Universität Stuttgart

Institut für Software Engineering
Empirical Software Engineering

Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Toolunterstütztes Refactoring von Microservices-Architekturen: Eine industrielle Fallstudie

Axel Herrmann

Studiengang:	Software Engineering
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	Jonas Fritzsch, M.Sc. Frank Intorp (levigo solutions)
Beginn am:	25. September 2023
Beendet am:	25. März 2024

Kurzfassung

Aufgrund zahlreicher Vorteile gegenüber monolithischen Anwendungen hat sich das Architekturprinzip von Microservices im letzten Jahrzehnt immer weiter verbreitet. Faktoren wie die bessere Skalierbarkeit oder größere Flexibilität bei der Kombination von Technologien können es für viele Produkte sinnvoll machen, von einer monolithischen zu einer Microservices-basierten Architektur zu migrieren. Es ist jedoch ein sehr zeit-, kostenintensiver und risikobehafteter Prozess, die grundlegende Architektur eines Produkts zu ändern. Eine klare Vorgehensweise für den Migrationsprozess zu finden, ist eine schwierige Aufgabe. Da viele Entwickler dabei nicht auf wissenschaftliche Literatur zurückgreifen, wurde in früheren Arbeiten das Tool Architecture Refactoring Helper (ARH) auf Basis des Frameworks Microservices Migration Framework (MMF) entwickelt, das den Migrationsprozess mit wissenschaftlichen Erkenntnissen unterstützen soll. Im Rahmen dieser Thesis wurde das Tool erstmals für das Refactoring eines industriellen Produkts mit Microservices-Architektur eingesetzt. Zunächst wurde mithilfe des Frameworks ein Architekturreview durchgeführt, um die gewünschten Qualitätsattribute für das Produkt *jadice flow* szenarienbasiert zu erfassen. Diese und weitere Filtereinstellungen dienten als Basis für die Suche mit dem ARH nach Migrationsverfahren. Die resultierenden Suchergebnisse wurden manuell evaluiert. Eine Anwendung der beiden bevorzugten Migrationsverfahren wurde versucht, konnte jedoch nicht fertiggestellt werden. Zur Evaluation der Studie wurden deshalb die beiden Verfahren sowie die Suchergebnisse der dritten Phase in Form von Best Practices und Patterns in Experteninterviews bewertet. Aus der Auswertung dieser und der während des Refactoring-Prozesses gesammelten Feldnotizen ergibt sich eine positive Evaluation des MMF. Die Experten hoben dabei die einzelnen Funktionen sowie das Vorgehen in Phasen positiv hervor. Außerdem wurde vermutet, dass die ausgewählten Migrationsverfahren bei zukünftiger, vollständiger Anwendung potentiell nützlich für *jadice flow* sein könnten. Daraus wird geschlussfolgert, dass die Anwendung des Frameworks in dieser Fallstudie erfolgreich war und das MMF auch für Refactorings von Produkten mit vorhandenen Microservices-Architekturen nützlich sein kann. Es gilt, diese in der Zukunft einzusetzen, sowie weitere Untersuchungen zum und Verbesserungen am ARH durchzuführen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface. 48

ARH Architecture Refactoring Helper. 3, 9, 17

ATAM Architecture Trade-off Analysis Method. 7, 24

BO Business Object. 49

BP Business Process. 21, 22, 47

BPMN Business Process Model and Notation. 47

CD Continuous Delivery. 16

CI Continuous Integration. 16

ESE Empirisches Software Engineering. 13

HAC Hierarchical Agglomerative Algorithm. 47

IIoT Industrial Internet of Things. 49

ISTE Institute of Software Engineering. 13

KI Künstliche Intelligenz. 76

LOC Lines of Code. 50

M3K Metaprocess for Microservice Migration Kernel. 21

MB Microservice Backlog. 48

MMF Microservices Migration Framework. 3, 7, 13

MSA Microservices-Architektur. 3, 13

PO Product Owner. 25

QA Qualitätsattribut. 3, 7, 17

REST Restful State Transfer. 15

SA Softwarearchitekt. 29

SAAM Software Architecture Analysis Method. 24

SE Softwareentwickler. 29

SIA Service-Identifikationsansatz. 18

SP System Property. 43

SQL Structured Query Language. 49

UX User Experience. 68

1. Einleitung

In einer Zeit, in der viele Softwarelösungen auf Cloud Computing basieren, ist auch die Microservices-Architektur immer gebräuchlicher geworden. Viele Produkte wurden innerhalb des letzten Jahrzehnts schon in diese Architektur übergeführt. Das Hauptziel ist, von Vorteilen wie besserer Skalierbarkeit, Wartbarkeit und Flexibilität zu profitieren [FBWZ19; TLP17]. Abhängig von dem Ausmaß und der Komplexität monolithischer Systeme kann die Migration in eine Microservices-Architektur (MSA) jedoch sehr zeitaufwendig, kostenintensiv und kompliziert sein. Es liegt aktuell eine Vielzahl akademischer Forschungsarbeiten zum Thema Microservices vor, deren Erkenntnisse jedoch nur selten in der Industrie eingesetzt werden [FBH+22].

Um diese Migration zu vereinfachen und die wissenschaftliche Expertise der Industrie leichter zugänglich zu machen, hat die Abteilung Empirisches Software Engineering (ESE) des Institute of Software Engineering (ISTE) das Microservices Migration Framework (MMF) entwickelt. Mit der Entwicklung des MMF, das Erkenntnisse über die Migration von monolithischen Anwendungen hin zu Microservices in die Industrie bringen soll, hat das ESE erste Schritte unternommen, um die erwähnte Lücke zwischen Industrie und Forschung zu schließen.

Bei der Anwendung des auf dem Framework basierenden Tools in der Industrie besteht jedoch noch Verbesserungsbedarf. Praktische Anwendungen sind für die Weiterentwicklung und Verbesserung des Frameworks und des Tools wichtig. In der Arbeit von Knodel [Kno23] wurde eine erste industrielle Fallstudie zu dem Framework und dem Tool mit der (Teil-) Migration eines Monolithen durchgeführt. Diese Bachelorarbeit bildet die Fortsetzung der Fallstudien zum MMF. Das Framework wird erstmalig für das Refactoring der bereits vorhandenen MSA des realen Produkts *jadice flow* verwendet. Die Ergebnisse dieses Refactorings werden zur Bewertung der Effektivität des Frameworks und des Tools verwendet, wodurch diese in Zukunft weiter verbessert werden sollen.

Bei der Analyse von *jadice flow* mithilfe des Frameworks werden primär drei Designaspekte von Microservices betrachtet:

[1] Die optimale Service-Granularität. Bei dieser muss zwischen Overhead durch zu viele Einheiten und Overhead durch zu große Einheiten abgewogen werden.

[2] Paradigma der Ansteuerung der einzelnen Einheiten, Kommunikationsmodell zwischen den Einheiten.

[3] Reduktion des IO-Flaschenhalses, Art des Payload-Transfers. Der IO-Flaschenhals entsteht dadurch, dass die einzelnen Einheiten, bevor sie ihre Arbeit verrichten können, immer die Eingangsdokumente herunterladen und anschließend die Ergebnisse wieder hochladen müssen.

Daraus folgt die Forschungsfrage, die in dieser Thesis untersucht wird:

FF: Wie kann eine bereits bestehende Microservices-Architektur mit Hilfe des Microservices Migration Framework (MMF) hinsichtlich konkreter Qualitätsaspekte weiter optimiert werden?

1.1 Welche Refactoring-Verfahren eignen sich zur Bestimmung der optimalen Service-Granularität einer bestehenden Microservices-Architektur?

1.2 Welche Ansätze, Patterns oder Best Practices eignen sich zur Optimierung des Kommunikationsmodells und der Verringerung des IO-Flaschenhalses zwischen den einzelnen Services?

Um die Forschungsfragen zu beantworten, ist diese Thesis in die folgenden Kapitel aufgeteilt:

Kapitel 2 führt in den theoretischen Hintergrund ein, der für diese Arbeit relevant ist. Dabei werden auch das MMF und *jadice flow* kurz beschrieben und ähnliche Arbeiten vorgestellt. **Kapitel 3** beschreibt die Methodik, mit der diese Thesis durchgeführt wird und die Forschungsfrage beantwortet wird. Unter Verwendung des MMF wird in **Kapitel 4** am Produkt *jadice flow* die Planung eines Refactorings durchgeführt. Dabei werden die drei Phasen des Frameworks ausgeführt, welche einen Großteil der Arbeit dieser Thesis ausmachen. Eine Bewertung der Ergebnisse dieser Anwendung wird in **Kapitel 5** beschrieben. Abschließend werden in **Kapitel 6** Einschränkungen und die Gültigkeit dieser Arbeit diskutiert und in **Kapitel 7** ein Fazit gezogen sowie ein Ausblick gegeben.

2. Theoretischer Hintergrund

Zum Architekturprinzip der Microservices gibt es bereits eine Vielzahl akademischer Publikationen. In diesem Kapitel wird in den theoretischen Hintergrund der für diese Thesis relevanten Literatur eingeführt. Zunächst wird ein kurzer Vergleich zwischen dem häufig verwendeten Legacy-Architekturprinzip *Monolith* und dem behandelten, modernen Architekturprinzip *Microservices* gezogen. Anschließend werden aktuelle Arbeiten zur Migration zu Microservices-Architekturen beschrieben. Des Weiteren wird das Microservices Migration Framework (MMF) vorgestellt, das ein wesentlicher Bestandteil und Untersuchungsobjekt dieser Arbeit ist. Abschließend wird ein Überblick über das Produkt gegeben, das in dieser Thesis mithilfe des MMF überarbeitet wurde.

2.1. Vergleich von monolithischen und Microservices-Systemen

Die monolithische Systemarchitektur hat lange Zeit einen Großteil der Softwareprodukte ausgemacht und ist die konventionelle Art, Software zu entwickeln. Moderne Programmiersprachen ermöglichen zwar die Modularisierung von Monolithen, jedoch befindet sich die gesamte Funktionalität eines Monolithen in einer Applikation, die aus nur einem ausführbaren Artefakt besteht. Nach den meisten Definitionen (siehe [DGL+17]) sind Monolithen dadurch gekennzeichnet, dass ihre einzelnen Module nicht unabhängig voneinander ausführbar sind. Auch wenn in dieser Arbeit die Migration weg von dieser Architektur thematisiert wird, hat sie ihre Vorteile. Da es sich um eine simple Architektur mit nur einer Codebasis handelt, ist das Erstellen, Testen, Deployen und auch Monitoring kleiner Anwendungen einfacher und schneller [VF23]. Es ist kein kompliziertes Kommunikationsmodell notwendig, da Monolithen oft in einer Programmiersprache und sogar mit nur einem Framework geschrieben werden.

Je größer Monolithen jedoch werden, desto stärker zeigen sich die Schwächen dieser Architektur. Es entstehen riesige Codebasen und kleine Änderungen erzwingen erneutes Bauen und Testen der gesamten Applikation [DGL+17]. Dies steht auch im Widerspruch zu modernen agilen Arbeitsprinzipien. Wenn nur einzelne Komponenten eines Monolithen eine hohe Last erfahren, muss trotzdem das gesamte System skaliert werden, was zu einer Verschwendung von Rechenleistung führen kann und die Anwendung ineffizient macht [DGL+17]. Außerdem kann schnell das Problem der „Dependency Hell“ [DGL+17] auftreten, wobei das Hinzufügen oder Ändern von Abhängigkeiten dazu führen kann, dass das gesamte Projekt nicht mehr kompiliert oder ausgeführt werden kann.

Die meisten dieser Probleme werden durch die Microservices-Architektur gelöst. Dabei besteht ein System aus vielen kleinen (Micro-) Services. Diese zeichnen sich dadurch aus, dass sie möglichst klein gehalten werden, genau eine Aufgabe haben und unabhängig ausgeführt werden können [VF23]. Durch die Entkopplung der Services wird im Gegensatz zu Monolithen ein Kommunikationsmodell zwischen den Services notwendig. Dafür existieren verschiedene Lösungen, die im Rahmen dieser Arbeit näher betrachtet werden. Eine sehr häufig eingesetzte Lösung ist beispielsweise Restful State Transfer (REST).

Die Vorteile dieser Architektur gegenüber Monolithen sind vielseitig. Da die Services unabhängig voneinander konzipiert sind, können sie einzeln implementiert, getestet und gebaut werden. Dies erleichtert die Automatisierung des Bauens und Testens durch Continuous Integration (CI)/Continuous Delivery (CD). Es ermöglicht auch, dass die Anwendung während eines Updates auf eine neue Version durchgehend online bleibt [VF23]. Des Weiteren können Services abhängig von der Last einzeln skaliert werden, was die Effizienz in bestimmten Szenarien deutlich erhöht. Darüber hinaus sind Entwickler flexibel in der Wahl der Programmiersprache und des Frameworks für einzelne Services, da die einzige Anforderung die Umsetzung des gewählten Kommunikationsmodells ist.

Eine Microservices-Architektur ist allerdings nicht universell einer monolithischen Architektur vorzuziehen. Gründe dafür sind einige Nachteile von Microservices-Architekturen, die von Velepucha und Flores [VF23] zusammengetragen wurden. Dazu gehört vor allem die Komplexität des Systems, die bei einer Microservices-Architektur deutlich höher ist. Dies liegt an der Aufteilung des Systems in einzelne Services und der nichttrivialen Kommunikation zwischen den Services. Viele Entwickler müssen die neue Architektur mit ihrer erhöhten Komplexität, sowie die zugehörigen Tools und Mechanismen für automatisierte Tests, Deployments und Monitoring erst erlernen. Der resultierende Netzwerk-Overhead durch die Kommunikation zwischen den Komponenten kann zu höheren Latenzen führen und das System ineffizient machen, wohingegen bei Monolithen lediglich Funktionen innerhalb desselben laufenden Programms aufgerufen werden. Des Weiteren ist es empfehlenswert, dass Microservices jeweils eigene Datenbanken haben. Das fügt weitere Komplexität zum Design hinzu, da Daten dupliziert werden müssen und die Konsistenz der Datenbanken gewährleistet sein sollte. All dies führt dazu, dass Experten empfehlen, für kleine Anwendungen bei monolithischen Architekturen zu bleiben [Sin16; VF23].

2.2. Migration zu Microservices

Aufgrund der beschriebenen Vorteile von Microservices-Architekturen ist es im Interesse vieler Unternehmen und Entwickler, existierende Applikationen und Systeme mit monolithischer Architektur zu Microservices-basierten Systemen zu migrieren. Der Planungsprozess dieser Migration ist allerdings sehr zeit- und kostenintensiv und durch große Risiken geprägt, da dabei Systeme tiefliegend verändert werden. Deswegen ist die Migration zu Microservices seit etwa 2015 ein stark untersuchtes Thema.

Bei der Migration zu Microservices gibt es mehrere zentrale Herausforderungen, die jedes Entwicklerteam bewältigen muss und auch in der Forschungsfrage als Untersuchungsobjekt dieser Thesis genannt werden. Eine der größten ist die Dekomposition des vorhandenen, großen, zusammenhängendem Systems in kleine, unabhängige Services [TLP17; TS19; VF23]. Die sogenannte optimale Granularität der Services zu bestimmen, ist eine sehr wichtige Aufgabe, da sie starke Auswirkungen auf die Performanz des Systems haben kann. Der Nachteil zu großer Services ist, dass sie ineffizient werden und womöglich ähnliche unerwünschte Eigenschaften erhalten wie monolithische Systeme. Dagegen ist auch eine Aufteilung in zu kleine einzelne Einheiten problematisch, da dadurch der Kommunikationsbedarf zwischen den Microservices stark erhöht werden kann, was ebenfalls leistungsbezogene Nachteile hat. Für diesen entscheidenden Prozess wäre es also wichtig für Entwicklerteams, wissenschaftlich gestützte Entscheidungen treffen zu können.

Ebenfalls ist die Konzipierung der Inter-Service-Kommunikation ein wichtiger Bestandteil, der starke leistungsbezogene Auswirkungen haben kann. Das liegt daran, dass im Vergleich zu monolithischen Systemen die Kommunikation zwischen den Services die Latenz für gleichartige Aufgaben erhöht. Um diesen Effekt zu minimieren, sollte möglichst wissenschaftlich gestützt vorgegangen werden und bewährte Best Practices sowie Patterns evaluiert und verwendet werden.

Bei der Migration zu Microservices gibt es neben den zwei genannten viele andere Herausforderungen, wie Mangel an Verständnis, Identifikation der Service-Grenzen, Testing, Fehlertoleranz, Serviceintegration, Datenkonsistenz, hohes Coupling im Legacy-System, organisatorische Herausforderungen, Sicherheit, Datenbank-Migration, Datenmanagement und Monitoring [AAQ+23]. Um diese verschiedenen Herausforderungen nicht einzeln angehen zu müssen, kann ein Framework hilfreich sein, das den gesamten Refactoring-Prozess in mehreren Phasen begleitet. Ein solches wird im nächsten Kapitel vorgestellt und anschließend in dieser Arbeit untersucht.

2.3. Microservices Migration Framework (MMF)

Es wurden bereits viele Gründe für die häufige Migration zu Microservices beschrieben, sowie die damit verbundenen Risiken und Herausforderungen. Der Migrationsprozess ist sehr komplex, riskant und fehleranfällig. Erschwert wird er durch das Fehlen klarer und strukturierter Anleitungen. Mehrere Metastudien haben bereits versucht, eine Übersicht über verschiedene Refactoring- und Migrationsverfahren zu geben, die für Entwickler hilfreich sein können. Jedoch bleibt dabei das Problem bestehen, dass sich der aktuelle Stand der Literatur schnell ändert und Metastudien dadurch teilweise obsolet werden. Dieses Problem könnte durch die Umsetzung der Anleitung in Form eines Tools, das Entwickler bei der Migration unterstützt, gelöst werden. Im Gegensatz zu Metastudien könnte dieses Tool dynamisch an den neuesten Forschungsstand angepasst werden.

Aus diesem Grund hat das ESE das Microservices Migration Framework (MMF) und das zugehörige Tool Architecture Refactoring Helper (ARH) [FHK22] entwickelt, welche in diesem Kapitel vorgestellt werden. Während Fritzsch et al. [FBH+22] den Entwurf des Frameworks für das Tool beschreiben, wurde das Tool im Rahmen von zwei Masterarbeiten entwickelt [Hal22; Koc22] und in einer ersten Fallstudie angewendet [Kno23]. Der ARH ist eine webbasierte Anleitung, die Entwickler durch drei Phasen der Migrationsplanung führt. Eine Übersicht über diese Phasen und die Funktionsweise des Frameworks ist in Abbildung 2.1 zu sehen. Im Folgenden werden diese drei Phasen genauer beschrieben:

- **Phase 1: System Comprehension:** In Phase 1 wird das Verständnis des zu migrierenden Systems angestrebt. Mit den Stakeholdern werden dabei strategische Ziele für Produkt und Unternehmen definiert, sowie Qualitätsattribute (QAs) und gewünschte Szenarien identifiziert. Es sollen sich dadurch mögliche Treiber für die neue Microservices-Architektur herauskristallisieren. Durch das resultierende Verständnis des Systems wird ein Vergleich der alten monolithischen Architektur und einer potentiellen neuen Microservices-Architektur möglich. Architekten sollen dann mithilfe dieser Informationen eine Entscheidung für oder gegen die Migration zu Microservices fällen. Zum Zeitpunkt der Durchführung dieser Arbeit konnte im Tool nur die Eingabe der Szenarien mit den zugehörigen QAs erfolgen. Die restlichen Schritte dieser Phase wurden erst nach Abschluss der Arbeit zum Tool hinzugefügt.

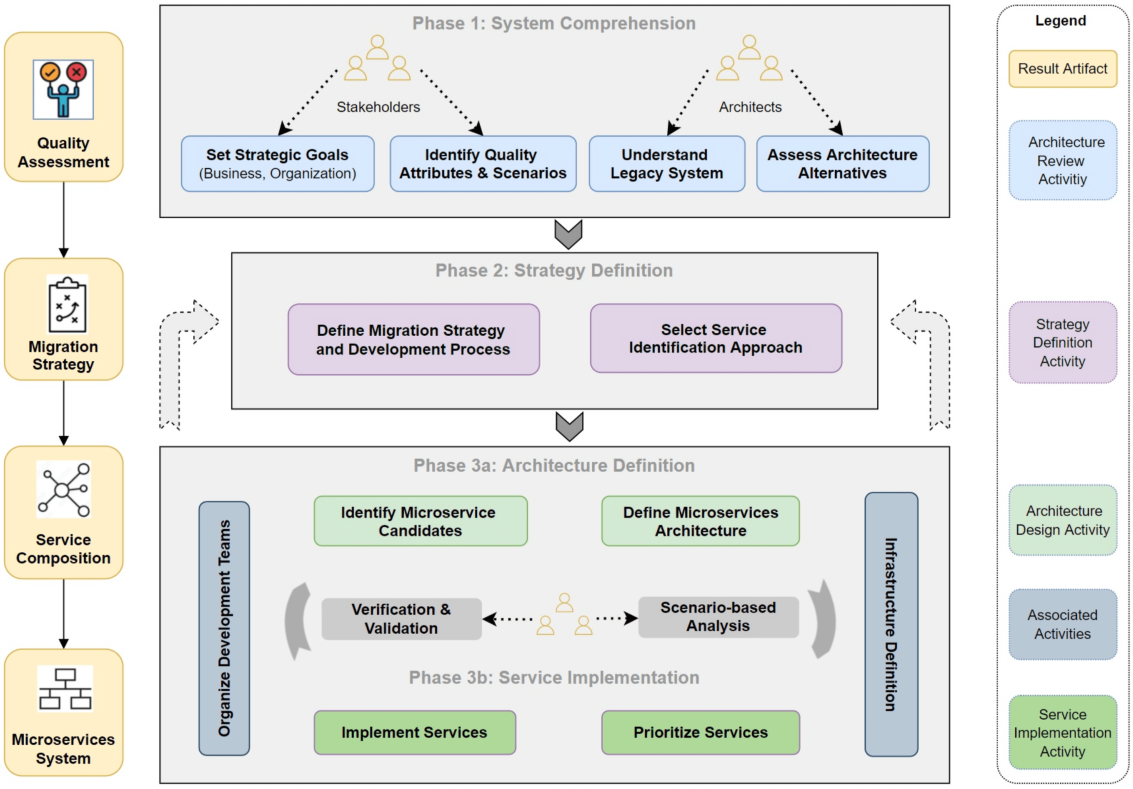

Abbildung 2.1.: Übersicht über das MMF des ESE [FBH+22]. Es sind die drei Phasen des Prozesses zu erkennen, sowie einige Schritte der einzelnen Phasen.

- **Phase 2: Strategy Definition:** Falls sich in Phase 1 für die Migration zu Microservices entschieden wurde, wird in Phase 2 die Planung der Migration begonnen. Die hauptsächliche Aufgabe in dieser Phase ist die Entscheidung, welche Migrationsstrategie benutzt werden soll, um das System zu modernisieren. Dafür kann zwischen (momentan) 115 verschiedenen Methoden gewählt werden, die aus akademischen Publikationen stammen. Das Tool kann Entwickler dabei unterstützen, indem es basierend auf Eingaben aus Phase 1 empfohlene Methoden vorschlägt. Einige dieser Methoden beinhalten die Nutzung von Tools, welche auch gesondert durchsucht werden können. Die verschiedenen Migrationsmethoden und Tools können von Admins importiert, exportiert und bearbeitet werden, wodurch die Erweiterung in zukünftigen Arbeiten vereinfacht möglich ist.
- **Phase 3a: Architecture Definition:** Phase 3 ist in zwei Teile aufgeteilt. Im ersten Abschnitt wird die neue Architektur definiert. Basierend auf der in der Planungsphase ausgewählten Methode zur Migration wird hier der Service-Identifikationsansatz (SIA) durchgeführt. Außerdem wird allgemeiner die Architektur des gesamten Systems geplant, wobei das Tool Entwickler durch eine Liste von vorgeschlagenen Patterns und Best Practices unterstützen kann. Phase 3a und 3b sind eng verbunden, denn durch Erkenntnisse in der Implementierung kann die Planung häufig noch mehrmals überarbeitet werden und dadurch eine neue Iteration der Implementierung begonnen werden.

- **Phase 3b: Service Implementation:** In Phase 3b startet dann ein Zyklus von Implementierungen der in Phase 3a definierten Services. Bei Implementierung selbst kann das Tool Entwickler nicht unterstützen. Doch Bestandteil der Entwicklungszyklen ist auch, dass das entstehende System anhand der Qualitätsmerkmale zu bewerten. Dadurch kann eine unpassende Architekturdefinition oder auch eine unpassende Aufteilung in Services frühzeitig erkannt und korrigiert werden. Ist diese Phase abgeschlossen, ist eine erste Version des migrierten Systems fertig.

2.4. jadice flow

Das vorgestellte Framework wurde in dieser Thesis nun erstmals zum Refactoring einer bereits vorhanden Microservices-Architektur in der Industrie verwendet. Deswegen wird in diesem Abschnitt näher auf das Produkt *jadice flow*[1] eingegangen, welches mit dem MMF überarbeitet wurde.

jadice flow ist ein Produkt der Firma *levigo solutions*[2]. *levigo solutions* beschäftigt über 30 Mitarbeiter, von denen ungefähr sechs seit vier Jahren an der Anwendung arbeiten. *jadice flow* basiert bereits auf einer Microservices-Architektur, ist jedoch erst die erste Generation nach der Überführung des vorherigen Monolithen *jadice server*[3]. Beide Produkte bieten workflowbasierte Dokumentenverarbeitung an, bei der mit komplexen Datenströmen umgegangen werden kann. Durch den Umstieg auf eine Microservices-Architektur können in der neuen Generation einzelne Verarbeitungsschritte skaliert werden. Der Großteil der Services ist in Java geschrieben, doch durch den Betrieb in Containern ist *jadice flow* prinzipiell unabhängig von der Programmiersprache.

