleicht gibt es ja einen Mittelweg.

IP

- Ich fand das Ganze ein super Projekt! Interessant, hat Spaß gemacht, gerne wieder!
- Einzigartiges Projekt an der TH. Vielen Dank. Noch nie so ein Engagement von einem Dozenten erlebt.
- Mir hat das Projekt insgesamt super viel Spaß gemacht. Mir hat es eigentlich nie wirklich an Motivation gemangelt. Ich hab sehr viel gelernt, auch wenn es nicht 100% Microservices waren. Außerdem fand ich es super, dass sie ihren eigenen Player entwickelt haben und so auf dieselben Probleme gestoßen sind wie wir, generell die Kommunikation auf Augenhöhe. Insgesamt ein tolles Projekt, auch wenn der Zeitumfang extrem groß war. Hoffe auf eine Weiterführung, am liebsten als Praxisprojekt ;)
- Von den Lehrenden und Organisatoren kann ich nur großes Lob aussprechen, dass sich an ein solch großes Projekt getraut wurde. Die Mitarbeit der unterschiedlichen Studiengänge ist cool gewesen und man hat gemerkt, dass das Thema die genannten Personen interessiert und sie sich viel Mühe gegeben haben.
- Die Idee ist gut und die Verantwortlichen haben sich Mühe gegeben.

MSA

- Klasse gemacht
- War ein sehr cooles Projekt und für viele der Kommilitonen das erste Mal, än was Richtigemärbeiten.

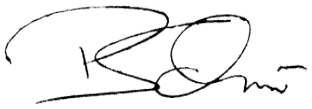
PL

- Sehr schönes Projekt, besonders mit der technischen Freiheit, die den Teams gegeben wird. Ich würde es mir wünschen, dass sich andere hiervon inspirieren lassen, denn ich denke, es gab bis jetzt kein besseres Projekt, um "richtig"Programmieren zu lehren.

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach den 15. Februar 2022



Unterzeichnet: Philipp Felix SCHMEIER