Einen beispielhaften Workflow stellt der *E-Mail converter* dar. Dabei werden E-Mails in Einzelteile aufgeteilt (Körper, Anhänge) und diese dann einzeln in das Zielformat konvertiert. Die einzelnen Arbeitsschritte dabei können parallelisiert und separat skaliert werden. Am Ende werden die Ergebnisse kombiniert. Im Gegensatz zu einfacher PDF-Darstellung von E-Mails in geläufigen Programmen bietet *jadice flow* zusätzliche Features, wie die tiefe Analyse und Darstellung von Archiven in Anhängen.

Da *jadice flow* die erste Generation in Microservices-Architektur ist, wird noch architekturelles Verbesserungspotential vermutet. Durch die Analyse von *jadice flow 1.0* mithilfe des Tools und weiterer Forschung im Rahmen dieser Bachelorarbeit sollte ein Refactoring-Prozess für *jadice flow 2.0* angestoßen werden.

Obwohl das Framework ursprünglich für die Anwendung auf Monolithen gedacht ist, wurde es in dieser Arbeit auf ein bereits migriertes System angewendet, um dessen Eigenschaften weiter zu verbessern. Dadurch wurde erforscht, inwiefern das Framework und das Tool für diesen Spezialfall einer bereits existierenden Microservices-Architektur geeignet sind. Falls es deshalb für diesen Anwendungsfall zielführend gewesen wäre, hätte zusätzlich die ursprüngliche monolithische Anwendung von *jadice flow* (*jadice server*) hinzugezogen werden können. Ein Vergleich der Ergebnisse dessen mit *jadice flow* wäre dann möglich gewesen.

[1] https://www.levigo.de/jadice-flow/
[2] https://www.levigo.de/dokumentenmanagement/
[3] https://www.levigo.de/jadice-server/

2.5. Ähnliche Forschung

Aufgrund vieler Vorteile von Microservices vor allem direkt gegenüber traditionellen Monolithen (ausführlicher im Abschnitt 2.1 diskutiert) ist die Migration zu Microservices zu einem großen Forschungsfeld geworden. In diesem Abschnitt wird ein Überblick über für diese Arbeit relevante Publikationen gegeben.

2.5.1. Grundlagen zum MMF

Das Hauptuntersuchungsobjekt dieser Thesis sind das MMF und der ARH des ESE. Obwohl das Tool sich noch in einer prototypischen Phase befindet, wurde es bereits durch mehrere akademische Publikationen weiterentwickelt. Im Folgenden wird ein Überblick über diese gegeben. Im Jahr 2019 kategorisierten Fritzsch et al. [FBZW19] zehn verschiedene Microservices-Migrationsmethoden. Dabei wird ein Mangel an praktisch anwendbaren Methoden hervorgehoben, die gute Toolunterstützung und Metriken zur Verifikation der Ergebnisse bieten. Als Folge darauf stellten Fritzsch et al. [FBH+22] die Planung eines solchen Frameworks vor. Der Migrationsprozess mit Framework und dessen Arbeitsweise in drei Phasen wird beschrieben. Außerdem wird die Entwicklung eines Tools erwähnt, dass das Framework in Form einer webbasierten Lösung umsetzt und Entwickler bei der Migration unterstützt. Die Entwicklung des Tools fand im Rahmen der Masterarbeiten von Haller [Hal22] und Koch [Koc22] statt. Haller forscht dabei an der erweiterbaren Speicherung von verschiedenen Migrationsmethoden und der Darstellung dieser in der Web-Applikation, dem ARH. Damit wurde größtenteils das Backend und Phase 2 des Tools umgesetzt. Schlussendlich wird außerdem eine empirische Bewertung des entstandenen Prototyps vorgenommen, die den Nutzen des Tools bestätigt und weiterführende Arbeit daran vorschlägt. Koch [Koc22] setzt die Arbeit an dem Tool fort, indem Phase 1 und 3 ergänzt werden. Dabei wird viel an QAs in der Migration zu Microservices geforscht. Zu Beginn werden relevante QAs gesammelt, wobei ein Großteil davon aus ISO 25010 [Int11] stammt. Diese werden durch weitere, oft Microservices-kontextuelle QAs, ergänzt. Außerdem wird der Einfluss der QAs auf Systemeigenschaften untersucht. Eine weitere durchgeführte Untersuchung ist für den ARH ebenfalls sehr relevant. Dabei wird gezeigt, wie QAs genutzt werden können, um für die QAs geeignete Migrationsverfahren sowie Best Practices und Patterns vorzuschlagen. Die Ergebnisse wurden später verwendet, um die Suchfunktionen in Phase 2 und 3 des ARH auf Basis der vom Nutzer in Phase 1 definierten QAs zu implementieren.

In 2023 führt Knodel im Rahmen einer Masterarbeit [Kno23] eine erste Fallstudie zur Anwendung des MMF durch. Dabei wird das Framework im Ganzen als geeignet für die Migration im Einzelfall der Fallstudie bestätigt. Außerdem werden kleinere Mängel hervorgehoben, die weitere Arbeit am Framework und Tool nahelegen. Knodels Fallstudie unterscheidet sich von der in dieser Arbeit durchgeführten Fallstudie in einigen Punkten. Zum einen bilden die Arbeiten verschiedene Anwendungsfälle ab. Während in Knodel [Kno23] der vermutlich gewöhnlichere Fall vorliegt, bei dem ein Monolith zu einer MSA umstrukturiert werden soll, fügt diese Arbeit einen neuen Anwendungsfall hinzu. Die Evaluierung und Verbesserung einer bereits vorhandenen MSA erfordert möglicherweise andere Qualitäten des Frameworks, insbesondere bei der gezielten Suche nach Verfahren, die für diesen speziellen Fall überhaupt anwendbar sind. Zum anderen steht eine weiter ausgeprägte Version des Tools ARH zur Verfügung. Während die Suche nach Migrationsverfahren in der Arbeit von Knodel manuell ausgeführt wurde [Kno23], kann zum Zeitpunkt dieser Arbeit die Suchfunktion des ARH dafür genutzt werden.

2.5.2. Andere Frameworks

Im Gegensatz zu den im nächsten Abschnitt behandelten Refactoringverfahren, die in großer Zahl akademisch publiziert wurden, liegt nur sehr wenig Forschung zu Verfahren auf der Metaebene vor [MBC23]. Das Funktionsprinzip des MMF ist dabei bisher einzigartig. Wie auch Knodel [Kno23] beschreibt, ist kein Framework bekannt, das den Vergleich und die Suche über viele konkrete Migrationsverfahren aus der Metaebene anbietet und damit direkt mit dem MMF vergleichbar ist. Es existieren nur wenige andere Frameworks, die im Gegensatz zu spezifischen Migrationsverfahren ebenfalls in der Metaebene Migrationsprozesse anleiten. Jene sollen unter diesem Aspekt mit dem MMF in Relation gesetzt werden.

Medeiros et al. [MBC23] stellen die Methode Metaprocess for Microservice Migration Kernel (M3K) vor, die auf dem OMG Essence Kernel [OMG18] basiert. Dieses Tool soll Entwicklerteams beim Migrationvorgang helfen, indem ein allgemeiner Metaprozess definiert wird, der als Grundlage für spezifische Migrationsprozesse verwendet werden kann. Die Evaluation von M3K steht noch aus, Fallstudien und Expertenumfragen sind jedoch geplant. Im Gegensatz zum MMF liefert M3K lediglich eine generische Prozessübersicht. Verweise auf konkrete Methoden, um die einzelnen Schritte jedes Teils des Migrationsprozesses umzusetzen, sind dabei nicht vorhanden. Nutzer werden dadurch bei der Umsetzung der identifizierten Schritte weniger unterstützt als beim MMF.

2.5.3. Konkrete Refactoring-/Migrationsverfahren

Im Gegensatz zu dem, was in dieser Arbeit bisher als *Frameworks* bezeichnet wurde, werden in diesem Abschnitt konkrete Refactoring-Verfahren betrachtet. Diese sind durch die Vorgabe einer spezifischen Strategie klar von Meta-Ansätzen wie dem MMF abzugrenzen. Der ARH speichert momentan 115 solcher Verfahren. Ein wichtiges Feature des MMF und ARH ist dabei die Kategorisierung dieser Verfahren, die eine individuelle Suche für verschiedene Nutzer des Frameworks mit unterschiedlichen Zielen ermöglicht. Um einen Überblick über die Merkmale zu geben, in denen sich Methoden unterscheiden, werden folgend einige in akademischen Publikationen beschriebene Verfahren vorgestellt.

Eines der primären Merkmale ist das Ziel eines Verfahrens. Die meisten Verfahren behandeln das Ziel, eine geeignete Dekomposition von Monolithen in Microservices zu finden, da dies eines der kompliziertesten Probleme bei der Migration darstellt. Jene Verfahren sind für diese Thesis besonders relevant, da in der Forschungsfrage gezielt danach gesucht wird. Beispiele für Methoden, die dieses Ziel verfolgen, sind [ACC+22; DBFP21; DEF+21; MCF+20; VPAG21]. Es gibt jedoch auch wenige andere Verfahren, die andere Ziele haben. Dazu gehören die Prognose der Nützlichkeit einer potenziellen MSA für Systeme vor der Migration [GIT23], die Bewertung und Analyse von Schwachpunkten nach einer Migration [SMNB21] oder die reine technische Umsetzung einer Dekomposition unter Angabe der erwünschten Service-Aufteilung [FFC21].

Die für diese Thesis relevante Kategorie der Methoden, die der Dekomposition eines Monolithen dienen, kann anhand weiterer Merkmale unterteilt werden. Dies umfasst hauptsächlich Merkmale, die die Funktionsweise der Verfahren betreffen. Eine erste Unterscheidung kann bezüglich der Form der Eingabe in die Methoden gemacht werden. Während einige mit Source-Code oder Laufzeit-Artefakten arbeiten und darauf statische und dynamische Analysen ausführen [MCF+20], erwarten andere Eingaben auf Business Process Ebene [DEF+21] oder *User Stories* [VPAG21].

Die meisten Verfahren erstellen dann aufgrund dieser Eingaben einen Graphen mit verschiedenen atomaren Einheiten als Knoten. Die Definition der atomaren Einheiten hängt dabei in der Regel von der Eingabe ab. Anschließend wird bei verschiedenen Methoden mit unterschiedlichen Algorithmen versucht, eine bestmögliche Gruppierung in dem Graph zu finden. Viele Verfahren verwenden unterschiedliche Versionen von *Clustering*-Algorithmen [DBFP21; DEF+21; MCF+20], andere lösen das Problem mit evolutionären beziehungsweise genetischen Algorithmen [ACC+22; VPAG21].

Es gibt verschiedene Stufen der Automatisierung, unter anderem abhängig von den Eingabemöglichkeiten. Vor allem Verfahren, die mit Code-Analysen arbeiten, sind teilweise voll automatisiert [FQA+23]. Für andere Eingabearten wie Business Process (BP) oder *User Stories* ist höchstens eine semi-automatische Umsetzung möglich, da diese Eingaben manuell erstellt werden müssen. Die algorithmische Optimierung eines erzeugten Graphen ist in der Regel theoretisch automatisch möglich. Allerdings ist dies nicht immer umgesetzt oder es wird lediglich auf mögliche Algorithmen für die Umsetzung verwiesen.

3. Methodik

In diesem Kapitel wird der methodische Aufbau dieser Thesis beschrieben und begründet. Die Arbeit ist als industrielle Fallstudie in Zusammenarbeit mit *levigo solutions* konzipiert und dient der Evaluation des MMF an einer bestehenden Microservices-Architektur. Diese Fallstudie besteht aus zwei Hauptbestandteilen. Im ersten Teil wird ein Refactoring des Produktes *jadice flow* nach Anleitung des Frameworks durchgeführt. Im zweiten Teil wird das Ergebnis dieses Refactorings verwendet, um eine Evaluation des Frameworks und Tools abzuschließen.

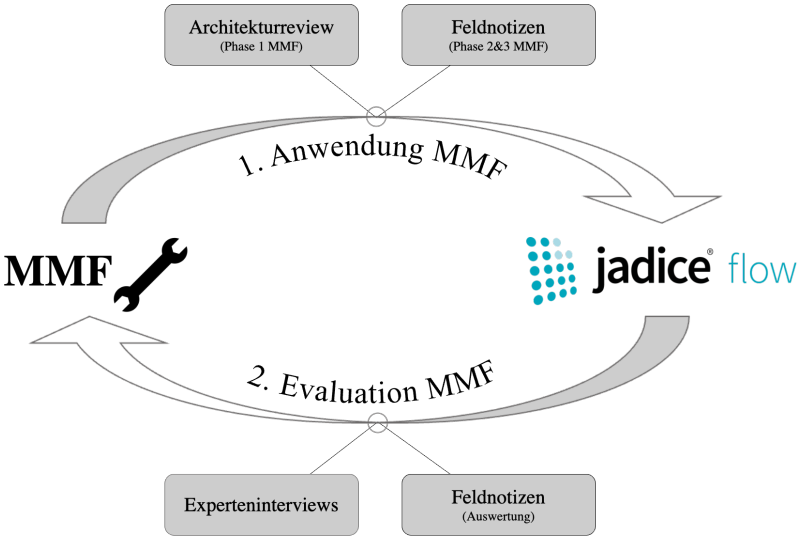

Abbildung 3.1.: Überblick über die Methodik dieser Fallstudie.

3.1. Anwendung des MMF

Die Vorgehensweise beim Refactoring ist durch das Framework sehr genau vorgegeben. Daher wird das Refactoring in dieser Thesis in die gleichen drei Phasen unterteilt wie in Abschnitt 2.3 beschrieben. In der ersten Phase wird gemeinsam mit den Stakeholdern des Produkts ein Architekturreview nach Svahnberg und Mårtensson [SM07] durchgeführt. Diese Methode wird in Abschnitt 3.1.1 näher beschrieben. Das wesentliche Ergebnis dieses Reviews sind die gewünschten QAs des Systems. Diese Attribute sind die Basis für Phase 2 und 3 und werden in den ARH eingegeben. Dieser führt Berechnungen mit den QAs durch und kann so eine Liste von Refactoring-Methoden vorschlagen, die nach ihrer Eignung für die QAs sortiert sind. Als weitere Eingabe für diese Suche dienen bestimmte Filter, die in Abschnitt 4.2 näher erläutert werden. Aus der resultierenden Liste von Migrationsmethoden werden die besten Vorschläge betrachtet und bewertet, bevor einer davon ausgewählt und in den folgenden Phasen verwendet wird. Analog dazu wird in Phase 3a auf Basis

der QAs und Filter aus Phase 2 eine sortierte Liste von Patterns und Best Practices vorgeschlagen. Das Vorgehen bei der Auswahl der Filter sowie die spätere Inspektion der Vorschläge des ARH wurde in Form von strukturierten Feldnotizen protokolliert. Deren genauere Funktion und Form wird in Abschnitt 3.1.2 näher erläutert.

3.1.1. Architekturreview

Die Methodik eines Architekturreviews wurde in dieser Thesis nach Vorschlag des MMF/ARH dazu verwendet, Qualitätsanforderungen an das System, sogenannte Qualitätsattribute (QAs), zu sammeln. Für Konformität mit dem Tool ARH, das als Ergebnis des Architekturreviews Szenarien benötigt, die Qualitätsattribute (QAs) beschreiben, beschränkt sich die Auswahl möglicher Verfahren für die erste Phase auf szenarienbasierte Architekturreviews. Das bedeutet, dass die gewünschten QAs des Systems in Form von Szenarien erfasst und dokumentiert werden. Ein Szenario beschreibt dabei eine Interaktion eines Stakeholders mit dem Produkt [KKC00]. Diese Konkretisierung von Qualitäten soll vor allem die Verständlichkeit fördern. Dabei werden mit jedem Szenario zugehörige QAs assoziiert, wodurch am Ende eine Abbildung aller QAs auf ihre Priorität möglich ist. Diese Abbildung ist das eigentliche Ergebnis, das der ARH als Eingabe für die weiteren Phasen benötigt. Die Szenarien sind dabei lediglich ein Zwischenprodukt und sollen dabei helfen, eine genauere Vorstellung davon zu bekommen, welche spezifischen Situationen mit den QAs verbunden sind. Sie ermöglichen auch eine Bewertung der Wichtigkeit und technischen Schwierigkeit anhand konkreter Situationen, was bei abstrakten QAs schwieriger wäre.

Folgend werden verschiedene akademische Methoden beschrieben und verglichen und anschließend die verwendete Methode als Abwandlung dieser Methoden erläutert.

Vergleich verschiedener Methoden

Geläufige szenarienbasierte Verfahren sind die Software Architecture Analysis Method (SAAM) nach Bass et al. [BCK98] und die Architecture Trade-off Analysis Method (ATAM) nach Kazman et al. [KKC00]. ATAM ist der Nachfolger und eine Überarbeitung von SAAM, weshalb folgend lediglich ATAM näher betrachtet wird. Die Autoren beschreiben mit ATAM eine Methode, die aus insgesamt neun Schritten besteht. Diese Schritte sollen mit den Stakeholdern des Produkts innerhalb von drei Arbeitstagen durchgeführt werden. Da das Ausmaß dieser allerdings nicht angemessen für den Rahmen einer Thesis ist [Kno23], würde für diese Arbeit höchstens eine stark abgewandelte Version dieser infrage kommen. Stattdessen wurde sich in dieser Thesis für eine Modifikation des Architekturreviews nach Svahnberg und Mårtensson [SM07] entschieden. Es handelt sich dabei ebenfalls um eine szenarienbasierte Methode, die auf SAAM und ATAM basiert, jedoch spezieller für kleinere Anwendungsfälle entwickelt wurde. Die Autoren erläutern die Verwendung dieser zur Evaluation von Studentenprojekten, die ein reales Softwareentwicklungsprojekt mit Industriekunden simulieren und einen Rahmen von 10-15 Teilnehmern und 20 Wochen Arbeitszeit vorsehen.

Architekturreview nach Svahnberg und Mårtensson [SM07]

Das Architekturreview nach Svahnberg und Mårtensson [SM07] umfasst sechs Schritte (Übersicht in Tabelle 3.1). Diese sollen in einer etwa vierstündigen, moderierten Gruppendiskussion mit den

Aktivität	Zeit (in Minuten)
1. Einleitung in das Projekt	15
2. Identifikation von Qualitätsanforderungen	20
3. Szenarienerhebung	45
Pause	20
4. Architekturpräsentation	20
5. Szenario- und Architekturanalyse	90
6. Fazit	15

Tabelle 3.1.: Zeitplan für ein Architekturreview nach Svahnberg und Mårtensson [SM07].

wichtigsten Stakeholdern des Produkts durchgeführt werden. Im ersten Schritt gibt der Product Owner (PO) eine Einführung in das Projekt, in der Ziele für das Produkt sowie die Organisation geschildert werden. Im zweiten Schritt wird in einer Brainstorming-Runde zusammen mit dem PO, End-Nutzern und Architekten die Identifikation von QAs durchgeführt. Hierfür dient die Liste der Qualitätskriterien gemäß ISO 9126 [Int00] als Vorlage für mögliche QAs. Im dritten Schritt ordnen PO und End-Nutzer die Liste der QAs nach ihrer Wichtigkeit und fügen den drei wichtigsten jeweils zwei Szenarien hinzu. Nach einer Pause präsentiert im vierten Schritt ein Softwarearchitekt die vorgeschlagene Architektur. Anschließend werden im fünften Schritt die Szenarien sequenziell analysiert und bewertet, inwiefern sie durch die Architektur erfüllt werden und wo etwaige Probleme bestehen. Im sechsten Schritt werden alle identifizierten Probleme und Aspekte, die mehr Aufmerksamkeit benötigen, final diskutiert und der gesamte Prozess zusammengefasst.

Anpassung des Architekturreviews auf den ARH

Ein volles Architekturreview, auf Deutsch auch mit Architekturbewertung zu übersetzen, ist generell bei der Verwendung des ARH und somit in dieser Thesis nicht das primäre Ziel. Vielmehr entfällt das Hauptziel eines herkömmlichen Architekturreviews: die Bewertung. Stattdessen wurde die in Tabelle 3.2 dargestellte Methodik zur Extraktion von Szenarien und QAs entworfen.

Schritt nach Svahnberg und Mårtensson [SM07]	Sub-Schritt	Zeit
1. Einleitung in die Methodik		10 min
2. Identifikation von Qualitätsanforderungen	1. Umfrage über die wichtigsten drei Sub-QAs 2. Diskussion über Priorisierung der Sub-QAs	20 min
3. Szenarienerhebung	1. Erstellen Szenarien zu den wichtigsten Sub-QAs 2. Hinzufügen Wichtigkeit & technische Schwierigkeit zu Szenarien 3. Szenarien mit weiteren QAs assoziieren	60 min
5. Szenario- und Architekturanalyse	1. Architekturbewertung anhand der Szenarien	10 min

Tabelle 3.2.: Zeitplan für das in Phase 1 dieser Thesis durchgeführte Architekturreview.

Wie in der Tabelle angemerkt, dienen als Vorlage für die Methodik die ersten Schritte des Architekturreviews nach Svahnberg und Mårtensson [SM07]. Außerdem werden auch einzelne Aspekte von ATAM [KKC00] verwendet. Die einzelnen Schritte werden folgend näher beschrieben.

Im ersten Schritt ist die Einleitung in die Methodik geplant. Dabei handelt es sich um eine Modifikation des nach Svahnberg und Mårtensson [SM07] geplanten ersten Schritts, bei dem stattdessen eine Einleitung in das Projekt erfolgt. Dieser stammt vermutlich aus dem Kontext der Studentenprojekte, bei denen mehrere Architekturreviews am selben Tag von den gleichen Personen durchgeführt werden und somit eine Einleitung in das Projekt nötig ist. In den meisten Fällen in der Industrie, und so auch in diesem, ist dieser Schritt nicht notwendig, da die Teilnehmer der Besprechung, die Stakeholder, bereits mit dem Produkt vertraut sind. Stattdessen wird (wie in ATAM [KKC00]) im ersten Schritt eine Einleitung in die verwendete Methodik gegeben, da diese dem Team noch unbekannt war.

Mit Schritt 2 beginnt das eigentliche Architekturreview. Dabei werden die wichtigsten QAs in zwei Sub-Schritten gesammelt: Zunächst erfolgt eine Umfrage, bei der die Teilnehmer jeweils eine Einschätzung über die drei wichtigsten Sub-QAs abgeben. Anschließend wird in einer gemeinsamen Diskussion eine Priorisierung der Sub-QAs finalisiert. Als Basis-QAs werden statt der ISO 9126 [Int00] nach Svahnberg und Mårtensson [SM07] die von Koch [Koc22] speziell für Microservices-Architekturen identifizierten QAs verwendet. Diese QAs sind in Abbildung 3.2 dargestellt. Es handelt sich hierbei größtenteils um die wichtigsten QAs aus der ISO 25010 [Int11],

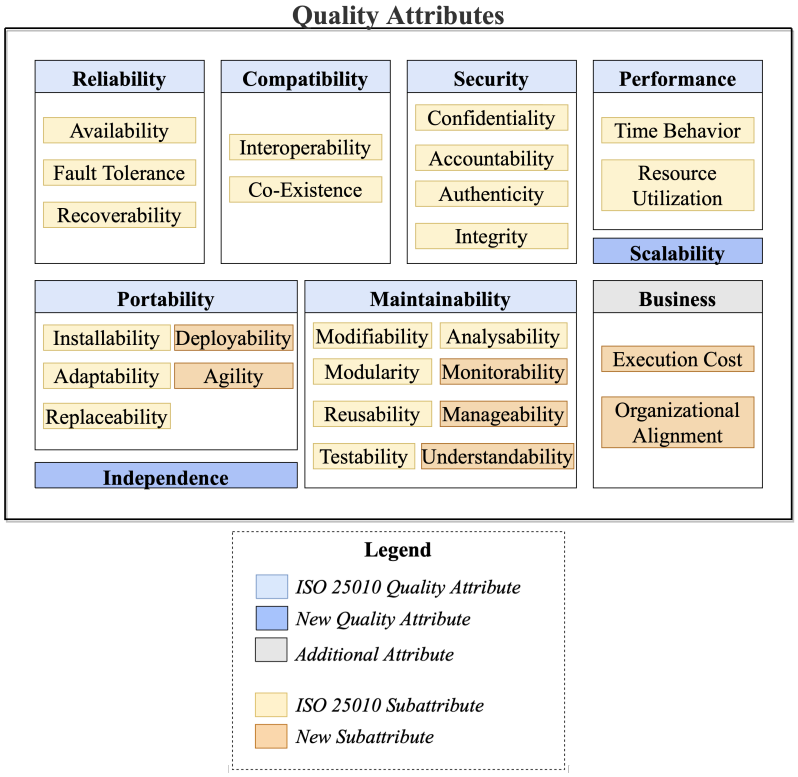

Abbildung 3.2.: Spezielle Qualitätsattribute (QAs) für Microservices-Architekturen nach Koch [Koc22].

die durch einige spezifische QAs für Microservices ergänzt wurden. Aufgrund der Verwendung des ARH ist es sinnvoll, diese zu nutzen, da der ARH nur diese zulässt.

Die im zweiten Schritt erstellte Priorisierung der QAs wird im dritten Schritt dann verwendet, um Szenarien zu erstellen. Dieser ist in drei Sub-Schritte unterteilt: Erst werden für die wichtigsten QAs Szenarien erstellt. Diese werden in Form eines Utility Trees festgehalten (siehe Abbildung 3.3). Dann wird inspiriert von ATAM [KKC00] jedes Szenario erneut betrachtet und eine Bewertung hinsichtlich der Wichtigkeit und technischen Schwierigkeit vorgenommen. Diese erfolgt auf einer Skala von A bis C, wobei A für die höchste Wichtigkeit und höchste technische Schwierigkeit steht. Abschließend wird jedes Szenario ein drittes Mal betrachtet und die Assoziation mit anderen QAs diskutiert und festgehalten. Bei diesem Schritt stammt nur der erste Teil aus der Methodik von Svahnberg und Mårtensson [SM07] und die anderen beiden Schritte stammen aus dem ARH. Dass bei der Vorlage von Svahnberg und Mårtensson [SM07] keine vergleichbare Bewertung der Szenarien und Assoziation der Szenarien mit weiteren QAs beschrieben ist, liegt vermutlich am Kontext des Architekturreviews. Im Kontext der qualitativen Bewertung durch Menschen sind diese nicht explizit notwendig. Im Vergleich dazu können bei der automatisierten Auswertung durch ein Tool wie den ARH die Szenarien nicht selbstständig eingeschätzt werden. Außerdem stammt die Visualisierung durch einen Utility Tree aus ATAM [KKC00]. Ein solcher soll dabei helfen, Qualitätsziele zu konkretisieren und zu priorisieren [KKC00]. Dabei werden nach einem semantisch unbedeutenden Wurzelknoten von links nach rechts die QAs in Sub-QAs verfeinert, worauf dann auf dritter Ebene Szenarien folgen. Wie in Abbildung 3.3 zu sehen ist, enthält die in dieser Thesis verwendete Notation des Utility Trees auch die Einschätzung der Szenarien in Wichtigkeit und technische Schwierigkeit.

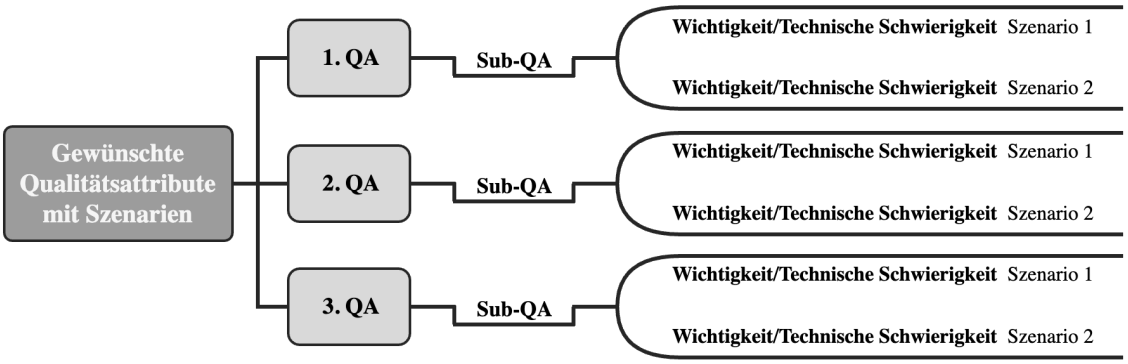

Abbildung 3.3.: Utility Tree nach ATAM [KKC00], der auf die Verwendung mit dem ARH angepasst ist.

Im letzten Schritt erfolgt eine Bewertung, bei der jedem Szenario zugeordnet wird, ob eine MSA dafür vorteilhaft ist. Diese Bewertung kann als Teil des fünften Schritts nach Svahnberg und Mårtensson [SM07] gesehen werden. Abgesehen davon wird allerdings der Teil der Bewertung der Architektur unter den Szenarien im Architekturreview komplett ausgelassen und der Rest der Schritte 4, 5 und 6 nach Svahnberg und Mårtensson [SM07] nicht durchgeführt. Während bei allen genannten szenarienbasierten Verfahren die Szenarien und QAs lediglich ein Zwischenprodukt und Mittel zur Bewertung der Systemarchitektur sind, stellen sie bei dem im Rahmen dieser Thesis durchgeführten Architekturreview das gewünschte Endprodukt dar.

Die vorgestellte Methodik kann als neue Methodik zur Extraktion von Szenarien und QAs betrachtet werden und könnte in Zukunft als Vorlage für die Anwendung mit dem ARH verwendet werden. Trotz des Fehlens des Bewertungsschritts der Methodik wird sie folgend zur Vereinfachung als Architekturreview bezeichnet.

3.1.2. Strukturierte Feldnotizen

Während Phase 1 mit der Methodik einer Fokusgruppe durchgeführt wird, findet die Bearbeitung in den folgenden Phasen größtenteils in Einzelarbeit statt. Um dabei strukturiert zu protokollieren, welche Aktivitäten im Rahmen dieser Phasen durchgeführt werden, werden systematisch durchgeführte Aktivitäten mit gemachten Erfahrungen und Herausforderungen dokumentiert. Dazu werden wie in der ähnlichen Arbeit von Knodel [Kno23] strukturierte Feldnotizen nach Seaman [Sea08] verwendet.

Jede Feldnotiz wurde zeitnah nach Beobachtung in einer zugehörigen Datei mit dem Namensschema F*Nummer*_P*PhaseMMF*_*Datum*_*Name*.tex erstellt. Dabei wurden dementsprechend Nummer der Feldnotiz, Phase des MMF, Datum (im Format *MM_DD_JJJJ*) und Name der Feldnotiz im Dateinamen festgehalten. Feldnotiz D.1 hat beispielsweise den Namen *F01_P2_11_20_2023_FilterAuswahl.tex*. Tabelle 3.3 gibt einen Überblick über die Eintrags-Felder und welchem Feld nach Seaman [Sea08] sie entsprechen.

Eintrag	**Eintrag nach Seaman [Sea08]**
Feldnotiz Nr.	-
Datum, Uhrzeit	Zeit
Ort	Ort
Beteiligte Personen	Teilnehmer der Beobachtung
Phase & Schritt des MMF	Verfolgte Ziele
Aktionen und Entscheidungen	Stattgefundene Handlungen und Diskussionen
Kommentare	Kommentare
Gut gelaufen	Diskussion
Schlecht gelaufen	Diskussion
Empfindungen	Ton und Stimmung der Interaktionen

Tabelle 3.3.: Aufbau der strukturierten Feldnotizen nach Seaman [Sea08].

3.2. Evaluation des MMF

Das Forschungsziel dieser Arbeit ist die Evaluation des MMF und des ARH. Nach dem Refactoring von *jadice flow* mit dem Framework wurde evaluiert, als wie hilfreich sich dieses bei der Optimierung einer bereits vorhandenen MSA herausgestellt hat. Dafür wurden Experteninterviews durchgeführt sowie die im Verlauf der Anwendung erstellten Feldnotizen ausgewertet.

3.2.1. Experteninterviews

Im Rahmen dieser Arbeit wurden Experteninterviews durchgeführt, um die Nützlichkeit des MMF und ARH bei der Migration zu Microservices im Spezialfall *jadice flow* zu evaluieren. Im Software Engineering allgemein werden Interviews oft genutzt, um Softwareprozesse zu bewerten [Sea08].

Aufbau

Es gibt verschiedene Arten von Interviews sowie viele verschiedene Vorgehensweisen bei der Planung, Durchführung und Auswertung der Interviews. In den nächsten Abschnitten wird beschrieben und begründet, welche Art und welche Vorgehensweisen verwendet werden.

Art: Ein wichtiges Unterscheidungsmerkmal bei Interviews ist die Offenheit der Fragen. Seaman [Sea08] beschreibt zwei Ausprägungen: strukturierte und unstrukturierte Interviews. Während strukturierte Interviews klare Antworten auf spezifische Fragen erwarten, haben unstrukturierte Interviews den Vorteil, dass auch unvorhergesehene Informationen zu den Antworten auf offene Fragen gehören können. Um die Vorteile beider Varianten zu kombinieren, wurden in dieser Arbeit semi-strukturierten Interviews [Sea08] verwendet. Dabei wurden spezifische Fragen gestellt, aber auch offene Fragen mit eingebunden, um die Flexibilität der potentiellen Ergebnisse zu erhöhen.

Experten: Wie aus dem Namen hervorgeht, ist die Auswahl der Experten eine wichtige Entscheidung bei der Planung von Experteninterviews. Die für die Interviews relevante Expertise ist „Betriebswissen“ [MN09]. Als Experten wurden Mitarbeiter mit guter Kenntnis des Produkts *jadice flow* ausgewählt, da die Fragen einen besonderen Bezug zur Nützlichkeit des Frameworks für das Produkt haben. Runeson und Höst [RH09] empfehlen, aufgrund der qualitativen Natur von Fallstudien, möglichst verschiedene Teilnehmer auszuwählen, anstatt nach Ähnlichkeiten zu suchen. Aus diesem Grund wurde bei der Wahl der Mitarbeiter nach ihrer Rolle in der Produktentwicklung von *jadice flow* differenziert. Für die abstrakteste Perspektive auf das Produkt wurde der Product Owner (nach Definition des SCRUM von Schwaber und Sutherland [SS11]) ausgewählt. Für die technische Perspektive wurde der Softwareentwickler (SE) hinzugezogen. Als weitere Meinung einer Person, die zwischen den beiden genannten Rollen liegt, wurde der Softwarearchitekt (SA) herangezogen.

Protokollierung: Zwei geläufige Optionen für die Aufnahme eines Interviews sind die Protokollierung während des Interviews, entweder durch den Interviewer oder einen separaten Protokollant [RH09; Sea08]. Der Interviewer selbst ist besonders im Fall von Unerfahrenheit keine gute Wahl, da es kompliziert sein kann, zwei Aufgaben gleichzeitig zu bewältigen und die Leitung des Interviews somit leiden kann. Die Lösung durch einen Protokollanten ist auch nicht optimal, da ein solcher nicht immer zur Verfügung steht. Außerdem kann es zu Interpretationsunterschieden kommen, welche Informationen relevant sind. Deswegen wurde, wie von Seaman [Sea08] sowie Runeson und Höst [RH09] empfohlen, eine Aufzeichnung durch Audioaufnahme verwendet.

Leitfaden: Eine weitere wichtige Vorbereitung für ein gut geplantes Interview ist die Erstellung eines Leitfadens [HA05; Sea08], welcher den Plan des Vorgehens enthält. Dieser muss nicht so strikt gegliedert sein wie ein Fragebogen, sondern soll neben den Fragen Notizen für den Interviewer enthalten, die bei der Leitung des Interviews helfen. Der für diese Interviews konzipierte Leitfaden ist in Anhang A zu finden. Er besteht aus hauptsächlich drei Teilen. Im ersten Abschnitt

werden Verständnisfragen gestellt. Diese sollen absichern, dass die Informationen aus dem Informationsbogen ausreichend verstanden wurden, um die folgenden Fragen sinnvoll beantworten zu können. Im zweiten Teil werden allgemeine inhaltliche Fragen zur Evaluation des MMF gestellt, ohne Bezug zur Anwendung in dieser Thesis. Mithilfe mehrerer Fragen sollen mögliche alternative Funktionsweisen, die Phasen des Frameworks und potenzielle Probleme dessen evaluiert werden. Der letzte Teil dient der Bewertung der spezifischen Anwendung des Frameworks in dieser Thesis. Dazu werden zunächst Fragen zu den mit dem ARH ermittelten Migrationsverfahren gestellt. Da ein gutes Verständnis der Verfahren dafür nötig ist, werden die Fragen weiter spezifiziert, falls der Kandidat bei offenen Fragen in falsche Richtungen abschweift oder keine Antwort findet. Außerdem wird für Bezug zur Forschungsfrage die Auswirkung der Verfahren auf die Granularität des Systems befragt. Anschließend werden die mit ARH erhaltenen Best Practices und Patterns bewertet. Auch hier werden zunächst offene Fragen gestellt und am Ende eine Frage mit Bezug zu den Qualitätsaspekten aus der Forschungsfrage gestellt. Zum Abschluss werden dem Interviewten noch zwei Fragen zur Gesamtbewertung der Anwendung des Frameworks vorgelegt. Dabei soll er seine Einzelbewertungen der letzten Schritte zusammenfassen und ein Fazit ziehen.

Vorbereitungsmaterial: Die Einleitung eines Interviews sollte nur einen kleinen Teil ausmachen, da Zeit in Interviews wertvoll und oft knapp bemessen ist [HA05; RH09; Sea08]. Aus diesem Grund wurden für die Interviews eine Präambel und ein Informationsbogen erstellt, welche die Teilnehmer im Voraus erhalten haben. Diese sind in den Anhängen B und C zu finden. Die Präambel fasst die generellen Richtlinien für die Experteninterviews zusammen. Dabei diente die Präambel von Bogner et al.[1] als Vorlage. Da der thematische Kontext des Interviews durch die Voraussetzung des Verständnisses von zwei Migrationsmethoden relativ komplex ist, wurde außerdem ein Informationsbogen erstellt, der bei der thematischen Einarbeitung helfen soll. Darin werden alle wichtigen Konzepte erklärt, die in den Interviews behandelt werden. Der Bogen enthält einen generellen Überblick über MMF/ARH sowie die Funktionsweisen der zwei diskutierten Migrationsverfahren. Es wird jedoch nicht erwartet, dass die Teilnehmer alle Inhalte im Voraus vollständig verstehen, da (wie im Leitfaden beschrieben) anfangs eine Runde zur Klärung der Verständnisfragen stattfindet.

Analyse der Daten

Die Analyse qualitativer Daten ist oft ein Prozess der parallel zu der Datensammlung stattfindet [RH09]. Es ist wichtig, einen gut strukturierten und systematischen Plan für die Datenanalyse zu haben, da die Erkenntnisse in der Analyse die Sammlung der nächsten Daten beeinflussen können. Wie Seaman [Sea08] empfiehlt, wurde die Datenanalyse deswegen im Voraus geplant. Obwohl es empfehlenswert ist, die Auswertung durch mehrere Forschende durchzuführen, um Voreingenommenheit zu reduzieren [RH09], wurde sie in dieser Arbeit aufgrund mangelnder Ressourcen nur vom Autor durchgeführt.

Die Rohdaten liegen in Form von drei Audioaufnahmen vor, die jeweils etwa eine Stunde lang sind. Die Audioaufnahmen wurden mithilfe der von Mayring und Fenzl [MF19] vorgestellten Techniken *Zusammenfassen* und *Explikation* getrennt transkribiert (siehe Abbildung 3.5). Die Ergebnisse dieser Transkription wurden auf zwei verschiedene Arten in diese Thesis übernommen. Zum einen wurde eine *Strukturierung* nach Mayring und Fenzl [MF19] durchgeführt. Dabei wurde für jede gestellte

[1] https://github.com/xJREB/research-microservices-interviews/blob/master/interview-preamble.md

Frage die Antwort kategorisiert. Das dafür entworfene „ordinal geordnete Kategoriensystem“ nach Mayring und Fenzl [MF19] ist in Abbildung 3.4 zu sehen. Auf der anderen Seite wurden die Transkripte sowie die Kodierungen in textueller Form aggregiert. Eine Übersicht über die dadurch entstandenen aggregierten Zusammenfassungen ist in Tabelle 3.4 gegeben.

Fragenabschnitt	**Frage**	**Kategorien**
Evaluation des MMF/ARH allgemein	Welche Aufgaben soll ein MMF übernehmen?	ARH übernimmt: alle / viele / manche / keine
	Gewünschte Qualitäten dabei	ARH erfüllt: alle / viele / manche / keine
	Funktionsweise und Phasen des MMF/ARH sinnvoll?	ja / größtenteils / teilweise / nein
	Probleme beim Framework	keine / wenig / einige / viele
Evaluation der gewählten Migrationsmethoden für *jadice flow*	Allgemeine Bewertung	gut / eher gut / eher schlecht / schlecht
	Bewertung auf abstrakter Ebene	gut / eher gut / eher schlecht / schlecht
	Probleme auf weniger abstrakter Ebene	keine / wenig / einige / viele
	Verbesserung der Granularität möglich?	ja / eher ja / eher nein / nein
Evaluation der Best Practices und Pat-terns für *jadice flow*	Sortierung passend?	ja / eher ja / eher nein / nein
	Anzahl bereits angewendet	viele / manche / wenig / keine
	Anzahl potentiell sinnvoll (zusätzlich zu angewendeten)	viele / manche / wenig / keine
	Enthält Lösungen für Optimierung des Kommunikationsmodells/ Netzwerk-Overheads	ja / eher ja / eher nein / nein
Evaluation der Anwendung des MMF/ ARH auf *jadice flow*: Zusammenfassung	Gewinnbringend für *jadice flow*?	ja / eher ja / eher nein / nein
	Gewinnbringend für *jadice server*?	ja / eher ja / eher nein / nein

Abbildung 3.4.: Kodierleitfaden für die Auswertung der Experteninterviews.

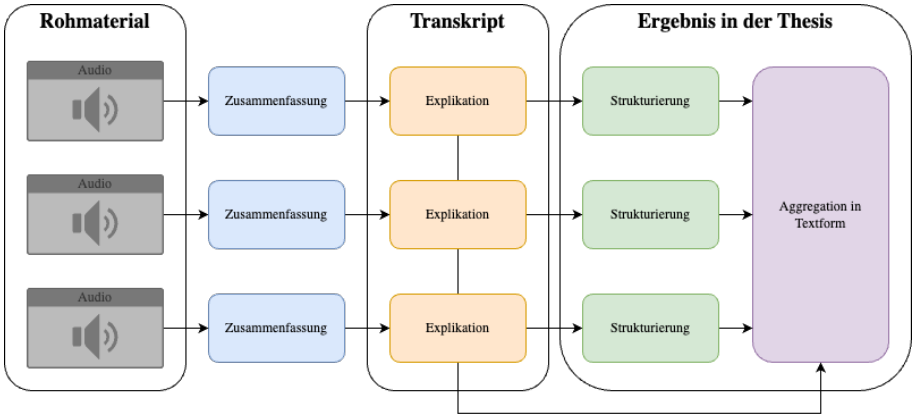

Abbildung 3.5.: Vorgehen bei der Datenanalyse der Experteninterviews.

Kategorie	Fragenabschnitt	Abschnitt
Evaluation des MMF/ARH allgemein		Abschnitt 5.1.1
Evaluation der Anwendung des MMF/ARH auf *jadice flow*	Evaluation der ausgewählten Migrationsmethoden für *jadice flow*	Abschnitt 5.1.2
	Evaluation der Best Practices und Patterns für *jadice flow*	Abschnitt 5.1.2
	Evaluation der Anwendung des MMF/ARH auf *jadice flow* insgesamt	Abschnitt 5.1.2

Tabelle 3.4.: Abschnitte der Auswertung der Experteninterviews.

3.2.2. Strukturierte Feldnotizen

Nachdem alle Feldnotizen erstellt waren, wurden diese qualitativ ausgewertet. Auch hier wurde die Strukturierung nach Mayring und Fenzl [MF19] verwendet. Dafür wurde eine Kodierung erstellt, welche in Tabelle 3.5 zu sehen ist. An vielen Stellen wurde diese aus der ähnlichen Arbeit von Knodel [Kno23] inspiriert.

Bewusst wurde an einigen Stellen vom Original abgewichen. Die Definition der Codes „Mehr gut gelaufen“ und „Mehr schlecht gelaufen“ über die Anzahl der positiven Aspekte im Vergleich zur Anzahl der negativen Aspekte wurde zum Beispiel nicht übernommen. Es wurde sich stattdessen für eine Definition dieser über die Wahrnehmung des Autors nach Betrachtung der positiven und negativen Aspekte der jeweiligen Feldnotiz entschieden. Das hat den Nachteil einer geringeren Objektivität. Trotzdem wurde es als sinnvoller eingeschätzt, da sonst keine Gewichtung der positiven und negativen Aspekte stattfindet. Dass die Ergebnisse je nach Definition dieser Kodierung abweichen, wurde bei Betrachtung erkannt und sich deswegen für die angepasste Variante entschieden.

Kodierung	Subkodierungen	Bedeutung
Gut gelaufen		Feldnotiz hat Eintrag bei „Gut gelaufen“
Schlecht Gelaufen		Feldnotiz hat Eintrag bei „Schlecht gelaufen“
Mehr gut gelaufen		Nach subjektiver Bewertung des Autors bei Betrachtung der Einträge in „Gut gelaufen“ und „Schlecht gelaufen“ ist mehr gut gelaufen
Mehr schlecht gelaufen		Nach subjektiver Bewertung des Autors bei Betrachtung der Einträge in „Gut gelaufen“ und „Schlecht gelaufen“ ist mehr schlecht gelaufen
Gutes Gefühl		Eintrag eines guten Gefühls in „Empfindungen“
Schlechtes Gefühl	Unsicherheit	Eintrag eines Gefühls von Unsicherheit in „Empfindungen“
	Frustration	Eintrag eines Gefühls von Frustration in „Empfindungen“
Verbesserungsvorschlag		In der Feldnotiz ist direkt oder indirekt in einem beliebigen Feld ein Verbesserungsvorschlag für den ARH enthalten
Einzelarbeit		Axel Herrmann ist alleinige beteiligte Person an der Aktivität der Feldnotiz
Besprechung		Die Aktivität der Feldnotiz wurde mit anderen Personen in einer Besprechung durchgeführt. Gegenteil von „Einzelarbeit“

Tabelle 3.5.: Verwendete Kodierung für die Auswertung der Feldnotizen.

4. Anwendung des Frameworks auf jadice flow

In diesem Kapitel wird ein Architektur-Refactoring am Produkt *jadice flow* geplant. Dazu werden das Microservices Migration Framework (MMF) beziehungsweise der Architecture Refactoring Helper (ARH) verwendet, welche den Prozess in drei Phasen unterteilen. Eine genauere Beschreibung des Frameworks ist in Abschnitt 2.3 zu finden. Sehr abstrahiert und vereinfacht bestehen die Phasen aus den folgenden Aktivitäten:

- In Abschnitt 4.1 wird die Durchführung der ersten Phase in Form eines Architekturreviews mit den wichtigsten Stakeholdern beschrieben.
- Im Rahmen der zweiten Phase werden in Abschnitt 4.2 Filter konfiguriert und anschließend nach adäquaten Migrationsstrategien gesucht.
- In Abschnitt 4.3 wird in der dritten Phase eine Suche nach geeigneten Best Practices und Patterns beschrieben.

4.1. Phase 1 - Systemverständnis

Das Ziel der ersten Phase ist, ein Verständnis des Systems aufzubauen. Das ist zum einen auf Seite der Stakeholder wichtig. Diese sollten spätestens nach dieser Phase wissen, welche Qualitätsattribute (QAs) besonders wichtig für das System sind. Zum anderen sollte der ARH nach dieser Phase durch Eingabe der QAs ein Verständnis des Systems erlangen, das er in den nächsten Phasen zur Unterstützung der Entwickler bei Migration verwenden kann.

Um diese Eingaben für den ARH zu generieren, wurde am 6. November 2023 ein Architekturreview wie in Abschnitt 3.1.1 beschrieben durchgeführt und am 14. November 2023 fortgesetzt. Teilgenommen haben vier Softwareentwickler beziehungsweise -architekten, der Product Owner nach Definition des SCRUM von Schwaber und Sutherland [SS11] und der Autor selbst in der Rolle des Moderators. Leider war es nicht möglich, einen Kunden oder Nutzer des Produkts als Stakeholder für dieses Review zu organisieren. Da das Entwicklungsteam jedoch regelmäßigen Kontakt mit Kunden hat, versuchten sie bestmöglich, auch die Interessen der Kunden zu repräsentieren. Der Zeitrahmen für diese Besprechung war durch die Planung, die in Abschnitt 3.1.1 beschrieben ist, auf zwei Stunden festgelegt. Im Folgenden wird der Begriff *Schritt* immer auf den Schritt nach der beschriebenen Methodik bezogen und nicht auf Schritte des ARH. Da der erste Schritt dabei lediglich eine Einführung in die Methodik ist, wird direkt mit den Ergebnissen des zweiten Schritts fortgefahren.

4.1.1. Priorisierung der Qualitätsattribute (Schritt 2)

Im zweiten Schritt werden die gewünschten QAs des Systems gesammelt. Im Rahmen dessen sollte jeder Teilnehmer seine Einschätzung darüber, welche die wichtigsten drei Sub-QAs sind, per Nachricht mitteilen. Die daraus resultierende Anzahl von Stimmen pro Sub-QA ist in Abbildung 4.1 dargestellt.

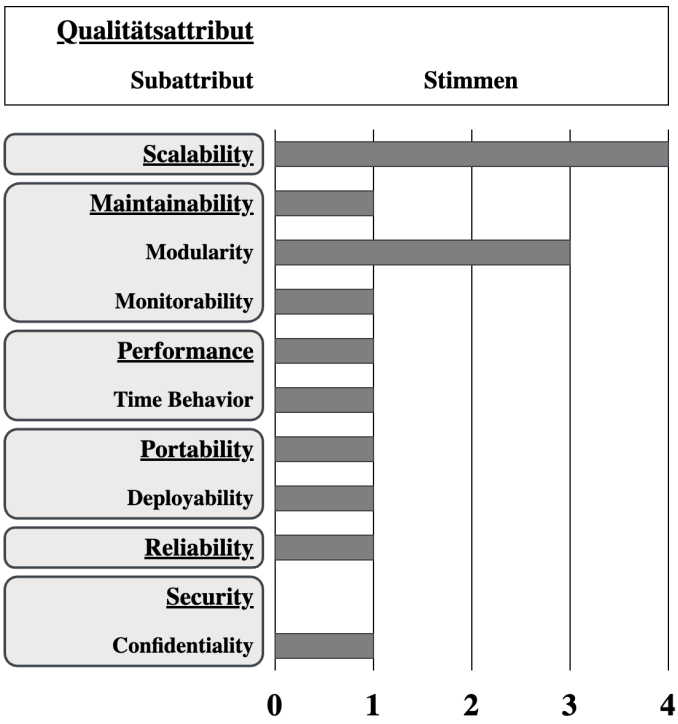

Abbildung 4.1.: Umfrageergebnisse bei der Suche nach den wichtigsten (Sub-) QAs im Architekturreview in Phase 1 der Migration mit N = 5.

Dabei ist auffällig, dass auch Qualitätsattribute Stimmen erhalten haben. Das liegt daran, dass nicht alle Teilnehmer der Umfrage sich auf Sub-QAs beschränkt haben, sondern auch die QAs *Performance*, *Reliability*, *Maintainability* und *Portability* genannt wurden. Auch wenn das nicht so vorgesehen war, wurde davon abgesehen, die Teilnehmer darauf hinzuweisen und es zu korrigieren, da die Ergebnisse nicht direkt zur Priorisierung der Attribute führen müssen. Stattdessen wurden die Umfrageergebnisse und als vage Basis für eine freiere Diskussion über die Priorisierung verwendet. Deren Ergebnis war die schlussendliche Platzierung der sechs wichtigsten Sub-QAs. Im Folgenden werden diese erläutert.

1. ***Scalability*** ist ein QA ohne Subattribute, es wird allerdings auch oft *Performance* zugeordnet. Nach Koch [Koc22] gibt es an, wie effizient ein System skaliert werden kann [BCK12]. Im Kontext von Microservices ist dabei vor allem die horizontale Skalierung relevant, welche die Skalierung über mehrere Instanzen von der Microservices beschreibt [LZJ+21].
2. ***Modularity*** ist ein Subattribut von *Maintainability*. Nach Koch [Koc22] gibt es an, wie gut die Komplexität des Systems auf verschiedene Komponenten verteilt ist [Int11].

3. ***Time Behavior*** ist Subattribut von *Performance*. Nach Koch [Koc22] gibt es an, wie schnell ein System oder Teile eines Systems ankommende Anfragen beantworten [ALFT21; Int11; SZ12].
4. ***Deployability*** ist ein Subattribut von *Portability*. Nach Koch [Koc22] gibt es an, wie einfach und schnell ein Produkt gebaut und ausgeliefert werden kann [BCK12; MM22].
5. ***Fault Tolerance*** ist Subattribut von *Reliability*. Nach Koch [Koc22] gibt es an, inwieweit ein System wie erwartet funktionieren kann, obwohl in Teilen des Systems Fehler auftreten [AAE16; Int11].
6. ***Recoverability*** ist ebefalls Subattribut von *Reliability*. Nach Koch [Koc22] gibt es an, inwieweit ein System nach einem Ausfall wieder den Zustand vor dem Ausfall herstellen sowie Daten erhalten kann [Int11].

Die genaue Anzahl der wichtigsten (Sub-)Attribute war nicht vor dem Termin definiert, sondern wurde ebenfalls von der Gruppe diskutiert (in einem späteren Schritt) und gemeinsam auf sechs festgelegt. Der gesamte Prozess der Umfrage mit anschließender Diskussion, um die Prioritäten der Attribute zu setzen, dauerte etwa 15 Minuten und entsprach somit der geplanten Zeit.

4.1.2. Szenarienerhebung (Schritt 3)

Im nächsten Schritt war die Aufgabe, für jedes dieser (Sub-) QAs zwei Szenarien zu definieren, die das Attribut gut und auf verschiedene Arten beschreiben. Es wurde keine spezielle Methodik angewandt, sondern in einer offenen Diskussion der gesamten Gruppe von verschiedenen Teilnehmern nacheinander für die Attribute Szenarien vorgeschlagen und anschließend überarbeitet oder teilweise auch abgelehnt. Diese Szenarien wurden in Form eines *Utility Trees*, beschrieben in Abschnitt 3.1.1, festgehalten. Nach einer Diskussionszeit von etwa 35 Minuten ergaben sich die Szenarien, die in Abbildung 4.2 abgebildet sind.

Wie aus der Abbildung ersichtlich, wurden für jedes (Sub-)Attribut mit Ausnahme von *Fault Tolerance* und *Recoverability* jeweils zwei Szenarien erstellt. Dies hat zwei Gründe. Zum einen wurden für diese Subattribute keine ausreichend unterschiedlichen Szenarien gefunden. Zum anderen wurde in diesem Fall ein Szenario als ausreichend betrachtet, da Reliability das einzige QA ist, für das mehrere Subattribute eingeschlossen wurden. Darüber hinaus ist es das am geringsten priorisierte Attribut.

Um zusätzlich eine Priorisierung der Szenarien vornehmen zu können, sieht der ARH jeweils eine dreistufige Bewertung der Szenarien hinsichtlich ihrer Wichtigkeit und ihrer technischen Schwierigkeit vor. Diese wurde im nächsten Schritt vorgenommen. Dabei haben die Teilnehmer die Szenarien der Reihe nach betrachtet und die genannten Eigenschaften diskutiert. Nicht immer waren alle anfangs einer Meinung, doch schlussendlich konnte nach etwa 35 Minuten für jedes Szenario Konsens gefunden und dieser Schritt abgeschlossen werden.

Da die Vollendung des dritten Schritts nach Svahnberg und Mårtensson [SM07] länger gedauert hat als geplant, waren am Ende dieses Termins nur noch wenige Minuten übrig. Für alle Sub-Schritte dieses Schritts zusammen waren nur 60 Minuten eingeplant, das Sammeln der Szenarien und das

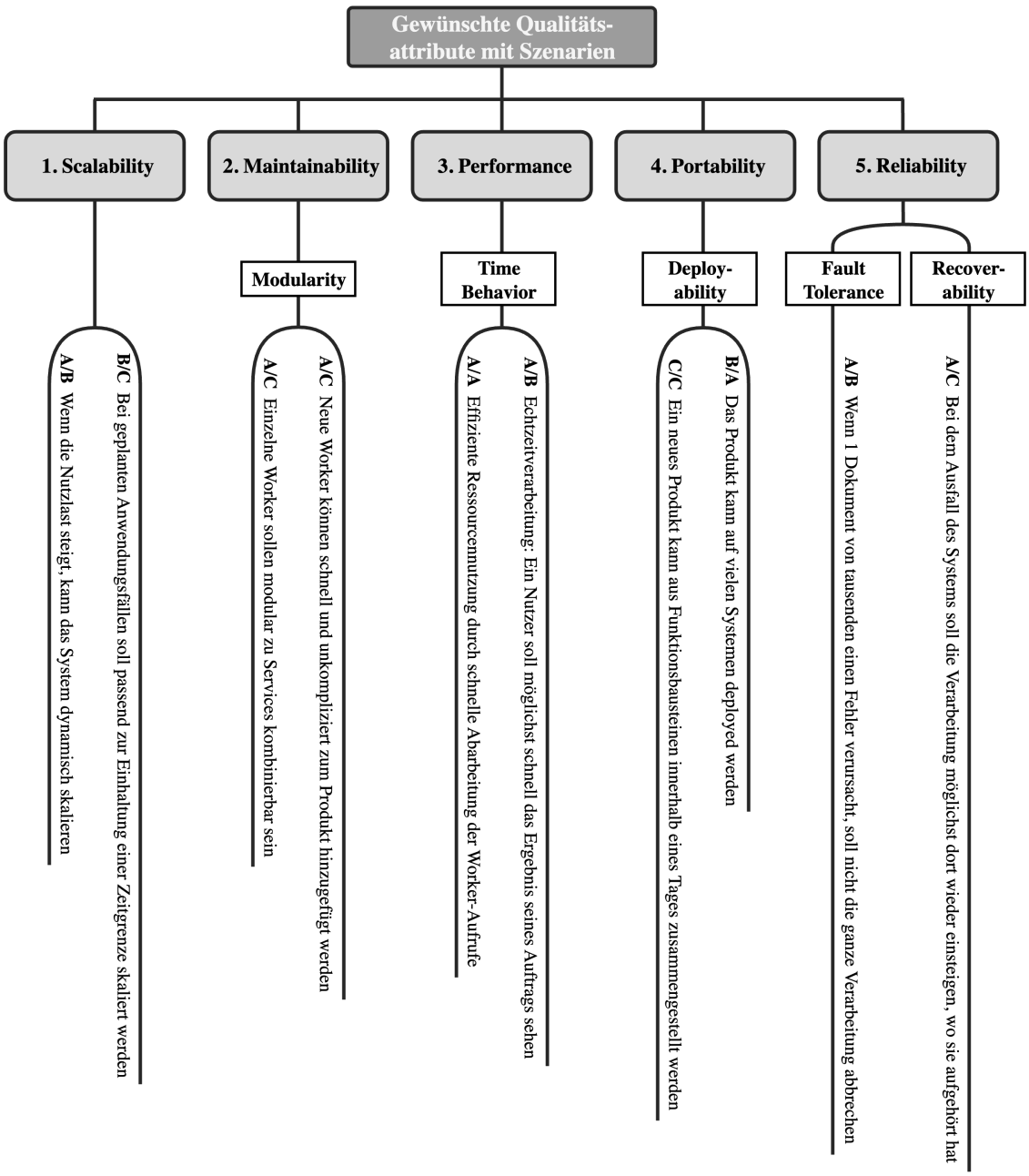

Abbildung 4.2.: Der Utility Tree mit Szenarien, die aus dem in Phase 1 durchgeführten Architekturreview resultieren. Von oben nach unten enthält der Baum folgende Elementarten: [1] Wurzel (ohne Bedeutung), [2] QA, [3] Subattribut, [4] Beurteilung des Szenarios hinsichtlich Wichtigkeit und technischer Schwierigkeit, [5] Szenariobeschreibung.

Bewerten dieser (also zwei von drei Sub-Schritten) nahm jedoch schon 70 Minuten in Anspruch. Daher wurde entschieden, die restlichen Punkte in einer weiteren Sitzung in der nächsten Woche zu bearbeiten.

Am 14. November 2023 wurde dann der zweite Teil des Architekturreviews durchgeführt. Anwesend waren dieselben Personen. Da bei diesem Treffen ein neuer Blickwinkel auf die Thematik möglich gewesen wäre, wurden zunächst zehn Minuten darauf verwendet, die bereits vorhandenen Szenarien und die Bewertung dieser hinsichtlich Wichtigkeit oder technischer Schwierigkeit erneut zu überprüfen und Raum für mögliche Änderungen zu schaffen. Es wurden jedoch keine Änderungen vorgenommen und die Teilnehmer bestätigten lediglich erneut die bereits vorhandenen Szenarien.

Anschließend wurde wie geplant damit fortgefahren, den Szenarien weitere QAs zuzuordnen. Oft können trotz der Erstellung eines Szenarios für ein bestimmtes QA weitere QAs damit assoziiert werden. Deswegen wurde erneut jedes Szenario betrachtet und diskutiert, welche weiteren (Sub-) Attribute darauf zutreffen könnten. Nach etwa 15 Minuten Diskussion waren alle Szenarien behandelt und für jedes einstimmig geklärt, welche weiteren Subattribute damit assoziiert werden. Diese werden folgend im Bezug auf die jeweiligen Szenarien sekundäre QAs genannt, wohingegen ein QA, für das das Szenario erstellt wurde, primäres QA genannt wird. Die resultierenden Assoziationen sind in der Tabelle 4.1 aufgeführt.

Damit wurde der für die Extraktion der Qualitätsanforderungen des Systems relevante Teil des Architekturreviews abgeschlossen. Zusammenfassend kann gesagt werden, dass die Phase größtenteils wie geplant durchgeführt werden konnte. An einigen Stellen sind die verschiedenen Schritte etwas verschmolzen oder es wurde kurz zu vorherigen Schritten zurückgesprungen, was jedoch vollkommen normal ist und auch beispielsweise in ATAM von Kazman et al. [KKC00] als gängige Praxis beschrieben wird. So wurde in diesem Fall beim Erstellen der Szenarien entschieden, bis zu welcher Priorität die QAs mit Szenarien versehen werden, obwohl die Wahl der Anzahl der wichtigsten tatsächlich für den vorherigen Schritt geplant war. Die Beteiligung der Teilnehmer war nicht komplett ausgeglichen, aber jeder hat regelmäßig etwas beigetragen. Es kann angenommen werden, dass das Ergebnis von allen Teilnehmern ausreichend geformt wurde und dass keine einseitige Beeinflussung vorliegt.

4.1.3. Architekturanalyse anhand Szenarien (Schritt 5)

Neben der Erfassung der wichtigsten Szenarien und QAs hatte die Fokusgruppe als sekundäres Ziel, die Wahl der Architektur für *jadice flow* zu bewerten. Diese Bewertung wurde ebenfalls in der zweiten Sitzung des Architekturreviews durchgeführt und entspricht dem vierten Schritt *Assessment* der ersten Phase des ARH. Da dieser Schritt jedoch noch nicht im Tool implementiert ist, wurde er manuell durchgeführt. Im Gegensatz zu vorherigen Schritten der ersten Phase ist dieser nicht relevant für die nächsten Phasen, weshalb es kein Problem ist, den ARH hierbei nicht zu verwenden. In diesem Schritt wurde die Frage diskutiert, ob eine Microservices-Architektur individuell für jedes Szenario vorteilhaft ist im Vergleich zu einer monolithischen Architektur.

Name	Beschreibung	W/S	QAs	MS
Dynamische Skalierbarkeit	Wenn die Nutzlast steigt, kann das System dynamisch skalieren	A/B	Scalability, Resource Utilization, Adaptability, Execution Cost	
Statische Skalierbarkeit	Bei geplanten Anwendungsfällen soll passend zur Einhaltung einer Zeitgrenze skaliert werden	B/C	Scalability, Resource Utilization, Time Behavior	
Jobtemplates	Einzelne Worker sollen modular zu Services kombinierbar sein	A/C	Modularity, Reusability	-
Neue Worker	Neue Worker können schnell und unkompliziert zum Produkt hinzugefügt werden	A/C	Modularity, Reusability	
Schnelle Abarbeitung	Effiziente Ressourcennutzung durch schnelle Abarbeitung der Worker-Aufrufe	A/A	Time Behavior, Resource Utilization	
„Echtzeit“-Verarbeitung	Ein Nutzer soll möglichst schnell das Ergebnis seines Auftrags sehen	A/B	Time Behavior	
Einfaches Deployment	Ein neues Produkt kann aus Funktionsbausteinen innerhalb eines Tages zusammengestellt werden	C/C	Deployability, Modularity, Agility	
Platformunabhängigkeit	Das Produkt kann auf vielen Systemen deployed werden	B/A	Deployability, Installability	
Fehlertoleranz Massenverarbeitung	Wenn 1 Dokument von tausenden einen Fehler verursacht, soll nicht die ganze Verarbeitung abbrechen	A/B	Fault-Tolerance	
Erholen nach Systemausfall	Bei dem Ausfall des Systems soll die Verarbeitung möglichst dort wieder einsteigen, wo sie aufgehört hat	A/C	Recoverability	-

Tabelle 4.1.: Szenarien, die aus dem in Phase 1 durchgeführten Architekturreview resultieren. *W/S* gibt die Wichtigkeit/Schwierigkeit der Szenarien in drei Stufen an (A steht für sehr wichtig und sehr schwierig). *QAs* gibt die Assoziation der Szenarien zu bestimmten Qualitätsattribute (QAs) an; zuerst genannte sind primäre QAs, folgende sekundäre QAs. *MS* gibt die Einschätzung darüber an, ob das jeweilige Szenario von einer Microservices-Architektur profitiert.

Hierbei wurde in drei Stufen unterschieden:

-  : Dieses Szenario profitiert wesentlich mehr von einer MSA als von einer monolithischen Architektur.

-  : Dieses Szenario profitiert wesentlich mehr von einer monolithischen Architektur als von einer MSA.
- - : Dieses Szenario profitiert in verschiedenen Punkten sowohl von einer MSA als auch von einer monolithischen Architektur, wobei sich beide Seiten ungefähr gleichen.

Bei Szenarien wie beispielsweise denen des *Scalability* Attributs war es nicht schwer, Konsens zu finden, da die Möglichkeit der Skalierung auf Service-Ebene einer der größten Vorteile von MSAs gegenüber Monolithen ist. In anderen Fällen jedoch war es schwieriger, abzuwägen, ob die Vorteile einer MSA oder die eines Monolithen überwiegen. Schlussendlich konnte jedoch in Form von offener Diskussion für jedes Szenario einstimmig geklärt werden, welche der drei Optionen gewählt werden sollte, sodass keine Abstimmungen notwendig waren.

Das Ergebnis dieser Einschätzungen ist ebenfalls in Tabelle 4.1 zu sehen. Insgesamt wurde eine MSA in fünf Fällen als vorteilhaft und nur in drei Fällen als unvorteilhaft bewertet. Des Weiteren ist zu beachten, dass die Szenarien in der Tabelle nach der Wichtigkeit der primären QAs sortiert sind und die obersten beiden Szenarien die Hauptfaktoren für die Migration zu einer Microservices-Architektur waren. Durch die Überlegenheit der MSA-Favorisierungen und dass diese verstärkt bei den höchst priorisierten QAs (siehe Tabelle 4.1) vorliegen, kann diese Auswertung die Entscheidung zur Migration zu einer MSA bestätigen.

Nachdem nun erwünschte Szenarien und QAs erhoben wurden und die Wahl einer MSA für *jadice flow* bekräftigt wurde, kann mit Phase 2 fortgefahren werden.

4.2. Phase 2 - Strategieplanung

In dieser Phase wird die Suchfunktion des ARH verwendet, um geeignete Migrationsstrategien zu finden. Diese Suche wird dabei von zwei Faktoren beeinflusst: dem Ergebnis der Phase 1 in Form von QAs beziehungsweise Szenarien und zusätzlichen Filtern, die vor der Suche konfiguriert werden müssen. Im Folgenden werden diese Filter konfiguriert, mehrere Suchen mit ihnen und den QAs aus Phase 1 definiert, durchgeführt und dann aggregiert. Abschließend werden die Ergebnisse dessen analysiert.

4.2.1. Filterselektion

Um möglichst passende Ergebnisse bei der Suche nach Migrationsmethoden zu erhalten, die den Vorstellungen der Architekten und dem Zielsystem entsprechen, bietet der ARH neben der Konfiguration der Szenarien weitere Filtermöglichkeiten. In diesem Abschnitt werden Filter aus einer Auswahl von insgesamt 71 verschiedenen Filtern aus fünf Kategorien und 16 Unterkategorien ausgewählt. Für jede der Eigenschaften aus Tabelle 4.2 kann eine der folgenden drei Präferenzen angegeben werden:

- **Include:** Methoden, die die jeweilige Eigenschaft besitzen, werden höher platziert.
- **Neutral:** Ob Methoden die jeweilige Eigenschaft besitzen oder nicht, wirkt sich nicht auf ihre Platzierung aus.
- **Exclude:** Methoden, die die jeweilige Eigenschaft nicht besitzen, werden höher platziert.

Kategorie	**Subkategorie**	**Filter**
Quality Preferences	System Properties	**Autonomy**, Cohesion, Complexity, Coupling, **Granularity**, Isolation, **Technology Heterogenity**
Input Preferences	Domain Artifacts	Documentation, **Human Expertise**, Ontology, Version Control System
	Runtime Artifacts	Log Traces, User-Application Interactions
	Model artifacts	Activity diagram, Business Process Model, Class diagram, Custom model, Data Flow Diagram, Entity Model, State Machine Diagram, Use Case Model
	Executables	**API / Interface**, Database File, Source Code (Java), **Source Code (No Specification)**, Source Code (Python), Test Cases
Process Preferences	Directions	Bottom-Up, Mixed, Top-Down
	Levels of Automation	Automatic, Manual, Semi-Automatic
	Analysis Types	Dynamic, Historic, Lexical, Static
	Techniques	Clustering, Custom Heuristics, Data-Flow Driven, Domain-Driven Design, Execution-Trace Modeling, General Guidelines, Genetic Algorithm, Graph-based, Machine Learning, Multi-Tenancy, Performance Modeling, Scenario Analysis, Wrapping / Black Box
	Process Strategy	Continuous Evolution, Extension, Greenfield, **Refactor**, Rewrite / Rebuild, Strangler
	Atomar Unit	Business Capability, Class, Entity, Function, Functionality, Interface, Other
Output Preferences	Representation	Guideline / Workflow, **List of services**, Source Code, **Splitting recommendations**, Visualization
Usability Preferences	Validation Methods	Case Study, Experiment, Industry, No Validation
	Accuracy of SIA	High, Medium, Low, Not Available
	Tool Support	No Tool Support
	Tool Types	Database, Decomposition, Dynamic Analysis, Java, Open Source, Other, Reverse Engineering, Static Analysis, Visualization

Tabelle 4.2.: Mögliche Filter des ARH in Phase 2 bei der Suche nach Migrationsstrategien. Die in dieser Thesis verwendeten Filter (mit *include*) sind folgendermaßen markiert: Unterstrichene Filter sind Filter mit geringerer Priorität (Priorität 2, insgesamt sechs Filter). **Unterstrichene und fett markierte Filter** sind Filter mit hoher Priorität (Priorität 1, insgesamt neun Filter).

Die Entwickler des ARH empfehlen, präferierte Eigenschaften auf *include* zu setzen und die anderen auf der Standardeinstellung *neutral* zu belassen und die Einstellung *exclude* nur selten zu verwenden, da der Ausschluss einer Eigenschaft typischerweise nicht das Ziel ist. Wenn eine Eigenschaft nicht erwünscht ist, kann es in vielen Fällen möglich sein, die Methode minimal abzuändern oder einen Teil der Methode zu ignorieren. Dadurch sollte eine solche unerwünschte Eigenschaft trotzdem kein Problem darstellen und die Methode nicht schlechter bewertet werden.

Für die Wahl der Filter für das Refactoring von *jadice flow* wurden alle Filter gemeinsam von Autor und PO betrachtet und dabei evaluiert, welche am besten auf diesen Fall zutreffen (Feldnotizen D.1 und D.2). *Ausgewählt* bedeutet in diesem Kontext, dass der entsprechende Filter auf *include* gesetzt wurde. Die Option *exclude* wurde nicht verwendet. Dabei wurden die ausgewählten Filter in zwei Prioritätsstufen unterteilt (Feldnotiz D.3). Filter mit Priorität 1 sind dabei die höher priorisierten Filter. Diese bilden die realistische Auswahl an Filtern ab, die verwendet worden wären, wenn nur ein Suchdurchgang durchgeführt werden würde. Diese Unterteilung in mehrere Prioritätsstufen ist dazu konzipiert, Suchvorgänge mit verschiedenen Filterkonfigurationen durchführen zu können, um mehrere mögliche Anwendungsfälle zu testen. Eine ausführlichere Beschreibung dazu ist in Abschnitt 4.2.2 gegeben. Die ausgewählten Filter sind in Tabelle 4.2 markiert und in Tabelle 4.3 einzeln aufgelistet.

Kategorie	**Subkategorie**	**Filter Priorität 1**	**Filter Priorität 2**
Quality Preferences	System Properties	Autonomy, Granularity, Technology Heterogenity	Complexity, Isolation
Input Preferences	Domain Artifacts	Human Expertise	Version Control System
	Runtime Artifacts	-	Log Traces
	Executables	API/Interface, Source Code (No specification)	Source Code (Java)
Process Preferences	Process Strategy	Refactor	-
Output Preferences	Representation	List of services, Splitting recommendations	Guideline/ Workflow
Usability Preferences	-	-	-
Summe		**9**	**6**

Tabelle 4.3.: Gewählte Filter des ARH in Phase 2 bei der Suche nach Migrationsstrategien. Unterteilt in zwei Prioritätsstufen, wobei Filter der Priorität 1 in einer Evaluationsrunde vom Autor und PO als wichtiger eingestuft wurden.

Bei der Auswahl der Filter, die vom Autor und PO durchgeführt wurde, haben die folgenden Begründungen zur schlussendlichen Filterwahl geführt: Allgemein wurde darauf geachtet, nur die notwendigsten Filter auszuwählen, um den Einfluss einzelner Filter nicht zu verringern. Als gewünschte Zieleigenschaften des Systems wurden die System Properties (SPs) als besonders wichtig erachtet. Daher wurden insgesamt fünf von sieben verfügbaren Filtern aus dieser Kategorie aufgenommen. Neben den Filtern dieser Kategorie wurden vor allem die Filter der Kategorien *Input Preferences* und *Output Preferences* betrachtet. Diese hängen ebenfalls stark vom System ab und betreffen die Ausgangslage sowie das gewünschte Ergebnis. Deshalb wurden insgesamt sechs Filter

aus der Kategorie *Input Preferences* und vier aus der Kategorie *Output Preferences* gewählt. Andere Kategorien, wie *Process Preferences* und *Usability Preferences*, welche die interne Funktionsweise der Methode betreffen, wurden als unwichtiger und vor allem unabhängiger vom betreffenden System eingestuft. Als einziger Filter in diesen Kategorien wurde bei der Prozess-Strategie die Eigenschaft *Refactoring* gewählt. Da bereits eine MSA vorliegt wären die anderen Optionen wie *Rewrite/Rebuild* in diesem Fall wenig sinnvoll.

4.2.2. Konfiguration der Suchen

Die im vorherigen Abschnitt gewählten und in zwei Prioritäten unterteilten Filter werden nun verwendet, um mehrere Suchdurchgänge zu definieren. Das Ziel des Einsatzes mehrerer Suchen mit jeweils unterschiedlichen Eingaben besteht darin, die Filter- und Suchfunktion anhand der unterschiedlichen Ergebnisse zu evaluieren. Um mehrere Suchen zu definieren, wurde die Konfiguration ausgehend von der realistischen Suchkonfiguration, die folgend *Wichtigste Filter* genannt wird, variiert. Realistisch bedeutet dabei, dass diese Konfiguration verwendet worden wäre, wenn nur eine Suche durchgeführt worden wäre.

Wie in Tabelle 4.4 in der zweiten Spalte dargestellt, beinhaltet diese *wichtigste Suche* alle QAs und die Filter der Priorität 1. Ausgehend davon wurde für die weiteren Suchkonfigurationen in zwei Dimensionen variiert. In der ersten wurden die QAs gleich belassen und die Filter variiert (Spalten 2 und 3 der Tabelle 4.4). In der zweiten Dimension wurden ausgehend von der realistischen Eingabe der Filter (höchst priorisierte Filter) die QAs variiert. Da die Anzahl der Szenarien mit zehn als relativ hoch eingeschätzt wurde und das Erschließen neuer Szenarien ein erneutes Architekturreview benötigt hätte, wurde vom Autor beschlossen, in dieser Dimension lediglich eine Reduktion der Anzahl der Szenarien durchzuführen. Das Entfernen von bis zu 4 Szenarien ergab hierbei keine veränderten Ergebnisse, weswegen in dieser Dimension nur eine zusätzliche Suche komplett ohne QAs aufgeführt wird. Dadurch soll die Zahl der Suchen begrenzt werden.

Suche / **Filter**	**Wichtigste Filter** Realistischste Eingabe	**Alle Filter** Dimension Filter	**Keine Filter** Dimension Filter	**Keine QAs** Dimension QAs
QAs (17)	Inkl.	Inkl.	Inkl.	Exkl.
Filter Priorität 1 (9)	Inkl.	Inkl.	Exkl.	Inkl.
Filter Priorität 2 (15)	Exkl.	Inkl.	Exkl.	Exkl.

Tabelle 4.4.: Konfigurationen der verschiedenen Suchen nach Migrationsverfahren in Phase 2 des MMF. Die Zeilen beschreiben verschiedene Teilmengen der Filter und QAs. In den Spalten sind die Konfigurationen der vier Suchen zu sehen und welche der Teilmengen sie inkludieren (inkl.) oder exkludieren (exkl.). In Klammern hinter den Überschriften ist dargestellt, wie viele Filter die Teilmengen beinhalten.

4.2.3. Selektions- und Aggregationsstrategie

Bei Anwendung der im vorherigen Abschnitt definierten Suchen werden sich vier Listen von Ergebnissen ergeben. Um ein strukturiertes Vorgehen bei der Auswahl der Ergebnisse, die manuell betrachtet werden sollen, zu gewährleisten, werden diese Listen vor Betrachtung zu einer Liste aggregiert. Folgend wird beschrieben, wie diese Aggregation entworfen und durchgeführt wurde.

Um die Auswahl der Ergebnisse, die manuell betrachtet werden sollen, auf eine adäquate Größe einzugrenzen, werden nur die ersten fünf Ergebnisse jeder Suche in die aggregierte Liste übernommen. Außerdem werden alle dem fünften Ergebnis folgenden Ergebnisse übernommen, die die gleiche Anzahl an Übereinstimmungen aufweisen wie das fünfte. Dadurch soll die Anzahl an Übereinstimmung als einziges Kriterium für die Ordnung und Auswahl sichergestellt werden, damit andere Faktoren wie eine mögliche lexikalische Ordnung oder Ordnung nach Identifikationsnummer keinen Einfluss erhalten.

Für eine bessere Übersicht über die selektierten Ergebnisse werden diese anschließend geordnet. Dafür wird die Metrik m_{gesamt} für die Gesamtwertung jedes Verfahrens definiert. Diese gesamte prozentuale Übereinstimmung jedes Verfahrens m_{gesamt} ist ein gewichtetes arithmetisches Mittel der prozentualen Übereinstimmungen des Verfahrens in jeder der vier Suchen. Diese sind $m_{Wichtigste\ Filter}$, $m_{Alle\ Filter}$, $m_{Keine\ Filter}$ und $m_{Keine\ QAs}$, wobei die Namen sich nach Tabelle 4.4 richten und die Werte in der Spalte *Matches, Insgesamt* der Tabelle 4.5 gefunden werden können. Das wären beispielsweise im Fall Habibullah et al. [HLT18] $m_{Wichtigste\ Filter} = \frac{14}{26} \approx 53,8\%$.

m_{gesamt} wird wie folgt berechnet:

$$m_{gesamt} = \frac{m_{Wichtigste\ Filter} + 0,75 \cdot m_{Alle\ Filter} + 0,5 \cdot m_{Keine\ Filter} + 0,25 \cdot m_{Keine\ QAs}}{2,5}$$

Die Gewichtung der verschiedenen Suchen ist so gewählt, dass die realistische Suche das Gewicht 1 hat und die anderen Suchen (mit $0,75$, $0,5$ und $0,25$) weniger stark gewichtet werden. Damit soll die Relevanz der Suche (eingeschätzt durch den Autor) in die Metrik einfließen. Für das reine Ordnen der Ergebnisse ist eine Division durch $2,5$ theoretisch nicht nötig. Um das Ergebnis allerdings eine prozentuale Übereinstimmung nennen zu können, wird durch die Summe der Gewichtungen geteilt.

4.2.4. Suchergebnisbetrachtung

Die Suchen nach Migrationsverfahren wurden wie in Abschnitt 4.2.2 beschrieben durchgeführt und wie in Abschnitt 4.2.3 aggregiert. Daraus resultieren die Suchergebnisse, die in Tabelle 4.5 dargestellt sind. Die Aggregation der Ergebnisse nach der beschriebenen Aggregations-Metrik ist in Tabelle 4.6 zu sehen. Dabei wurde eine Ausnahme im Bezug auf die beschriebene Selektions- und Aggregationsstrategie bei der Suche ohne QAs gemacht. Bei dieser Suche folgten dem dritten Ergebnis zwölf weitere Ergebnisse mit derselben Punktzahl. Um die Zahl der Ergebnisse nicht unnötig zu vergrößern, wurde hierbei eine Ausnahme gemacht und nur die ersten beiden Ergebnisse übernommen. Da die Suche in dieser Form experimentell ist und durch das Fehlen von QAs nur geringe Relevanz hat, wird darin keine Gefahr für die Validität der Ergebnisse gesehen.

In der Reihenfolge werden die einzelnen Methoden im Folgenden kurz zusammengefasst und anschließend ihre Anwendbarkeit verglichen.

Publikation	Matches		
	Insgesamt	QA	SP
Wichtigste Filter (9)			
Habibullah et al. [HLT18]	14/26	11/17	0/3
Daoud et al. [DEF+21]	13/26	7/17	1/3
Vera-Rivera et al. [VPAG21]	11/26	7/17	1/3
De Alwis et al. [DBFP21]	10/26	6/17	1/3
Freitas et al. [FFC21]	10/26	7/17	0/3
Matias et al. [MCF+20]	10/26	7/17	0/3
Assunção et al. [ACC+22]	10/26	5/17	1/3
Alle Filter (15)			
Habibullah et al. [HLT18]	15/32	11/17	0/5
Daoud et al. [DEF+21]	13/32	7/17	1/5
Vera-Rivera et al. [VPAG21]	12/32	7/17	2/5
Freitas et al. [FFC21]	11/32	7/17	0/5
Matias et al. [MCF+20]	11/32	7/17	0/5
Keine Filter (Nur QAs)			
Habibullah et al. [HLT18]	11/17	11/17	-
Vera-Rivera et al. [VPAG21]	7/17	7/17	-
Daoud et al. [DEF+21]	7/17	7/17	-
Freitas et al. [FFC21]	7/17	7/17	-
Matias et al. [MCF+20]	7/17	7/17	-
Keine QAs			
Selmadji et al. [SSB+20]	8/9	-	3/3
Daoud et al. [DEF+21]	6/9	-	1/3

Tabelle 4.5.: Ergebnisse der Suche nach Migrationsverfahren mit ARH. Es wurden verschiedene Filter verwendet, wie in Abschnitt 4.2.1 näher beschrieben. Es ist nach Matches sortiert. Insgesamte Matches, SP Matches und QA Matches entsprechen den im ARH angezeigten Matches.

Ergebnis 1: „An approach to evolving legacy enterprise system to microservice-based architecture through feature-driven evolution rules“ [HLT18]

In dieser Arbeit stellen Habibullah et al. ein konzeptuelles Evolutions-Framework vor, das mithilfe von Transformationsregeln die Migration von Legacy-Systemen zu Microservices unterstützen soll. Das Vorgehen ist in drei Teilabschnitte unterteilt.

Funktionsweise: Das *Core System* besteht aus dem Legacy System und einer Auswahl von Transformationsregeln. Diese können aus der Sammlung der vorgestellten Regeln der Autoren stammen, aber auch durch eigene ergänzt werden. Daraus wird ein *Middle Layer* erstellt, indem

Ergebnis	Matches Prozent (m_{gesamt})
Habibullah et al. [HLT18]	52%
Daoud et al. [DEF+21]	47%
Vera-Rivera et al. [VPAG21]	41%
Freitas et al. [FFC21]	37%
Matias et al. [MCF+20]	37%
De Alwis et al. [DBFP21]	36%
Assunção et al. [ACC+22]	36%
Selmadji et al. [SSB+20]	32%

Tabelle 4.6.: Prozentuale Übereinstimmung der Suchergebnisse in Phase 2 des MMF. Die Metrik m_{gesamt} ist in Abschnitt 4.2.3 beschrieben.

diese Transformationsregeln auf das Legacy-System angewendet werden. Das Resultat dessen ist eine Dekomposition des Systems in Microservices. Das *Target System* kann dann entwickelt werden, indem die erhaltenen Services in einer hybriden Cloud, bestehend aus einer privaten und einer öffentlichen Cloud, umgesetzt werden.

Evaluation: Diese Vorgehensweise wurde auch in einer Fallstudie überprüft. Bei dieser wurde ein öffentliches GitHub Projekt als Legacy System verwendet, welches dann mit drei der vorgestellten Transformationsregeln zu einer MSA migriert wurde. Durch die Architekturänderung konnte das *Coupling* reduziert, sowie die Modularität, Skalierbarkeit und Anfragezeit verbessert werden. Die Autoren weisen jedoch darauf hin, dass diese Ergebnisse nur bedingt aussagekräftig sind und noch an mittleren bis großen Industriesystemen reproduziert werden müssen.

Ergebnis 2: „A multi-model based microservices identification approach“ [DEF+21]

In diesem Artikel stellen die Autoren eine Methode vor, die sie anhand einer Fallstudie mit Barcelonas Fahrradvermietungssystem *Bicing* erklären und validieren.

Funktionsweise: Die Methode besteht aus drei übergeordneten Schritten. Im ersten Schritt werden die Eingaben für die Methode in Form von Business Processes (BPs) gesammelt und logisch verbunden, beispielsweise in Form eines Business Process Model and Notation (BPMN)-Diagramms. Für die Bestandteile dieser Eingabe (genannt Aktivitäten) werden im zweiten Schritt Abhängigkeiten in drei verschiedenen Formen analysiert: *control dependencies*, *data dependencies* und *semantic dependencies*. Eine *control dependency* beschreibt eine Abhängigkeit zweier Aktivitäten im Kontrollfluss, also dass eine der Aktivitäten mit hoher Wahrscheinlichkeit auf die andere folgt. Eine *data dependency* bildet einen Datenfluss zwischen den Ein- oder Ausgaben zweier Aktivitäten ab. Eine *semantic dependency* gibt die Verwandtschaft des Zwecks zweier Aktivitäten an. Die Messung solcher Abhängigkeiten kann anhand der Ähnlichkeit der Namen der Aktivitäten erfolgen. Schließlich wird die *Machine Learning*-Technik *Collaborative Clustering* verwendet, um die Menge von Aktivitäten in Gruppen (genannt *Cluster*) zu unterteilen. Die Autoren haben dazu den Hierarchical Agglomerative Algorithm (HAC) [ML14] erweitert.

Evaluation: Die Methode wurde mit bereits erwähntem System *Bicing* und einer weiteren Anwendung getestet. Die Ergebnisse konnten sich gegen andere Verfahren durchsetzen.

Ergebnis 3: „Microservices Backlog–A Genetic Programming Technique for Identification and Evaluation of Microservices From User Stories“ [VPAG21]

Vera-Rivera et al. beschreiben die Methode Microservice Backlog (MB), ein semiautomatisches Modell zur Bewertung der Granularität einer MSA. Im Gegensatz zu den meisten anderen im Rahmen dieser Thesis betrachteten Methoden liegt hier der Kontext eines Refactorings statt einer Migration vor. Es wird als Ausgangsarchitektur eine MSA erwartet.

Funktionsweise: Als Eingabe werden *User Stories* aus dem *Product Backlog* oder der Release-Planung verwendet. Außerdem kann MB die Metriken *Coupling*, *Kohäsion*, *Granularität*, *semantische Ähnlichkeit* und *Komplexität* für das Gesamtsystem und für einzelne Microservices berechnen. Diese können ebenfalls visualisiert und so übersichtlich betrachtet werden. Anhand dieser Metriken könnte eine manuelle Überarbeitung der Serviceaufteilung angestoßen werden. Wie bei den meisten anderen Verfahren kann außerdem ein Algorithmus genutzt werden, um eine Verbesserung der Metriken zu erreichen. Hierfür wird im Artikel ein genetischer Algorithmus vorgestellt.

Evaluation: Diese Methode wurde in drei Fallstudien überprüft, bei einer davon handelt es sich um eine industrielle Anwendung.

Ergebnis 4: „Refactoring Java Monoliths into Executable Microservice-Based Applications“ [FFC21]

In diesem Artikel beschreiben Freitas et al. eine automatisierte Methode mit zugehörigem Tool *MicroRefact*, mit dem Java-Monolithen in funktional äquivalente Applikationen bestehend aus Microservices umgewandelt werden können.

Funktionsweise: Als Eingabe der Methode fungieren der Quellcode des Monolithen sowie ein Vorschlag für die Microservices-Aufteilung. Dieser wird in Form von Klassen pro Microservice angegeben, wobei jede Klasse nur einem Microservice angehören kann. In der ersten Phase des Tools, der *Information Extraction*, nutzt es den Quellcode, um strukturelle Informationen zu gewinnen und Abhängigkeiten zwischen den gegebenen Microservices zu ermitteln. Diese Abhängigkeiten werden in der zweiten Phase (*Database Refactoring*) dann verwendet, um die Entitäten des Systems und die Beziehungen zwischen diesen zu identifizieren und ein Refactoring der Beziehungen anzustoßen. In der letzten Phase werden die strukturellen Informationen und Abhängigkeiten zwischen den Microservices verwendet, um Abhängigkeiten zwischen den Klassen zu analysieren. Im Zuge dessen werden die Klassen überarbeitet, die abhängig von den Klassen anderer Microservices sind. Dabei werden für Kommunikation zwischen verschiedenen Microservices lokale Funktionsaufrufe durch Application Programming Interfaces (APIs) und Aufrufe auf diese ersetzt.

Evaluation: Um diese Methode beziehungsweise das Tool zu validieren, wurde ein Experiment mit zehn passenden, zufällig ausgewählten GitHub-Projekten durchgeführt. Die Verwendung von *MicroRefact* zur Umstrukturierung dieser Projekte zu Microservices-Applikationen führte zu einer Erfolgsquote von 80%. Es wurde jedoch nur die technische Durchführbarkeit bestätigt und nicht überprüft, ob die neue Architektur in irgendeiner Weise vorteilhaft für die Systeme ist.

Ergebnis 5: „Determining Microservice Boundaries: A Case Study Using Static and Dynamic Software Analysis“ [MCF+20]

In diesem Artikel beschreiben Matias et al. eine systematische Methode zur Dekomposition von Monolithen in Microservices mit einer eingeschränkten Umsetzung im zugehörigen Tool *MonoBreaker*.

Funktionsweise: Anfangs wird durch eine statische und dynamische Analyse des Systems ein Graph der Komponenten des Systems aufgebaut. Die gewichteten Kanten bilden die Stärke der Abhängigkeiten zwischen den Komponenten ab. Dieser Graph kann dann durch einen Clustering-Algorithmus gruppiert werden, um stark abhängige Komponenten zu einem Microservice zusammenzufassen. Dabei sollten sie für geringes *Coupling* möglichst schwach mit anderen Komponenten verbunden sein. Es wird nicht festgelegt, welcher Algorithmus dafür verwendet werden soll.

Evaluation: In einer anschließenden industriellen Fallstudie wurde *MonoBreaker* an einer Webapplikation getestet. Die Entwickler dieser Anwendung wurden zu der vorgeschlagenen Dekomposition befragt. Das Ergebnis des Fragebogens war eine positive Bewertung der erhaltenen Dekomposition sowie eine bessere Bewertung der Nutzung von dynamischer und statischer Analyse statt nur statischer.

Ergebnis 6: „Microservice Remodularisation of Monolithic Enterprise Systems for Embedding in Industrial IoT Networks“ [DBFP21]

De Alwis et al. beschreiben eine Methode zur Erkennung von Microservices in großen, monolithischen Systemen, die auf Industrial Internet of Things (IIoT) Knoten laufen können. Dabei wird sowohl semantische Kenntnis des Systems sowie syntaktisches Wissen über den Code verwendet.

Funktionsweise: Im ersten Schritt werden durch die Analyse von Structured Query Language (SQL)-Queries und Datenbankschemas Business Objects (BOs) abgeleitet. Im zweiten Schritt werden semantische und strukturelle Beziehungen durch Analyse der BO- und Methodenbeziehungen identifiziert. In den folgenden zwei Schritten wird außerdem syntaktische Analyse angewendet, wobei strukturelle Ähnlichkeiten von Methoden sowie Interaktionen zwischen diesen analysiert werden. Im fünften Schritt wird auf Basis der nun bekannten Details über das System ein *k-Means Clustering Algorithmus* angewendet, der letztendlich eine Aufteilung in k Microservices vorschlägt.

Evaluation: Die Methode wurde in einer Fallstudie getestet. Dabei wurde das quelloffene professionelle System *Dolibarr* zu einer MSA umstrukturiert. Es konnten Erfolge in Performance und Effizienz gemessen werden.

Ergebnis 7: „Analysis of a many-objective optimization approach for identifying microservices from legacy systems“ [ACC+22]

Assunção et al. stellen das Verfahren *toMicroservices* vor, das auf *Mulit-Objective*-Optimierung basiert.

Funktionsweise: Als Ziele des Problems werden die Kriterien *Coupling*, *Cohesion*, *Feature Modularization*, *Network Overhead* und *Reuse* definiert. Die Eingabe wird dabei auf Funktionsebene erwartet, beispielsweise durch ein Klassendiagramm. Dazu gehört ebenfalls eine Liste von Features mit Verweisen auf die Stellen, an denen sie implementiert sind. Außerdem muss eine Nummer für die Anzahl an erwarteten Microservices angegeben werden. Mit dem daraus entstehenden Eingabe-Graphen wird durch einen auf dem modernem evolutionären Algorithmus *NSGA-III* [DJ14] basierenden Algorithmus eine Menge an möglichen Lösungen berechnet. Jede dieser besteht aus einer Liste von Microservices, wobei jeder Microservice eine Liste von Funktionen beinhaltet. Es wird hervorgehoben, dass das Ergebnis nicht direkt zur Umgestaltung verwendet werden kann, sondern nur als Vorlage und Startpunkt für eine mögliche Umgestaltung dienen soll.

Evaluation: In der durchgeführten Fallstudie zeigen die Autoren, dass alle mit ihrem Ansatz produzierten Microservices-Kandidaten bessere Ergebnisse liefern als bei der Durchführung mit *Random Search*.

Ergebnis 8: „From Monolithic Architecture Style to Microservice one Based on a Semi-Automatic Approach“ [SSB+20]

Selmadji et al. stellen einen Ansatz vor, mit dem in Objektorientierten Applikationen Microservices identifiziert werden können.

Funktionsweise: Die Besonderheit dieses Ansatzes ist der optionale Einsatz von Expertenwissen. Die Autoren heben hervor, dass anderen Verfahren der Einsatz von Architektenempfehlungen fehlt oder sie diesen voraussetzen. Der vorgestellte Ansatz soll in diesem Aspekt flexibler sein und dieses Expertenwissen nur zusätzlich zu den ebenfalls vorgestellten Qualitätsmetriken verwenden. Die atomare Einheit sind Klassen. Das Verfahren wird zwar semi-automatisch genannt, da Algorithmen für das Clustering von Klassen bei Experteninformationen vorgestellt werden. Eine Implementierung dieser ist allerdings nicht veröffentlicht.

Evaluation: Die Methode wurde durch Anwendung an drei kleinen Applikationen validiert, allerdings hatte die größte dieser Anwendungen nur 13449 Lines of Code (LOC).

Vergleich der Ergebnisse

Nach Betrachtung der acht beschriebenen Ergebnisse werden diese nun bezüglich ihrer Eignung für das weitere Vorgehen verglichen. Um den Vergleich übersichtlicher zu gestalten, werden zu Beginn des Vergleichs direkt zwei Methoden ausgeschlossen, da sie offensichtlich nicht infrage kommen.

Ausschluss: Ergebnis 4 [FFC21] ermöglicht die automatische Umgestaltung eines Java-Monolithen in Microservices unter Angabe einer Dekomposition. Das Tool setzt eine bereits vorgegebene Dekomposition in Microservices als Eingabe voraus und führt lediglich die technische Umwandlung des Monolithen in Microservices durch. Das Ziel dieser Arbeit besteht jedoch darin, eine geeignete Dekomposition mit optimaler Service-Granularität zu finden. Da die Methode dieses Ziel nicht behandelt, kann sie ausgeschlossen werden und wird zur Vereinfachung nachfolgend nicht aufgeführt. Aus ähnlichem Grund wird auch Ergebnis 1 [HLT18] ausgeschlossen. Die Methode schlägt lediglich einen sehr abstrakten und allgemeinen Plan vor, nach dem bei der Migration mit den Transformations-Regelsammlungen vorgegangen werden kann. Diese Regeln sind nur sehr abstrakt

und deswegen wenig hilfreich. Eine der Regeln bedeutet beispielsweise das Folgende: Solange das System ein Legacy-System ist, sollen Microservices erstellt werden, bis die Modularität und Skalierbarkeit maximal sind. Es fehlt also auch hier ein konkreter SIA, weswegen diese Methode ebenfalls ausgeschlossen und nicht weiter betrachtet wird.

Es folgt ein Vergleich der verbleibenden Methoden (Übersicht in Tabelle 4.7). Bevor die Unterschiede der restlichen sechs Verfahren erläutert werden, sollen die Gemeinsamkeiten aller hier gelisteten und allgemein der meisten existierenden Verfahren zur SIA beleuchtet werden.

Methode	**Eingabe**	**Dekompositions-Funktionsweise**	**Vorteile**	**Nachteile**
Ergebnis 3 [VPAG21]	User Stories	Genetischer Algorithmus	Ausgangsarchitektur passend, Visualisierung möglich, hohe Abstraktionsebene ermöglicht Anwendung auf Teilsystem	Tool nicht öffentlich, ohne Visualisierung nutzlos
Ergebnis 2 [DEF+21]	BPs	Clustering	Hohe Abstraktionsebene ermöglicht Anwendung auf Teilsystem	Tool nicht öffentlich
Ergebnis 5 [MCF+20]	Quellcode	Clustering		Tool kann nicht verwendet werden
Ergebnis 7 [ACC+22]	Klassendiagramm oder Quellcode auf Funktionsebene	Evolutionärer Algorithmus	Clustering-Algorithmus ist in Python implementiert	Kein Tool, Extraktion der Funktionen manuell
Ergebnis 8 [SSB+20]	Klassen, Expertenmeinung	Clustering		Kein Tool, Clustering-Algorithmus nur bei Zugabe von Architektenempfehlungen möglich, Anwendung sehr manuell
Ergebnis 6 [DBFP21]	Quellcode, Datenbankschema	Clustering		Datenbank als Eingabe unpassend, Kontext IIoT unpassend

Tabelle 4.7.: Vergleich der Suchergebnisse in Phase 2 des MMF. Diese sind nach der subjektiven Platzierung der Methoden sortiert (Feldnotizen D.7, D.11 und D.14).

Generelle Funktionsweise: Während die Eingaben für die Verfahren sehr unterschiedlich sein können, ist die Funktionsweise zur weiteren Datenverarbeitung oft sehr ähnlich. In der Regel wird aus den Eingaben ein Graph erstellt, der aus Knoten der atomaren Einheiten der Eingabe besteht.

Diese sind mit Kanten verbunden, die die Abhängigkeit verschiedener atomarer Einheiten voneinander abbilden. Anschließend werden mithilfe verschiedener Algorithmen und Strategien in dem Graph Gruppen gebildet, die potenzielle Microservices darstellen. Die Gruppierung erfolgt anhand bestimmter Metriken, wobei Cohesion und Coupling die häufigsten sind. Das Ziel besteht darin, dass Knoten innerhalb einer Gruppe möglichst viele und starke Abhängigkeiten aufweisen und zwischen verschiedenen Gruppen eine möglichst geringe Abhängigkeit besteht. Je nach Verfahren können auch andere Metriken wie Komplexität, semantische Ähnlichkeit oder Granularität berücksichtigt werden.

Eingaben: Der erste und in diesem Fall entscheidendste Aspekt, in dem sich die Methoden unterschieden, ist die Art der Eingabe. Diese bestimmt die atomaren Einheiten in dem Verfahren. Die ausgewählten Verfahren weisen dabei drei verschiedene Ausprägungen auf. Am häufigsten ist die Analyse von Quellcode (Ergebnisse 5, 6, 7 und 8). Dabei sind verschiedene Abstraktionsstufen und Herangehensweisen möglich. Für manche Verfahren werden Funktionen als atomare Einheiten verwendet und für manche Klassen. In einem Ansatz werden außerdem gezielt SQL-Queries betrachtet, wodurch Beziehungen zwischen Datenbank-Objekten analysiert werden. Auch wenn Quellcode als Eingabemöglichkeit eine Mehrheit ausmacht, gibt es unter den sechs ausgewählten Methoden noch zwei, die andere Eingaben verwenden. Dabei handelt es sich um *User Stories* (Ergebnis 3 [VPAG21]) und BPs (Ergebnis 2 [DEF+21]). Im Gegensatz zu den meisten quellcodebasierten Verfahren wird hier vermutlich mehr Subjektivität Einfluss in die Ergebnisse erhalten, da die Eingabe vollständig von Menschen erzeugt werden muss, wohingegen in den anderen Verfahren meist nur der Quellcode und keine manuellen Eingaben verwendet werden. Für den Anwendungsfall in dieser Arbeit wurden diese manuellen Eingabemöglichkeiten trotzdem als besser bewertet, da sie eine höhere Abstraktionsebene ermöglichen und somit der zeitliche Aufwand besser angepasst werden kann. Außerdem ist bei einigen Verfahren, die Quellcode als Eingabe verwenden, die bereits vorhandene MSA von *jadice flow* von Nachteil, wenn diese beispielsweise statische Codeanalysen auf einem Monolithen vorsehen.

Gruppierungsalgorithmus: Nachdem unter Anwendung der Eingabedaten ein Ausgangsgraph erstellt wurde, besteht der nächste Unterschied der Methoden darin, wie aus diesem hochqualitative Microservices identifiziert werden können. Hierbei werden im Allgemeinen hauptsächlich zwei Arten von Algorithmen verwendet: Clustering-Algorithmen (Ergebnisse 2, 5, 6 und 8) sowie genetische beziehungsweise evolutionäre Algorithmen (Ergebnisse 3 und 7). In dieser Fallstudie wurde entschieden, diesen Aspekt der internen Funktionsweise der Verfahren nicht in die Bewertung mit einzubeziehen. Auf der einen Seite ist unklar, inwiefern sich die unterschiedlichen Algorithmen auf das Ergebnis auswirken. Es konnte keine Forschung über den Vergleich der beiden Algorithmus-Arten gefunden werden. Auf der anderen Seite wurden andere Aspekte höher priorisiert.

Weitere Unterschiede: Abgesehen von diesen beiden Hauptkategorien der Unterschiede in der Funktionsweise gibt es noch viele weitere Unterschiede in den Verfahren. Wie bereits erwähnt ist der zeitliche Aspekt in diesem Fall eines der entscheidendsten Kriterien für die Auswahl. Dafür wäre es sehr förderlich, ein Verfahren mit Toolunterstützung zu verwenden, da manueller Aufwand tendenziell höher ist. Leider kann bei keinem der Verfahren Toolunterstützung verwendet werden. Einige Verfahren erwähnen zwar die Entwicklung und Verwendung eines Tools. Doch entweder konnte das Tool nicht öffentlich gefunden werden (Ergebnisse 2 und 3), das Tool ist nicht mit den Technologien von *jadice flow* kompatibel (5) oder es existiert kein Tool (Ergebnisse 7 und 8).

Als Resultat des Vergleichs wurden zwei Methoden als Favoriten identifiziert. Diese sollen in den weiteren Phasen genauer untersucht werden. Am besten geeignet erscheint Ergebnis 3 [VPAG21]. Es hat den Vorteil, dass der erwartete Ausgangspunkt im Gegensatz zum Standard eines Monolithen eine bereits vorhandene MSA ist. Im Vergleich zu anderen Methoden ermöglicht dieses Verfahren eine direktere Behandlung der Ausnahme, ohne Umwege in der Methodik. Zudem bietet es vergleichsweise viele Metriken sowie eine Visualisierung dieser an. Allerdings ist das Verfahren durch diese Features relativ komplex. Ohne das erwähnte Tool MB wird eine manuelle Durchführung des Verfahrens vom Autor und PO als vermutlich zu zeitaufwendig eingeschätzt.

Neben Ergebnis 3 [VPAG21] ist Ergebnis 2 [DEF+21] das einzige andere Verfahren, bei dem durch die manuelle Eingabe ein regulierbarer Aufwand vermutet wird. Diese beiden Verfahren werden als Favoriten für die Anwendung in der nächsten Phase verwendet.

4.3. Phase 3a - Architekturplanung

In dieser Phase des Microservices Migration Framework (MMF) wird das in der vorherigen Phase gewählte Migrationsverfahren angewendet. Außerdem wird ähnlich wie in Phase 2 eine Suchfunktion des ARH verwendet, um geeignete Best Practices und Patterns zu finden, die möglicherweise in die neue Architektur eingebaut werden.

4.3.1. Anwendung Migrationsverfahren

In diesem Abschnitt wird, anknüpfend an die Ordnung der Verfahren nach ihrer Eignung in Phase 2, die Planung des weiteren Vorgehens beschrieben. Dabei ist zu beachten, dass es in dieser Arbeit nicht darum geht, eines der Verfahren originalgetreu umzusetzen und speziell zu testen, sondern ein für den Einzelfall geeignetes Verfahren zu finden. Dies kann durch die Modifikation eines Verfahrens, aber auch durch die Kombination mehrerer Verfahren oder den zusätzlichen Einsatz eines Tools geschehen. Aufgrund des zeitlich begrenzten Rahmens des Refactoring-Prozesses in dieser Thesis ist eines der wichtigsten Kriterien für die Planung des SIA der zeitliche Aufwand der Methode.

Favorit 1: Begonnen wurde mit der Suche nach dem Tool MB des Ergebnisses 3 (Feldnotiz D.15). Dieses konnte weder im Artikel noch auf anderen offiziellen Wegen gefunden werden. Bei der Suche auf GitHub wurde ein Projekt von einem der Autoren mit passendem Namen gefunden[1]. Dieses konnte jedoch auch nach der Behebung mehrerer Startfehler nicht verwendet werden, da bei jeder Navigation von der Startseite des webbasierten Projekts Datenbankfehler auftraten. Da keine Dokumentation vorliegt, der Code teilweise in brasilianischer Sprache geschrieben ist und die Autoren des Artikels auf Anfrage per E-Mail nicht reagierten, scheidet die Verwendung von MB aus. Wie bereits im vorherigen Abschnitt erwähnt, war MB zwar der Favorit, wurde aber ohne funktionierendes Tool als zu aufwendig für diese Thesis erachtet.

[1] https://github.com/ivanmviveros/microservicesbacklog

Favorit 2: Als nächstbeste Option wurde folgend die mögliche Verwendung von Ergebnis 2 [DEF+21] untersucht (Feldnotizen D.16 bis D.18). Auch hier wurde eine Implementierung des Verfahrens in einem Prototyp beschrieben. Diesem wurde kein Name gegeben, die Suche danach blieb erfolglos. Auch eine Anfrage per E-Mail blieb ohne Antwort. Dieses Verfahren wurde im Vergleich zu Ergebnis 3 [VPAG21] als leichter manuell umsetzbar eingeschätzt. Da außerdem keine anwendbaren Methoden mit Toolunterstützung gefunden wurden, wurde folgend eine manuelle Umsetzung dieses Verfahrens begonnen.

Als hauptsächliches Forschungsergebnis dieses Verfahrens wurde im Artikel der entworfene Algorithmus *cHAC* genannt, weshalb mit dessen Implementierung begonnen wurde. Es wurde versucht, diesen mit Top-Down Vorgehen in Java zu umzusetzen. Für einige Details des Pseudocodes wurden Java-spezifische Konstrukte anstelle der im Algorithmus spezifizierten verwendet. Beispielsweise wurde die Synchronisationsvariable *@SIC* des Algorithmus durch eine CyclicBarrier[2] implementiert. Die abstrakte Funktionsweise des Algorithmus konnte erfolgreich umgesetzt werden (Feldnotiz D.16).

Die Implementierung einiger atomarer Bestandteile des Algorithmus war jedoch nicht erfolgreich (Feldnotiz D.17). Die wenig detaillierte Dokumentation und Beschreibung der Formeln und Funktionen des Algorithmus machten eine Umsetzung des Algorithmus in der zur Verfügung stehenden Zeit letztlich unmöglich. An einigen Stellen wurden Funktionen des Algorithmus nicht definiert, sondern nur oberflächlich das Ziel der Funktion beschrieben und die Umsetzung offengelassen. An anderen Stellen wurden im Algorithmus erwähnte Funktionen im Text durch Formeln beschrieben. Hierbei werden Unklarheiten wie Schreib- und Logikfehler im Algorithmus und in mindestens einer Formel vermutet. Diese konnten mit der sehr kurzen Beschreibung im Artikel nicht geklärt werden. Auch andere Details des Algorithmus waren schwer verständlich und oft nicht bis ins Detail erklärt. Aufgrund der Vielzahl dieser Probleme musste die Arbeit an der Implementierung des Algorithmus und damit auch die Implementierung des Verfahrens eingestellt werden.

Tools: Zusätzlich zu den untersuchten Verfahren wurden die vom ARH gelisteten Tools betrachtet, vor allem aufgrund des Mangels von Toolunterstützung bei den ausgewählten Verfahren. Dabei wurden insbesondere die fünf Tools, die in der Arbeit von Fritzsch et al. [FCBW23] beschrieben wurden, näher untersucht. Letztendlich konnte allerdings jedes der Tools aus verschiedenen Gründen nicht verwendet werden. Eines ist durch die Eingabeanforderungen explizit nicht kompatibel, die meisten anderen wären nur mit der monolithischen, älteren Version von *jadice flow* kompatibel gewesen. Außerdem werden viele der Projekte nicht weiterentwickelt oder instandgehalten, wodurch sich die meisten nicht einfach starten ließen. Mit größerem zeitlichen Rahmen wäre bei einigen vermutlich eine Anwendung möglich gewesen, doch in diesem Fall haben gegen jedes der Tools zu viele Punkte gesprochen, als dass der nötige Aufwand in einem sinnvollen Verhältnis zu der verfügbaren Zeit gestanden hätte.

Abschluss der Anwendung: Da die Anwendung der zwei favorisierten Verfahren als (im Zeitrahmen) nicht erfolgversprechend bewertet wurde und auch die gezielte Suche nach Tools für die Anwendung an *jadice flow* keine passende Lösung bieten konnte, wird das Refactoring von *jadice flow* im Rahmen dieser Thesis abgebrochen. Folgend wird in der Auswertung dieser Thesis nicht die durchgeführte Anwendung eines Migrationsverfahrens, sondern die hypothetische Durchführbarkeit der Verfahren bewertet.

[2] https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CyclicBarrier.html

4.3.2. Suche nach Best Practices und Patterns

Die Suche nach Best Practices und Patterns wird wie auch die Suche nach Migrationsverfahren durch die definierten QAs beeinflusst. Das bedeutet, dass alle nötigen Eingaben durch die Durchführung von Phase 1 bereits vorliegen und die Suche ohne weitere Vorbereitungen durchgeführt werden kann. Insgesamt stehen 51 Patterns und 43 Best Practices zur Auswahl. Im Folgenden werden jeweils ersten 20 Suchergebnisse aufgelistet.

Pattern	**Matches**
Service Registry	11/17
CQRS	10/17
Pipes and Filters	10/17
Asynchronous Messaging	9/17
Gateway Offloading	9/17
Load Balancer	9/17
Ambassador	7/17
Backend for Frontend	7/17
Container	7/17
Database is the Service	7/17
Scalable Store	7/17
API Gateway	6/17
Circuit Breaker	6/17
Deploy clusters and orchestrate containers	6/17
Edge Server	6/17
External Load Balancer	6/17
Gateway Aggregation	6/17
Internal Load Balancer	6/17
Local Database Proxy	6/17
Local Sharding-based Router	6/17

Best Practices	**Matches**
Isolated State	11/17
Dynamic Scheduling	9/17
Infrastructure Abstraction	8/17
Use Infrastructure as Code	8/17
Acyclic Calls	7/17
Automated Infrastructure	6/17
Automated Monitoring	6/17
Operation Outsourcing	6/17
Cloud Vendor Abstraction	5/17
Immutable Artifacts	5/17
Standardized Deployment Unit	5/17
Built-in Autoscaling	4/17
Persistent Communication	4/17
API-based Communication	3/17
Automated Restarts	3/17
Autonomous Fault Handling	3/17
Coarse-Grained Microservices	3/17
Communication Indirection	3/17
Guarded Ingress	3/17
Loose Coupling	3/17

Tabelle 4.8.: Ergebnisse der Suche mit ARH nach Best Practices und Patterns, nach Matches sortiert. Die Matches entsprechen den vom ARH ausgegebenen.

Bei erster Betrachtung dieser Ergebnislisten konnten der Autor und der PO von *jadice flow* in einer vorläufigen Bewertung feststellen, dass die einzelnen Best Practices und Patterns für *jadice flow* wertvolle Ergebnisse enthalten. Detaillierter wurde die Qualität dieser Funktion in den Experteninterviews untersucht. Die darin enthaltene Evaluation der Ordnung der Ergebnisse in Bezug auf die definierten QAs sowie das Vorkommen von für die Forschungsfrage zentralen Ergebnissen wird in Kapitel 5 beschrieben.

5. Auswertung

Die Auswertung dieser Fallstudie erfolgt mithilfe zwei verschiedener wissenschaftlicher Methoden. Eine detaillierte Erläuterung der Methodik ist in Kapitel 3 gegeben. Zuerst werden in Abschnitt 5.1 die durchgeführten Experteninterviews ausgewertet. Außerdem werden in Abschnitt 5.2 die im Verlauf der Anwendung des MMF in dieser Thesis erstellten Feldnotizen analysiert. Zum Abschluss dieses Kapitels werden die Ergebnisse dessen in Abschnitt 5.3 diskutiert und durch eine qualitative Bewertung des Refactoring-Prozesses durch den Autor ergänzt.

5.1. Experteninterviews

In diesem Abschnitt werden die Ergebnisse der Experteninterviews beschrieben und diskutiert. Die Experteninterviews wurden mit dem Ziel durchgeführt, eine Evaluation der Anwendung des MMF/ARH auf *jadice flow* von weiteren Personen zu erhalten, die nicht an der Durchführung beteiligt waren. Die genaue Methodik der Interviews ist in Abschnitt 3.2.1 erläutert und der Leitfaden kann in Anhang A gefunden werden.

Das erste Interview wurde gezielt mit dem PO von *jadice flow* durchgeführt. Er begleitete alle Phasen der Thesis und benötigte deshalb keine thematische Einweisung. Der PO hat die Präambel und das Informationsmaterial erst wenige Stunden vor dem Interview erhalten. Durch die vorherige Mitarbeit des Teilnehmers an der Thesis konnte im ersten Interview zusätzlich evaluiert werden, ob die Verständnisfragen, inhaltlichen Fragen und der Zeitrahmen passend gewählt wurden. Den anderen Teilnehmern, die nicht mit dem Inhalt der Thesis vertraut waren, wurden vier Tage vor den Interviews eine Präambel (Anhang B) und fachliches Informationsmaterial (Anhang C) zur Verfügung gestellt. Die Interviews konnten ohne größere Probleme oder Änderungen im Vergleich zur Planung durchgeführt werden. Für jedes Interview wurde eine Stunde Zeit eingeplant, obwohl der Zeitaufwand auf 45 Minuten geschätzt worden war. Die durchgeführten Interviews dauerten im Durchschnitt 50 Minuten. Dadurch entstand kein Zeitdruck und die Teilnehmer mussten zu keinem Zeitpunkt in ihren Antworten unterbrochen werden.

Die Antworten der Teilnehmer wurden in Abbildung 5.1 hinsichtlich der Nützlichkeit des ARH kodiert (Kodierleitfaden in Abbildung 3.4). Eine Übersicht über die drei interviewten Teilnehmer und deren Erfahrung ist in Tabelle 5.1 zu finden. In den folgenden Abschnitten werden die Ergebnisse dessen sowie explizierte und aggregierte Zusammenfassungen der Aussagen der Teilnehmer dargelegt und diskutiert.

Fragenabschnitt	**Frage**	**Erhaltene Kodierungen** Interview 1	2	3
Evaluation des MMF/ARH allgemein	Welche Aufgaben soll ein MMF übernehmen?	3		2
	Gewünschte Qualitäten dabei	3		
	Funktionsweise und Phasen des MMF/ARH sinnvoll?	3	3	3
	Probleme beim Framework	2	2	2
Evaluation der gewählten Migrationsmethoden für *jadice flow*	Allgemeine Bewertung	3	2	2
	Bewertung auf abstrakter Ebene	3	3	2
	Probleme auf weniger abstrakter Ebene	3	2	1
	Verbesserung der Granularität möglich?	3	2	2
Evaluation der Best Practices und Patterns für *jadice flow*	Sortierung passend?	2	2	2
	Anzahl bereits angewendet	2	3	3
	Anzahl potentiell sinnvoll (zusätzlich zu angewendeten)	2	2	3
	Enthält Lösungen für Optimierung des Kommunikationsmodells/ Netzwerk-Overheads	2	2	2
Evaluation der Anwendung des MMF/ ARH auf *jadice flow*: Zusammenfassung	Gewinnbringend für *jadice flow*?	3	2	3
	Gewinnbringend für *jadice server*?	3	3	3

Legende

3	Gute Bewertung des MMF/ARH durch die Frage
2	Eher gute Bewertung des MMF/ARH durch die Frage
1	Eher schlechte Bewertung des MMF/ARH durch die Frage
0	Schlechte Bewertung des MMF/ARH durch die Frage

Abbildung 5.1.: Auswertung der Kodierung der Experteninterviews.

ID	Rolle	Erfahrung in Jahren			An Thesis beteiligt
		Software-entwicklung	Micro-services	*jadice flow/ jadice server*	
PO	Product Owner	20	6	13	Ja
SA	Softwarearchitekt	13	7	2	Nein
SE	Softwareentwickler	8	8	6	Nein

Tabelle 5.1.: Teilnehmer der Experteninterviews. An Thesis beteiligt bedeutet, dass der Teilnehmer an der Wahl der Migrationsverfahren beteiligt war. Der SE war ebenfalls an Phase 1 des MMF beteiligt, das hat aber keine Relevanz für diese Interviews.

5.1.1. Evaluation des MMF/ARH allgemein

Dieser Teil der Interviews diente der allgemeinen Bewertung des Frameworks und des Tools. Die Fragen wurden insbesondere ohne Bezug auf die in dieser Arbeit durchgeführte Migration gestellt. Hierfür wurde ein Zeitrahmen von zehn Minuten vorgesehen, der mit durchschnittlich neun Minuten pro Interview auch eingehalten wurde.

Gewünschte Features eines MMF: In der ersten Frage sollte ohne Bezug zum MMF oder ARH untersucht werden, welche Aufgaben ein Microservices Migration Framework allgemein für Entwickler bei der Migration von Monolithen zu Microservices übernehmen könnte. Das Ziel ist es, die Antworten mit den tatsächlichen Funktionen des ARH abzugleichen und ggf. Ideen für zukünftige Funktionen des ARH zu sammeln. In den Interviews hat sich herausgestellt, dass die Frage schwer verständlich ist, da zwei von drei Teilnehmern oft Konzepte eines Meta-Frameworks wie dem ARH mit konkreten Migrationsverfahren vermischt haben. In einem Interview wurde daher die Frage schlussendlich übersprungen, da auch eine Wiederholung und Vertiefung der Frage keine Antwort erbrachte. In nur einem Interview wurde die Frage hinreichend verstanden und beantwortet, um sinnvoll die damit verknüpfte zweite Frage stellen zu können. Diese wurde in den anderen beiden Fällen schlicht übersprungen.

Zu großen Teilen wurden in den Antworten auf die erste Frage Funktionen genannt, die der ARH anbietet. Dazu gehören das Sammeln von Informationen und die Definition von Anforderungen (umgesetzt in Phase 1) sowie die Suche nach Verfahren, durch die primär die Granularität der Microservices optimiert werden kann (umgesetzt in Phase 2). Als gewünschte Qualität wurde dabei genannt, dass eine große Auswahl an Verfahren bevorzugt wird, von denen dann mehrere manuell verglichen werden können. Weitere Ideen sind die Visualisierung der gesammelten Informationen, ein Teilen der Informationen, um besser im Team arbeiten zu können und Tipps für den Migrationsprozess. Letzteres könnte als teilweise umgesetzt bewertet werden, da das gesamte Framework als Leitfaden für die Migration gesehen werden kann. Generell ist die Aussagekraft dieses Fragenabschnittes dadurch eingeschränkt, dass die Teilnehmer im Vorfeld über die Funktionsweise des ARH informiert wurden und somit diese Fragen nicht mehr völlig unbeeinflusst beantworten konnten.

Funktionsweise des MMF: Anschließend folgen Fragen zur Bewertung der generellen Funktionsweise des MMF in Phasen. Alle Experten haben angegeben, dass sie die Phasenaufteilung als sinnvoll ansehen. Die Durchführung einer Analyse des Systems und der Planung der Ziele in Phase 1 wurde als natürlicher Ausgangspunkt für allgemeine Refactoring-Prozesse wahrgenommen.

Vorschläge von Migrationsverfahren in der zweiten Phase wurden ebenfalls als positiv bewertet. Es wird vermutet, dass so Verfahren gefunden werden können, die sonst nicht betrachtet worden wären. Das Zusammenspiel der Phasen wurde ebenfalls als sinnvoll bewertet. Es würde eine stetige Evaluierung der gewählten Vorgehensweise begünstigt und angestoßen werden.

Probleme des MMF/ARH: Auf die Frage nach Problemen am MMF oder ARH wurden einige potentielle Probleme identifiziert. Ein Teilnehmer kritisierte die Dokumentation des ARH. Ein Beispiel dafür sei die Beschreibung der Filteroptionen für die Suche nach Migrationsverfahren. Diese Kritik wurde vom Teilnehmer selbst direkt im Bezug auf das junge Entwicklungsstadium des Tools und die wissenschaftliche Orientierung relativiert. Außerdem wurde von einem Experten hervorgehoben, dass der Haupteinflussfaktor auf die Nützlichkeit des Tools die verfügbare Datenbasis ist. Diese müsse stetig aktualisiert und gut kategorisiert werden. Eine weitere Kritik bezieht sich auf die Einschränkung auf bestimmte QAs in der ersten Phase. Da die verfügbaren QAs auch in der Kategorisierung der Verfahren vorhanden sein müssen, ist das Modell nicht erweiterbar und bildet möglicherweise in bestimmten Anwendungsfällen nicht die nötigen Anforderungen ab. Die Schwere dieser Kritiken wurde sowohl von den Teilnehmern als auch vom Autor als relativ gering eingeschätzt.

5.1.2. Evaluation der Anwendung des MMF/ARH auf *jadice flow*

Während im ersten Teil der Fragen der Fokus auf der allgemeinen Funktionsweise des MMF und ARH lag, wurde in diesem Teil ein Bezug zu der in dieser Arbeit durchgeführten Anwendung des Frameworks auf *jadice flow* hergestellt. Dafür wurden erst Fragen zu zwei ausgewählten Migrationsmethoden und dann zu der Liste von Best Practices und Patterns, die der ARH für *jadice flow* anbietet, gestellt. Als Hauptteil wurde in diesem Abschnitt der Interviews die größte Dauer vermutet, jedoch mit 20 Minuten trotzdem zu gering kalkuliert. In nur einem Interview konnte diese Zeitspanne eingehalten werden, im Durchschnitt benötigte dieser Abschnitt 27 Minuten. Vor allem die Befragung zu den Best Practices und Patterns dauerte mit durchschnittlich 14 Minuten länger als erwartet. Durch das Einhalten des allgemeinen Zeitrahmens stellt das jedoch kein Problem dar.

Evaluation der ausgewählten Migrationsmethoden für *jadice flow*

In diesem Fragenabschnitt wurden den Teilnehmern Fragen zu den ausgewählten Migrationsverfahren gestellt. Diese wurden vor Beginn der ersten inhaltlichen Fragen im Rahmen der Verständnisfragen (siehe Anhang A.3.1) mit den Teilnehmern besprochen, um sicherzustellen, dass das Verständnis der Verfahren für eine Bewertung dieser ausreicht. Im Folgenden werden die Verfahren auf Basis ihrer Reihenfolge im Informationsmaterial in Anhang C referenziert. Verfahren 1 entspricht demnach Ergebnis 2 [DEF+21] und Verfahren 2 Ergebnis 3 [VPAG21]. Als Einstieg dieses Teils wurde eine allgemeine Bewertung der Nützlichkeit der Verfahren im Bezug auf *jadice flow* abgefragt. Oft wurden dadurch die folgenden Fragen zur Bewertung auf abstrakter und auf weniger abstrakter, technischer Ebene bereits teilweise beantwortet.

Abstrakte Funktionsweise: Auf abstrakter Ebene wurden beide Verfahren von allen Teilnehmern als potentiell geeignet bewertet. Vor allem die Art der Eingabe(n) und Ausgaben der Verfahren stand bei den Antworten im Fokus. Ein Experte favorisierte das zweite vor dem ersten Verfahren, die anderen beiden umgekehrt. Als Grund für die Bevorzugung des zweiten Verfahrens wurde die

besser passende Eingabe einer bereits vorhandenen MSA sowie die Visualisierung der Metriken für eine manuelle Überarbeitung der Architektur genannt. Diese wird zusätzlich zur Optimierung der Serviceaufteilung mit einem Algorithmus, die auch das zweite Verfahren beinhaltet, angeboten. Die anderen Experten bewerteten diese Vorteile des ersten Verfahrens als weniger wichtig und gaben an, dass sie die Eingabe in Form von einem BPMN-Diagramm den User Stories vorziehen würden. Das liegt vor allem daran, dass die User Stories eine weniger technische Ebene abbilden und damit schwerer greifbar ist, wie die Verbindung zwischen verschiedenen Einheiten analysiert und die Architektur verbessert werden soll. Sie evaluierten die zweite Methode dennoch als potentiell sinnvoll anwendbar, nur eben als Ergänzung zur ersten Methode.

Technische Ebene: Auf abstrakter Ebenen kann ein Konzept sinnvoll klingen und in der Realität trotzdem nicht funktionieren. Deshalb wurde gefragt, ob die Experten sich konkrete Probleme bei der Umsetzung der Verfahren vorstellen können. Hierbei muss beachtet werden, dass die Antwortmöglichkeiten der Experten bei der Frage sehr beschränkt sind, da sie die Verfahren nicht im Detail kennen. Selbst bei besserer Informationslage über die Verfahren kann es passieren, dass bestimmte Probleme erst in der Praxis auffallen. Die Evaluation dieser Frage soll deshalb nicht als Beweis für das sichere Gelingen der Verfahren in einer Anwendung mit *jadice flow* dienen, sondern nur beleuchten, ob die Verfahren schon bei anfänglicher Betrachtung Probleme aufwerfen, die den Versuch einer Anwendung infrage stellen.

Insgesamt wurden auf diese Frage wenige Probleme von den Experten genannt. Ein Experte sah gar keine Probleme und konnte auf die kritische Nachfrage des Interviewers, der ein potentielles Problem benannte, sinnvoll argumentieren, dass er dieses nicht als Problem sieht. Die anderen Experten nannten einige Probleme, von denen sie aber keines als zu kritisch für eine Anwendung der Verfahren einschätzten. Unter anderem wurde für die Eingabe von BPMN-Diagrammen argumentiert, dass man „nur das bekommt, was man hereingibt“. Damit wird kritisiert, dass manuell erstellte Eingaben nicht objektiv sind und diese Subjektivität das Ergebnis beeinflussen kann. Außerdem merkte ein anderer Experte allgemeiner an, dass nie komplett vollständige Eingabedaten vorliegen und durch fehlende Daten unvorhergesehene Probleme entstehen können. Als Beispiel wurden *Edge Cases* genannt, die womöglich nicht vom Verfahren berücksichtigt werden können, aber einen großen Einfluss auf den Erfolg einer Architektur haben können.

Bezug zur Forschungsfrage: Abschließend zu den Verfahren wurde jeweils die Frage gestellt, ob summa summarum eine Anwendung der Verfahren als potentiell gewinnbringend im Bezug auf die Granularität der Microservices von *jadice flow* erachtet wird. Da die Frage sehr hypothetisch ist, waren sich einige Teilnehmer etwas unsicher, insgesamt wurde aber von zwei Teilnehmern eine positive Tendenz verzeichnet und vom anderen Teilnehmer ein klares „Ja“.

Evaluation der Best Practices und Patterns für *jadice flow*

Als ebenfalls wichtiger Bestandteil des Frameworks wurde in den Interviews auch konkreter zur dritten Phase befragt. Da im Gegensatz zu den Migrationsverfahren die Namen von Best Practices und Patterns in vielen Fällen genug Aussagekraft haben, damit die Teilnehmer das Konzept verstehen, wurde hier keine Vorselektion getroffen, sondern alle Einträge des ARH den Teilnehmern gezeigt. Dabei ist klar, dass den Experten nicht alle Ansätze im Voraus bekannt sind oder durch den Namen klar werden. Die einzige zeitlich sinnvolle Lösung dafür ist, alle nicht verstandenen Best Practices und Patterns zu ignorieren, was den Experten so auch gesagt wurde. Das Ziel dieses

Teils ist dabei die Bewertung der Experten, ob das Tool für Findung und Wahl von passenden Best Practices und Patterns zielführend ist und auch einen Vorteil gegenüber manueller Recherche bieten kann.

Ordnung der Ergebnisse: Im ersten Schritt sollten die Experten die Sortierung der Best Practices und Patterns bewerten. Dabei wurde hervorgehoben, dass diese auf Basis der QAs *Scalability*, *Maintainability*, *Performance* und *Portability* entsteht. Die Sortierung wurde grundlegend von allen Teilnehmern als sinnvoll bewertet. An wenigen Stellen wurden Ergebnisse in einer anderen Priorisierung eingeschätzt, größtenteils wurden dabei aber Einwände wieder relativiert, nachdem erneut über die eingegebenen QAs nachgedacht wurde. Ein Teilnehmer erwähnte auch, dass man mit anderer Erwartungshaltung an diese Liste herangehen sollte als beispielsweise an die Liste der Migrationsverfahren. Das liegt daran, dass sich nicht auf einzelne Best Practices und Patterns festgelegt werden muss, sondern mehrere kombinierbar sind. Es wurde speziell hervorgehoben, dass auch zum Beispiel manche Ergebnisse mit nur einer Übereinstimmung gewinnbringend sein können, wenn sie für ein bestimmtes QA sehr gut sind, aber kein anderes bedienen.

Bereits umgesetzte und potentiell sinnvolle Best Practices und Patterns: Als nächstes wurden die Einschätzungen der Experten zu der Anzahl der bereits in *jadice flow* umgesetzten Best Practices und Patterns sowie zu den noch nicht umgesetzten, aber potentiell sinnvollen, abgefragt. Vom Interviewer wurde explizit angemerkt, dass lediglich eine Einschätzung abgegeben werden soll und Ergebnisse mit unbekannten Namen ignoriert werden sollen. Im Durchschnitt wurde eingeschätzt, dass 13 Patterns und 15 Best Practices bereits in *jadice flow* (teilweise) umgesetzt sind und 5 Patterns sowie 4 Best Practices für die zukünftige Implementierung potentiell sinnvoll wären. Die Abweichung der Antworten war dabei sehr hoch, vermutlich aufgrund der spekulativen Fragestellung. Eine Einschätzung ist schwierig, wenn lediglich die Namen der Ergebnisse betrachtet werden. Ein vorsichtiger Teilnehmer, der keine falschen Ergebnisse miteinschließen will, könnte zu geringeren Zahlen kommen als ein selbstsicherer, optimistischer Teilnehmer. Außerdem wurde die Anweisung, nur eine grobe Einschätzung abzugeben und nicht eine knapp 100 Ergebnisse lange Liste genau zu betrachten, unterschiedlich ernst genommen. Dabei wurde beobachtet, dass der Teilnehmer mit der größten Abweichung deutlich schneller geantwortet hat. Des Weiteren war bei diesen Antworten auffällig, dass in Prozentanteil von der Gesamtergebnisanzahl (beispielsweise „60% der Patterns“) geantwortet wurde statt in absoluten Zahlen. Das könnte auf eine erhöhte Ungenauigkeit der Einschätzung dieses Teilnehmers hindeuten.

Bezug zur Forschungsfrage: Abschließend wurde gefragt, ob eine Verbesserung des Kommunikationsmodells beziehungsweise des Netzwerk-Overheads durch die Best Practices und Patterns der Liste für möglich gehalten wird. Die Resonanz dabei war positiv, aber nicht übermäßig positiv. Es wurden von den Experten jeweils ein bis zwei Möglichkeiten genannt, aber auch betont, dass die Zahl der dafür geeigneten Ergebnisse relativ gering ist und diese Ergebnisse in der Liste nicht hoch priorisiert sind. Ein Experte erwähnte allerdings auch, dass die QAs nicht unbedingt auf dieses Ziel ausgerichtet wurden und dadurch vermutlich diese Sortierung zustande kommt.

Evaluation der Anwendung des MMF/ARH auf *jadice flow* insgesamt

Zum Abschluss der Interviews wurden jeweils zwei Fragen gestellt, die die Ergebnisse aller vorherigen Fragen aggregieren sollten. Es sollte insgesamt eingeschätzt werden, ob die Verwendung des Frameworks und des Tools zum momentanen Zeitpunkt gewinnbringend für *jadice flow* wäre. Dieselbe Frage wurde anschließend im Bezug auf eine Verwendung des ARH in der Vergangenheit mit dem Vorgängerprodukt mit monolithischer Architektur wiederholt.

Zwei der Teilnehmer bewerteten die Verwendung zum jetzigen Zeitpunkt als sinnvoll. Der andere Teilnehmer war unsicherer, tendierte jedoch auch dazu. Alle waren sich darin einig, dass eine Anwendung für die Migration in der Vergangenheit noch wertvoller gewesen wäre. Ein Experte war unsicher, ob die Verwendung des MMF neues Optimierungspotenzial aufzeigt, das noch nicht bekannt ist. Die Liste von Best Practices und Patterns sei zwar eine schöne Übersicht, aber wirklich neue Ansätze hätte er nicht gesehen. Ein anderer Experte hob dagegen hervor, dass das Tool eine gute Übersicht für Architekten bietet und neue Anregungen fördert. Bei diesen Meinungsverschiedenheiten der Experten konnte eine leichte Korrelation zwischen einer höheren Berufserfahrung der Experten und einer besseren Gesamtbewertung des Frameworks festgestellt werden. Die Aussagekraft dieses Ergebnisses sollte aufgrund der geringen Teilnehmerzahl jedoch nicht überbewertet werden.

5.2. Feldnotizen

Während der Anwendung des Frameworks auf *jadice flow* in dieser Arbeit wurden in den Phasen 2 und 3 strukturierte Feldnotizen erstellt, die wichtige Schritte dieser Phasen festhalten. Die Methodik für die Erstellung der Feldnotizen ist in Abschnitt 3.1.2 beschrieben. In diesem Abschnitt wird die Auswertung dieser Feldnotizen mit der in Abschnitt 3.2.2 erläuterten Codierung durchgeführt.

Insgesamt wurden 18 Feldnotizen erstellt, wovon 14 in die zweite Phase und vier in die dritte Phase des MMF fallen. In Phase 1 wurden keine Feldnotizen formuliert. Aufgrund der großen Anzahl von Feldnotizen in der zweiten Phase wurde die Phase für die Auswertung weiter in die übergeordneten Schritte *Filterwahl* mit den Feldnotizen D.1 bis D.3 und *Ergebnisbetrachtung* mit den Feldnotizen D.4 bis D.14 unterteilt. Im Folgenden wird die Auswertung der zugehörigen Feldnotizen für jeden dieser Schritte beschrieben.

5.2.1. Phase 2: Filterwahl

In diesem Schritt wurden die Filter für die Suche nach Migrationsverfahren ausgewählt. Die Entscheidungen dafür wurden ausschließlich in Besprechungen zwischen dem Autor und dem PO von *jadice flow* oder dem universitären Betreuer getroffen. Dabei wurden drei Feldnotizen erstellt, deren Auswertung durch die festgelegte Kodierung in Tabelle 5.2 zu sehen ist.

Wie in der Tabelle erkennbar ist, verlief dieser Schritt sehr positiv. In jeder Feldnotiz wurde eine Sache vermerkt, die gut gelaufen ist und nur in einem Fall ein Problem. Das ist auch an der Häufigkeit von 100 Prozent der Kodierung „Mehr gut gelaufen“ zu sehen.

Code	Feldnotiz 1	2	3	Summe (3)
Gut gelaufen	✓	✓	✓	3 (100%)
Schlecht gelaufen	✓			1 (33%)
Mehr gut gelaufen	✓	✓	✓	3 (100%)
Mehr schlecht gelaufen				0 (0%)
Gutes Gefühl		✓	✓	2 (67%)
Schlechtes Gefühl	✓			1 (33%)
Unsicherheit	✓			1 (33%)
Frustration				0 (0%)
Verbesserungs-vorschlag	✓			1 (33%)
Einzelarbeit				0 (0%)
Besprechung	✓	✓	✓	3 (100%)

Tabelle 5.2.: Auswertung der Kodierung der Feldnotizen in Phase 2 beim übergeordneten Schritt *Filterwahl*. Prozentuale Angaben hinter der Summe beziehen sich auf den Anteil des Auftretens des Codes zu der Anzahl der Feldnotizen.

In den Einträgen der Empfindungen ist zu erkennen, dass trotz des positiven Verlaufs dieses Schritts einmal auch das schlechte Gefühl „Unsicherheit“ festgehalten wurde. Das hängt in diesem Fall jedoch nicht mit dem ARH oder der Migration zusammen, sondern wird auf die Unerfahrenheit des Autors mit der Anwendung der Methodik von Feldnotizen zurückgeführt. Ansonsten wurde in 67 Prozent der Feldnotizen ein positives Gefühl vermerkt. Zudem wurde ein Verbesserungsvorschlag kodiert, der direkt mit der Kodierung „Schlecht gelaufen“ der Feldnotiz D.1 zusammenhängt. Es wird die Beschreibung der Filteroptionen bemängelt. Da der ARH noch in der Entwicklung ist, sind Mängel in der Dokumentation zu erwarten.

5.2.2. Phase 2: Betrachtung der Filterergebnisse

In diesem Schritt wurde mithilfe der im vorherigen Schritt gesammelten Filter eine Suche nach Migrationsverfahren durchgeführt. Für jedes der acht in Einzelarbeit betrachteten Verfahren wurde je eine Feldnotiz erstellt. Außerdem wurden drei Feldnotizen bei Besprechungen über die Bewertung der Verfahren für *jadice flow* erstellt. Die Kodierung dieser ist in Tabelle 5.3 zu sehen.

Die Ergebnisse dieses Schritts fallen gemischt aus. In 73 Prozent der Feldnotizen wurden positive Aspekte notiert, jedoch wurden in fast genauso vielen Feldnotizen (64 Prozent) auch negative Aspekte festgehalten. Durch „Gut gelaufen“ und „Schlecht gelaufen“ wurde in diesem Schritt bei der Ergebnisbetrachtung der Eindruck von dem jeweiligen Verfahren kodiert. Bei nur zwei der Feldnotizen der Ergebnisbetrachtung in Einzelarbeit wurde „Gut gelaufen“ kodiert, ohne dass „Schlecht gelaufen“ kodiert wurde. Diese beiden Verfahren wurden für die Phase 3 ausgewählt. Letztendlich wurde jedoch doppelt so oft „Mehr gut gelaufen“ kodiert wie „Mehr schlecht gelaufen“. Der Schritt kann daher trotz gewisser Schwierigkeiten als größtenteils positiv verlaufen interpretiert

Code	Feldnotiz 4	5	6	7	8	9	10	11	12	13	14	Summe (11)
Gut gelaufen		✓	✓	✓		✓		✓	✓	✓	✓	8 (73%)
Schlecht gelaufen	✓				✓	✓	✓		✓	✓	✓	7 (64%)
Mehr gut gelaufen		✓	✓	✓				✓				4 (36%)
Mehr schlecht gelaufen	✓				✓							2 (18%)
Gutes Gefühl		✓	✓									2 (18%)
Schlechtes Gefühl	✓										✓	2 (18%)
Unsicherheit											✓	1 (09%)
Frustration	✓											1 (09%)
Verbesserungs-vorschlag	✓											1 (09%)
Einzelarbeit	✓	✓	✓		✓	✓	✓		✓	✓		8 (73%)
Besprechung				✓				✓			✓	3 (27%)

Tabelle 5.3.: Auswertung der Kodierung der Feldnotizen in Phase 2 beim übergeordneten Schritt *Ergebnisbetrachtung*. Prozentuale Angaben hinter der Summe beziehen sich auf den Anteil des Auftretens des Codes zu der Anzahl der Feldnotizen.

werden. In den Besprechungen wurde früh festgestellt, dass die Ergebnisse Optionen enthalten, deren Anwendung potentiell möglich ist. Das spiegelt sich auch darin wider, dass in 67 Prozent der Besprechungen mehr gut gelaufen ist und nur in einem Fall gleich viel gut wie schlecht gelaufen ist. Die große Anzahl der Kodierungen von „Schlecht gelaufen“ hängt größtenteils mit Verfahren zusammen, die sich weniger gut für die Anwendung an *jadice flow* eignen. Jedoch ist auch von anderen Suchmaschinen bekannt, dass nicht alle Suchergebnisse zur Vorstellung des Suchenden passen. In der Praxis wird sich jedoch vermutlich zeigen, dass es genügt, ein bis zwei passende Verfahren mit ARH zu finden zu und diese zu verwenden. Eine größere Auswahl an passenden Verfahren wäre dennoch wünschenswert. Allerdings ist es aufgrund des frühen Stadiums des ARH wahrscheinlich, dass die Auswahl an Verfahren in Zukunft noch erweitert wird. Außerdem bildet die durchgeführte Suche nur die Kriterien ab, die für *jadice flow* als sinnvoll befunden wurden. Die Anzahl der sinnvollen Ergebnisse kann für andere Anwendungsfälle variieren.

Die in diesem Schritt notierten Gefühle sind relativ neutral. In der ersten Feldnotiz (Feldnotiz D.4) wurde Frustration vermerkt, da das erste Suchergebnis, für welches der ARH mit Abstand die größte Übereinstimmung angezeigt hat, als ungeeignet befunden wurde. In der letzten Feldnotiz (Feldnotiz D.14) wurde vermerkt, dass es Unsicherheiten bezüglich der zeitlichen Umsetzbarkeit der beiden ausgewählten Verfahren gibt. Diese Gefühle überwiegen aber nicht den verzeichneten positiven Gefühlen bei der Betrachtung der zwei favorisierten Verfahren in Feldnotizen D.5 und D.6.

Der in Feldnotiz D.4 notierte Verbesserungsvorschlag bezieht sich auf die Frustration in dieser Feldnotiz. Es wird vorgeschlagen, eine Filteroption zu implementieren, um unpassende Ergebnisse auszuschließen.

5.2.3. Phase 3: Anwendung der Migrationsverfahren

In der dritten Phase des MMF wurden lediglich vier Feldnotizen erstellt, da nur ein kleiner Teil der Phase durchgeführt wurde. Die Umsetzung der Phase 3b wurde frühzeitig ausgeschlossen. Der Hauptteil der Phase 3a, welcher die Anwendung eines Migrationsverfahrens beinhaltet, konnte aufgrund der zeitlichen Beschränkung einer Thesis und Problemen mit den Verfahren nur teilweise durchgeführt werden. Da in dieser Thesis nicht bis zur Planung einer neuen Architektur fortgeschritten wurde, fand auch keine Wahl von Best Practices und Patterns statt.

Die vier vorhandenen Feldnotizen beschreiben den Versuch, die beiden als Favoriten identifizierten Migrationsverfahren anzuwenden. Wie in Tabelle 5.4 zu sehen ist, war dieser Versuch nicht erfolgreich. In nur einer Feldnotiz wurde etwas notiert, das gut gelaufen ist. Dagegen ist in allen

Code	**Feldnotiz**				**Summe** (4)
	15	**16**	**17**	**18**	
Gut gelaufen		✓			1 (25%)
Schlecht gelaufen	✓	✓	✓	✓	4 (100%)
Mehr gut gelaufen					0 (0%)
Mehr schlecht gelaufen	✓	✓	✓	✓	3 (75%)
Gutes Gefühl					0 (0%)
Schlechtes Gefühl	✓	✓	✓	✓	4 (100%)
Unsicherheit		✓			1 (25%)
Frustration	✓		✓	✓	3 (75%)
Verbesserungs-vorschlag					0 (0%)
Einzelarbeit	✓	✓	✓		3 (75%)
Besprechung				✓	1 (25%)

Tabelle 5.4.: Auswertung der Kodierung der Feldnotizen in Phase 3 beim übergeordneten Schritt *Methodenanwendung*. Prozentuale Angaben hinter der Summe beziehen sich auf den Anteil des Auftretens des Codes zu der Anzahl der Feldnotizen.

Feldnotizen etwas schlecht gelaufen. 75 Prozent der Notizen wurden mit „Mehr schlecht gelaufen“ bewertet. Jede Notiz enthielt ein negatives Gefühl, dreimal Frustration und einmal Unsicherheit. Ein positives Gefühl wurde nie vermerkt.

Die schlechten Ergebnisse in dieser Phase sind hauptsächlich auf Zeitgründe und die Schwierigkeit bei der Umsetzung der ausgewählten Migrationsverfahren zurückzuführen. Eine detailliertere Beschreibung der Probleme ist in Abschnitt 4.3.1 gegeben. Für die Probleme werden hauptsächlich zwei Quellen vermutet. Zum einen wurden die Verfahren nur von einer Person im Detail betrachtet und versucht umzusetzen. Diese Person ist relativ unerfahren, was das Verständnis von wissenschaftlichen Artikeln mit komplexen Algorithmen und Formeln einschränken könnte. Auf der anderen Seite sind bestimmte Details, zumindest im zweiten Verfahren, wenig ausführlich

oder gar nicht beschrieben. Für eine erfolgreiche Umsetzung dieses Verfahrens wird ein hoher Eigenentwicklungsaufwand eingeschätzt, um den Algorithmus und enthaltene Formeln zu verstehen und umzusetzen.

5.2.4. Überblick über die Feldnotizen

Nachdem die Ergebnisse der Auswertung der Feldnotizen pro Phase beziehungsweise Schritt in einzelnen Abschnitten beschrieben wurden, erfolgt in diesem Abschnitt eine Betrachtung der Ergebnisse der Feldnotizen aus übergeordneter Perspektive. Für jeden übergeordneten Schritt wurde die Summe der Kodierung aller enthaltenen Feldnotizen zusammen in der Tabelle 5.5 visualisiert.

Code	**Übergeordneter Schritt**			**Summe** (18)
	Filter-wahl (3)	**Ergebnis-betrachtung** (11)	**Methoden-anwendung** (4)	
Gut gelaufen	3 (100%)	8 (73%)	1 (25%)	12 (67%)
Schlecht gelaufen	1 (33%)	7 (64%)	4 (100%)	12 (67%)
Mehr gut gelaufen	3 (100%)	4 (36%)	0 (0%)	7 (39%)
Mehr schlecht gelaufen	0 (0%)	2 (18%)	3 (75%)	5 (28%)
Gutes Gefühl	2 (67%)	2 (18%)	0 (0%)	4 (22%)
Schlechtes Gefühl	1 (33%)	2 (18%)	4 (100%)	7 (39%)
Unsicherheit	1 (33%)	1 (09%)	1 (25%)	3 (17%)
Frustration	0 (0%)	1 (09%)	3 (75%)	4 (22%)
Verbesserungs-vorschlag	1 (33%)	1 (09%)	0 (0%)	2 (11%)
Einzelarbeit	0 (0%)	8 (73%)	3 (75%)	11 (61%)
Besprechung	3 (100%)	3 (27%)	1 (25%)	7 (39%)

Tabelle 5.5.: Auswertung der Kodierung der Feldnotizen. Prozentuale Angaben in Klammern beziehen sich auf den Anteil des Auftretens des Codes zu der Anzahl der Feldnotizen in diesem Schritt beziehungsweise insgesamt.

Bei Betrachtung der unterschiedlichen Ergebnisse in den drei Schritten fällt vor allem auf, dass sich die Kodierungen „Mehr schlecht gelaufen“ und „Mehr gut gelaufen“ im Verlauf sehr negativ entwickeln. Im ersten Schritt wurden 100 Prozent mit „Mehr gut gelaufen“ kodiert, das kehrt sich im letzten Schritt um. Dort sind 75 Prozent mit „Mehr schlecht gelaufen“ kodiert. Dabei lässt sich insbesondere folgendes beobachten: Mit abnehmendem Erfolg der Aktivitäten ist der ARH immer weniger involviert an den Aktivitäten. Der erste Schritt befasst sich noch komplett mit dem ARH und verläuft gut. Ab dem zweiten Schritt wird die Wertung schlechter und hier ist der ARH schon weniger involviert. Obwohl der ARH die Ergebnisse vorgibt, kann er nur entfernter mit der Betrachtung der wissenschaftlichen Artikel assoziiert werden. In der dritten Phase ist die Bewertung dann am schlechtesten, doch die Kritik in der dritten Phase kann fast nur auf die Verfahren bezogen werden und nur in sehr geringem Maße auf den ARH.

Daraus kann geschlossen werden, dass der ARH einen Migrationsprozess dort, wo es die Grenzen seiner Funktionsweise erlauben, gut begleitet. Es gibt immer Optimierungspotenzial. Aus der Perspektive dieser Arbeit waren die Verfahren sowie deren etwaige Qualität am kritischsten. Daher scheint es am wichtigsten, die Sammlung von Verfahren sowie die Bewertung und Kategorisierung dieser zu optimieren.

5.3. Diskussion

In diesem Abschnitt wird der gesamte Refactoring-Prozess, den das MMF anleitet und der ARH umsetzt, bewertet. Dabei werden die vorangegangenen Auswertungen der Feldnotizen und Experteninterviews zusammengefasst und diskutiert. Zusätzlich fließt eine qualitative Bewertung des Gesamtprozesses durch den Autor ein, die sich aus den gesammelten Eindrücken ergibt.

Der Migrationsprozess des ARH bietet folgende wesentlichen Funktionen, die in dieser Arbeit untersucht werden sollen:

- Erfassung der QAs in Phase 1
- Konfiguration der Suchfilter in Phase 2
- Suche nach Migrationsverfahren in Phase 2
- Suche nach Best Practices und Patterns in Phase 3

Die beiden ersten Funktionen produzieren, abgesehen von der User Experience (UX) bei der Verwendung der Funktionen, keine eigenen Ergebnisse, die evaluiert werden könnten. Da sie jedoch die Voraussetzung für die dritte und vierte Funktion sind und deren Ergebnisse beeinflussen, werden sie indirekt durch deren Analyse ausgewertet.

Die Diskussion dieser Funktionen ist nach den Unterfragen der folgend erneut aufgeführten Forschungsfrage gegliedert, da für die Ergebnisse der Phasen 2 und 3 jeweils eine Frage formuliert wurde.

FF: Wie kann eine bereits bestehende Microservices-Architektur mit Hilfe des Microservices Migration Framework (MMF) hinsichtlich konkreter Qualitätsaspekte weiter optimiert werden?

1.1 Welche Refactoring-Verfahren eignen sich zur Bestimmung der optimalen Service-Granularität einer bestehenden Microservices-Architektur?

1.2 Welche Ansätze, Patterns oder Best Practices eignen sich zur Optimierung des Kommunikationsmodells und der Verringerung des IO-Flaschenhalses zwischen den einzelnen Services?

5.3.1. Forschungsfrage 1.1

Dieser Teil der Forschungsfrage bezieht sich auf die Ergebnisse der zweiten Phase, in der geeignete Migrationsverfahren für *jadice flow* untersucht wurden. Insbesondere wurde dabei die Frage nach einer möglichen Verbesserung der Granularität gestellt. Sowohl die Feldnotizen als auch die Experteninterviews wurden zur Beantwortung dieser Frage herangezogen.

Die Feldnotizen zeigen, dass die Konfiguration der Filter sehr erfolgreich war und die Suche nach Migrationsverfahren größtenteils positiv verlief. Der ARH hat in diesen Phasen wie erwartet funktioniert und nur zu geringfügigen Unsicherheiten und Problemen geführt, die aufgrund des jungen Entwicklungsstands des Tools zu erwarten sind. Es können beispielsweise Verbesserungen an der Filterfunktion vorgenommen werden. Ein ungeeignetes Migrationsverfahren hätte durch eine zusätzliche Filtereinstellung aus den Ergebnissen entfernt werden können. Außerdem sind einige Verfahren weniger geeignet für die Verwendung mit bereits vorhandenen MSAs. Für die Verwendung des ARH mit solchen Produkten könnte das Tool bessere Unterstützung bieten. Außerdem hat die Verwendung von vier Suchen mit leicht variierenden Eingaben nach Bewertung des Autors keinen gravierenden Unterschied im Vergleich zur realistischen Suche ergeben. Auch das spricht für die Qualität der Suchfunktion des ARH. Zusammenfassend kann diese Funktion als größtenteils zufriedenstellend bewertet werden. Die Verwendung von vier Suchen mit leicht variierenden Eingaben hat nach Bewertung des Autors keinen gravierenden Unterschied im Vergleich zur realistischen Suche gemacht. Die Suchfunktion des ARH kann zusammenfassend als größtenteils zufriedenstellend bewertet werden.

Wie in Abschnitt 5.2.3 beschrieben, war die Anwendung der ausgewählten Migrationsverfahren in der dritten Phase weniger erfolgreich. Es können mehrere Faktoren genannt werden, die zu dieser Situation beitragen. Die größten sind jedoch der begrenzte zeitliche Rahmen und die Erschwerung der Durchführung der Verfahren in kurzer Zeit aufgrund des Fehlens gewisser Qualitäten der Veröffentlichungen zu den Verfahren. Die Anwendung von toolunterstützten Verfahren wäre grundlegend tendenziell leichter möglich gewesen. Die Nutzung einer Applikation wird als einfacher und schneller umsetzbar eingeschätzt als die Verwendung eines in einem wissenschaftlichen Artikel beschriebenen Verfahrens. Beide ausgewählten Verfahren haben ein Tool in dem Artikel vorgestellt, welches aber in beiden Fällen nicht veröffentlicht wurde. Die Umsetzung des bevorzugten Verfahrens wurde aufgrund zu großer Komplexität in der begrenzten Zeit nicht angestrebt. Beim zweiten Verfahren wurde eine Anwendung versucht, jedoch nach einiger Zeit abgebrochen. Einige Details des Algorithmus waren dem Autor unverständlich, da sie nicht ausreichend erklärt wurden. Diese Probleme traten bei der konkreten Umsetzung auf und waren im Voraus nicht wirklich abschätzbar. An dieser Stelle wäre es wünschenswert, im ARH mehr Informationen über die Verfahren zur besseren Einschätzung zu erhalten, was in realistischer Abschätzung jedoch auch schwer umsetzbar ist.

Bei der fehlgeschlagenen Anwendung in dieser Thesis ist zu beachten, dass die Ressourcen für die Umsetzung der Verfahren anders begrenzt sind als es in anderen Anwendungen in der Industrie der Fall wäre. Die durchgeführten Experteninterviews haben gezeigt, dass die Migrationsverfahren, die durch den ARH erhalten wurden, als grundsätzlich gut und für die Anwendung mit *jadice flow* geeignet eingeschätzt wurden. Die Experten haben sowohl die beiden vorgestellten Migrationsverfahren als auch ihre spezifische Auswirkung auf die Optimierung der Service-Granularität von *jadice flow* als gewinnbringend bewertet. Die genannten Probleme könnten in Zukunft gelöst und Erfolg erzielt werden, wenn die Anwendung der Verfahren nicht so stark durch Zeit und Mitarbeiterzahl eingeschränkt wäre.

Auch wenn eine Suche nach geeigneten Migrationsverfahren ohne den ARH nicht durchgeführt wurde, vermuten der Autor und der PO von *jadice flow*, dass diese der Suche der mit dem ARH durchgeführten nicht vorzuziehen gewesen wäre. Auch das könnten zukünftige Arbeiten untersuchen.

5.3.2. Forschungsfrage 1.2

Dieser Teil der Forschungsfrage bezieht sich auf die Ergebnisse der dritten Phase, in der geeignete Best Practices und Patterns für *jadice flow* gefunden werden sollten. Insbesondere wurde dabei die Frage nach einer möglichen Verbesserung des Kommunikationsmodells und des damit verbundenen Netzwerk-Overheads gestellt.

Im Gegensatz zur Auswahl eines oder weniger Migrationsverfahren für die Planung einer neuen Architektur können und sollten mehrere Best Practices und Patterns für diese ausgewählt werden. Als Basis sollte jedoch die in Phase 2 selektierte Migrationsmethode durchgeführt werden und eine neue Architektur vorhanden sein, um sinnvoll adäquate Best Practices und Patterns zu finden und umsetzen zu können. Da diese Voraussetzung in dieser Arbeit nicht gegeben war, wurde auch die Auswahl von Best Practices und Patterns nicht durchgeführt.

Dieser Teil der Forschungsfrage kann also nur durch die Ergebnisse der Experteninterviews erörtert werden, da keine Auswahl oder Anwendung von Best Practices und Patterns stattgefunden hat. Die vier Fragen, die in den Interviews mit Bezug auf die vom ARH für *jadice flow* vorgeschlagenen Best Practices und Patterns gestellt wurden, haben folgende Ergebnisse gezeigt: Die Experten bewerteten die Sortierung der Best Practices und Patterns im Verhältnis zu den eingegebenen QAs größtenteils als sinnvoll. Außerdem stellten sie fest, dass bereits etwa 13 Patterns und 15 Best Practices in *jadice flow* implementiert sind und dass es zusätzlich sinnvoll sein könnte, in Zukunft weitere 5 Patterns und 4 Best Practices zu implementieren. Dies ist ein Hinweis darauf, dass das Tool eine gute Auswahl an Best Practices und Patterns bietet und dass diese Funktion zumindest im konkreten Anwendungsfall als sinnvoll erachtet werden kann.

Darüber hinaus wurde festgestellt, dass die Liste der Best Practices und Patterns einige Vorschläge enthält, die das Kommunikationsmodell und den damit verbundenen Netzwerk-Overhead von *jadice flow* verbessern könnten. An dieser Stelle haben zwei Experten angemerkt, dass eine Erweiterung der Auswahl an Best Practices und Patterns wünschenswert wäre, da die Anzahl der für diese Frage nützlichen Ergebnisse als gering eingeschätzt wurde.

Es ist jedoch zu beachten, dass der in der Forschungsfrage erwähnte Weg zur Verbesserung des Kommunikationsmodells und des damit verbundenen Netzwerk-Overheads durch Best Practices und Patterns nicht der einzige Weg zum genannten Ziel ist. Eine Veränderung der Granularität durch ein in Phase 2 gewähltes Verfahren könnte zu einer stark veränderten Kommunikation führen und eine Optimierung des Netzwerk-Overheads ermöglichen. Ein Beispiel hierfür wäre die Vergrößerung der Microservices. Dadurch würde die Kommunikation zwischen vorher verschiedenen Microservices nun im selben Microservice stattfinden und müsste nicht mehr über das Netzwerk laufen.

6. Validität und Einschränkungen

Eine umfassende Berücksichtigung der Validitätseinschränkungen ist für die Bewertung der Relevanz einer empirischen Studie von großer Bedeutung. Jede Arbeit hat durch die verwendete Methodik Einschränkungen hinsichtlich ihrer Validität [CS15]. Auch diese Arbeit hat gewisse Limitationen, die in diesem Kapitel erörtert werden. Runeson und Höst [RH09] stellen eine mögliche Klassifizierung der verschiedenen Faktoren vor, die Einfluss auf die Validität haben. Dieses Kapitel ist entsprechend die vier Klassen *konstruktionsbedingte Validität*, *Interne Validität*, *Reliabilität* und *Externe Validität* strukturiert.

6.1. Konstruktionsbedingte Validität

Dieser Aspekt der Validität (original *Construct validity*) drückt aus, inwieweit die verwendete Methodik tatsächlich die gestellte Forschungsfrage untersucht und ein Ergebnis liefert, das diese Forschungsfrage beantwortet [RH09].

Die übergeordnete Methodik der gesamten Fallstudie besteht in erster Linie aus der Anwendung des MMF auf *jadice flow* und der anschließenden Auswertung der Ergebnisse dessen. Der erste Schritt (die Anwendung des MMF) wird in der Forschungsfrage erwähnt und zielt direkt auf deren Beantwortung ab. Im zweiten Schritt (der Evaluation der Ergebnisse der Anwendung) hängt die konstruktionsbedingte Validität von den verwendeten Methoden ab.

Die Frage in den Experteninterviews wurden so konzipiert, dass sie sehr gezielte Fragen mit starkem Bezug zu den Unterfragen der Forschungsfrage enthalten. Für die erste Teilfrage wurden in einem dedizierten Teil (mit Bezug zur Granularität) Fragen zu Migrationsverfahren und deren Anwendung auf *jadice flow* gestellt. Für die zweite Unterfrage wurden in einem weiteren Teil Fragen zur Eignung der vom ARH vorgeschlagenen Best Practices und Patterns für *jadice flow* gestellt. Auch hier bezieht sich eine Frage speziell auf den in der Forschungsfrage genannten Qualitätsaspekt. Obwohl viele Fragen einen direkten Bezug zur Forschungsfrage haben, kann die konstruktionsbedingte Validität trotzdem eingeschränkt sein, falls Teilnehmer die Fragen anders verstanden haben.

Die Auswertung der Feldnotizen ist dagegen nur in indirekterer Form auf die Forschungsfrage ausgerichtet. Aus den Feldnotizen geht nicht direkt hervor, ob und wie gut die ARH für das Refactoring von *jadice flow* geeignet ist. Vielmehr geben sie einen Einblick in die Durchführung des Prozesses und die damit verbundenen Probleme, Emotionen und Anmerkungen. Diese Stellung der Methodik ist bei der Betrachtung der Ergebnisse zu berücksichtigen.

6.2. Interne Validität

Die interne Validität ist gegeben, wenn die untersuchten Kausalbeziehungen isoliert sind und andere Kausalitäten, die das Ergebnis ebenfalls beeinflussen, bekannt sind und berücksichtigt werden [RH09].

In Abbildung 6.1 wurde versucht, mögliche Auswirkungen von erwünschten Messfaktoren auf die Schritte in dieser Arbeit sowie von unerwünschten Störfaktoren auf diese zu skizzieren. Als Hauptstörfaktoren werden die *beteiligten Personen* sowie die *verwendete Methodik und Vorgehensweise* vermutet. Dies sind jeweils nur Oberbegriffe für Sammlungen von möglichen Einflüssen.

Beteiligte Personen bezieht sich primär darauf, welche Personen beteiligt waren. Es sollte aber auch beinhalten, welche Probleme diese Personen mitbringen (Emotionen, Differenzen, Machtunterschiede). Es können auch der Grad der Erfahrung der Personen, menschliches Versagen und viele andere Einflüsse beinhaltet sein, die von den beteiligten Personen ausgehen.

Der Begriff *Verwendete Methodik und Vorgehensweise* bezieht sich auf den gewählten Ansatz zur Durchführung der Maßnahme. Im Falle der Wahl der QAs ist dies die in Abschnitt 3.1.1 beschriebene Methodik des Architekturreviews. In den anderen Schritten bezieht es sich auf weniger formale Vorgehensweisen, beispielsweise auf die gemeinsame Filterwahl oder die Art und Weise, wie verschiedene Suchergebnisse aggregiert werden. All diese Vorgehensweisen können Auswirkungen auf die Ergebnisse haben, die es zu minimieren gilt.

6.3. Reliabilität

Für eine hohe Verlässlichkeit (original *Reliability*) ist es wichtig, dass die Subjektivität des Autors möglichst wenig Einfluss auf die Ergebnisse der Studie hat [RH09]. Im Falle dieser Arbeit ist die Reliabilität in einigen Punkten eingeschränkt.

Zum einen wurden die meisten Schritte in dieser Arbeit von nur einem Studenten durchgeführt, der nur über begrenzte Erfahrung in Software Engineering und wissenschaftlichem Arbeiten verfügt. Dieses Risiko wurde versucht zu minimieren, indem in zumeist wöchentlichen Abständen mit dem PO von *jadice flow* und dem universitären Betreuer die durchgeführten und nächsten Schritte besprochen wurden. Außerdem wurde das Architekturreview zur Erfassung der QAs, die die späteren Ergebnisse maßgeblich beeinflussen, mit insgesamt fünf wichtigen Mitgliedern des Entwicklungsteams durchgeführt. Damit sollte eine möglichst hohe Qualität der QAs sichergestellt werden. Insbesondere bei zu komplexen Details ist diese Strategie des Vier-Augen-Prinzips nicht möglich, da eventuell helfende Personen keine Zeit für die notwendige Einarbeitung hätten. Konkret ist dieses Problem in der Arbeit bei der Anwendung des Migrationsverfahrens in Abschnitt 4.3.1 aufgetreten. Hier haben ein Algorithmus und Formeln auf einer sehr tiefen Detailebene zu Problemen geführt, die vom Autor nicht in der begrenzten Zeit gelöst werden konnten. Gleichzeitig waren sie jedoch zu komplex für andere Personen, um eventuell helfen zu können.

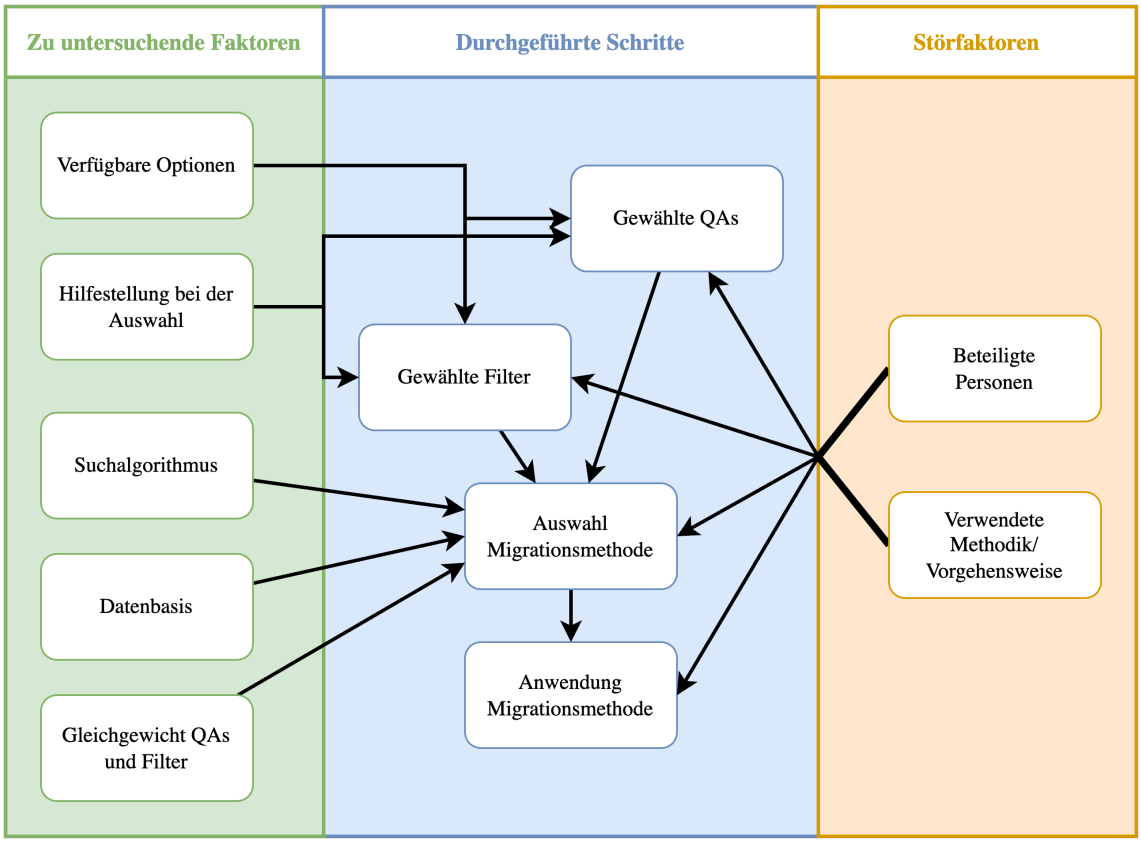

Abbildung 6.1.: Kausale Beziehungen zum Ergebnis dieser Fallstudie und generell der Anwendung des ARH.

Die gleiche Kritik gilt für die Methodik der Feldnotizen. Diese wurden nur von einer Person erstellt und ausgewertet, die keine Erfahrung mit dieser Methode hatte. Die Erstellung von Feldnotizen ist unabhängig von der Unerfahrenheit mit einem gewissen Maß an Subjektivität verbunden. Um diese Einschränkung zu minimieren, wurden für die Erhebung und Auswertung der Feldnotizen etablierte Methoden ausgewählt und vor der Anwendung umfangreiche Recherchen durchgeführt.

Eine weitere Einschränkung der Verlässlichkeit ergibt sich bei der Durchführung der Interviews. Sowohl die Befragten als auch der Interviewer könnten an einer positiven Bewertung der Arbeit interessiert sein, wodurch gewollt oder ungewollt mehr positive Ergebnisse produziert werden könnten. Außerdem können die persönlichen Beziehungen zu den Interviewteilnehmern und damit verbundene Sympathien und Antipathien Auswirkungen auf die Ergebnisse haben. Um die Objektivität zu erhöhen, wurden die Befragten explizit aufgefordert, ehrliche Antworten zu geben. Trotzdem kann eine gewisse Einschränkung nicht ausgeschlossen werden.

Darüber hinaus stellt die Auswertung der Daten generell ein Risiko für die Zuverlässigkeit dar. Diese wurde nur von einer unerfahrenen Person durchgeführt. Insbesondere bei der Kodierung der Daten ist eine gewisse Subjektivität unvermeidlich. Bei der Kodierung der Feldnotizen wurde versucht, einen möglichst objektiven Kodierleitfaden zu definieren. Die Kodierung der Experteninterviews

ist naturgemäß subjektiv. Da die Kodierung in diesem Fall aber auch nicht das Hauptergebnis ist, sondern die textliche Auswertung der Interviews wesentlich genauer beschrieben und fokussiert wird, stellt dies kein relevantes Problem dar.

6.4. Externe Validität

Die externe Validität beschreibt das Ausmaß, in dem die Ergebnisse der Studie verallgemeinert werden können [RH09]. In diesem Fallbeispiel bedeutet dies konkret, inwieweit von den Ergebnissen dieser Arbeit auf mögliche andere Anwendungen der ARH geschlossen werden kann. Externe Validität kann in der Regel nicht durch logische Maßnahmen sichergestellt werden, sondern nur durch die Annahme von Gesetzmäßigkeiten optimiert werden [CS15].

Die externe Validität ist hoch, wenn andere Anwendungen unter ähnlichen Bedingungen stattfinden [CS15]. Daher sind die variierenden Eingaben bei Verwendung des ARH eine wichtige Einschränkung. Die Ergebnisse der Suche nach Migrationsverfahren und der Suche nach Best Practices und Patterns hängen vollständig von den gewählten QAs und Filteroptionen ab. Wenn andere Eingaben verwendet werden, kann dies zu einer anderen Reihenfolge der Suchergebnisse und folgend anderen Ergebnissen in der Anwendung führen.

Darüber hinaus wird die externe Validität im Bezug auf die allgemeine Anwendung des ARH durch den Spezialfall in dieser Thesis, dass die Ausgangsarchitektur eine MSA ist, weiter eingeschränkt. Dadurch besteht eine geringere Ähnlichkeit zu anderen Anwendungsfällen mit Monolithen als Ausgangsarchitektur, was nach Campbell und Stanley [CS15] zu einer schlechteren externen Validität in diesen Fällen führt. Auf der anderen Seite erhöht genau diese Bedingung die externe Validität in Bezug auf die Verwendung des ARH mit MSAs im Vergleich zu beispielsweise der Fallstudie von Knodel [Kno23].

Die einzige Maßnahme, die zur Verbesserung der externen Validität ergriffen werden kann, ist die Sicherstellung einer möglichst hohen Konstruktvalidität, internen Validität und Reliabilität, sodass zumindest die Voraussetzungen für die externe Validität optimiert sind.

7. Fazit

Das Ziel dieser Arbeit bestand darin, die Verwendung des Microservices Migration Frameworks (MMF) und des Architecture Refactoring Helpers (ARH) der Abteilung Empirisches Software Engineering (ESE) des Institute of Software Engineering (ISTE) mit einer bestehenden Microservices-Architektur (MSA) in einer industriellen Fallstudie zu evaluieren. Dazu wurden das Framework und das Tool, welche in vorausgegangenen wissenschaftlichen Arbeiten entwickelt wurden, zum Refactoring des Produkts *jadice flow* der Firma *levigo solutions* verwendet. Der Fokus lag dabei auf der Beantwortung der Forschungsfrage:

FF: Wie kann eine bereits bestehende Microservices-Architektur mit Hilfe des Microservices Migration Framework (MMF) hinsichtlich konkreter Qualitätsaspekte weiter optimiert werden?

i. Welche Refactoring-Verfahren eignen sich zur Bestimmung der optimalen Service-Granularität einer bestehenden Microservices-Architektur?

ii. Welche Ansätze, Patterns oder Best Practices eignen sich zur Optimierung des Kommunikationsmodells und der Verringerung des IO-Flaschenhalses zwischen den einzelnen Services?

Um diese Fragen zu beantworten, wurde nach einer thematischen Übersicht in Kapitel 2 und der Planung der Methodik in Kapitel 3 in Kapitel 4 mit der Anwendung des MMF auf *jadice flow* begonnen. In der ersten Phase des Frameworks wurde ein Architekturreview durchgeführt, um die gewünschten QAs des Systems szenarienbasiert zu sammeln. Zusätzlich wurde am Anfang der zweiten Phase eine Konfiguration von weiteren Suchfiltern vorgenommen. Mit vier verschiedenen Variationen dieser Ausgangsdaten wurden dann vier Suchen nach Migrationsverfahren durchgeführt und die Ergebnisse dessen zu einer Liste aggregiert. Die ersten acht Ergebnisse dieser Liste wurden anschließend vom Autor detaillierter betrachtet und zusammen mit dem PO von *jadice flow* hinsichtlich ihrer Anwendbarkeit in dieser Thesis auf *jadice flow* evaluiert. Daraufhin wurde versucht, die daraus resultierenden zwei bevorzugten Verfahren im verfügbaren Zeitrahmen umzusetzen. Aufgrund von Zeitmangel und Problemen mit der Umsetzung der Verfahren ist dieser Teil fehlgeschlagen. Eine Auswertung der Fallstudie durch Evaluation konkreter Ergebnisse eines Refactorings war daher nicht möglich.

Die Auswertung wurde stattdessen mittels Experteninterviews zur Einschätzung der Anwendbarkeit beider Verfahren durchgeführt. Zusätzlich wurden während der Durchführung ab der zweiten Phase strukturierte Feldnotizen nach Seaman [Sea08] gesammelt. Zusammen mit den Experteninterviews bilden diese Methoden die gesamte Methodik zur Auswertung dieser Fallstudie. Obwohl die Anwendung der Migrationsverfahren in dieser Arbeit nicht vollendet werden konnte, zeigt die Auswertung der Feldnotizen, dass die Teile des Prozesses, in die der ARH involviert war, gut verlaufen sind. Des Weiteren wurde in den Experteninterviews untersucht, inwieweit die zwei favorisierten Verfahren in einer zukünftigen, weniger zeitlich und personell beschränkten Umgebung nützlich wären. Die Verfahren wurden von allen Experten als potenziell gewinnbringend für das Produkt eingeschätzt. Es wurde angegeben, dass eine Verbesserung der Granularität von *jadice flow* mit diesen für möglich

gehalten wird. Da die vom ARH vorgeschlagene Liste von Best Practices und Patterns in der Thesis nicht zur Anwendung kam, wurde hier ebenfalls eine Bewertung der Experten eingeholt. Auch dabei ergab sich eine positive Rückmeldung. Dies bezog sich auch auf die in der Forschungsfrage thematisierten Qualitäten des Kommunikationsmodells und Netzwerk-Overheads. Abschließend wurde von den Experten die generelle Funktionsweise des Frameworks evaluiert. Dabei wurden die einzelnen Funktionen sowie das Vorgehen in Phasen positiv hervorgehoben.

Zusammenfassend kann festgestellt werden, dass der ARH ein vielversprechendes Tool für die Migration von Monolithen zu MSAs darstellt und auch Potential zur Optimierung bestehender MSAs gesehen wird.

7.1. Ausblick

Es gibt noch einige Herausforderungen, die in Zukunft angegangen werden müssen, um die Effizienz und Effektivität des ARH zu verbessern. In der Arbeit wurden Verbesserungsvorschläge für das Framework und das Tool genannt, deren Umsetzung die Qualität erhöhen könnte. Außerdem ist die Datenbasis des ARH einer der größten Einflussfaktoren auf seine Qualität. Daher ist es wichtig, die Auswahl an Migrationsverfahren, Best Practices und Patterns in Zukunft zu erweitern und auf dem neuesten Stand zu halten. Außerdem kann der Filterungsmechanismus der Suchen in Phasen 2 und 3, neben den konkreten Vorschlägen in dieser Arbeit, allgemein immer weiter verbessert werden. Es könnte beispielsweise interessant sein, Künstliche Intelligenz (KI) zu verwenden, um die Verwendung der Filterfunktionen zu erleichtern sowie ihre Verfügbarkeit zu erhöhen und die damit verbundene Komplexität für den Nutzer trotzdem gering zu halten.

Außerdem sollte die Nutzung des ARH weiter untersucht werden, da die vorhandenen zwei Fallstudien vermutlich nur einen kleinen Teil der realen Anwendungsfälle repräsentieren und die externe Validität dieser somit eingeschränkt ist. Insbesondere der spezielle Anwendungsfall, bei dem eine MSA mit dem Tool überarbeitet wird, verursacht einige Risikofaktoren, die die Anwendung schwieriger und möglicherweise nicht in jedem Fall sinnvoll machen. Es gilt, diese Risikofaktoren weiter zu untersuchen.

Im Hinblick auf *jadice flow* wurde in dieser Arbeit gezeigt, dass eine Verbesserung der Architektur durch die Anwendung des ARH sowie konkreter die Anwendung der ausgewählten Migrationsmethoden als gewinnbringend angesehen wird.

In dieser Arbeit wurden wichtige Schritte zur Weiterentwicklung von *jadice flow* angestoßen. Im Rahmen der Recherche und der teilweisen Anwendung des Frameworks und des Tools wurden an vielen Stellen neue Lösungen für vorhandene Probleme erkannt. Diese sollten in Zukunft weiterverfolgt werden, um die Architektur von *jadice flow* zu optimieren.

Literaturverzeichnis

[AAE16] N. Alshuqayran, N. Ali, R. Evans. „A Systematic Mapping Study in Microservice Architecture“. In: *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. Nov. 2016, S. 44–51. DOI: 10.1109/SOCA.2016.15 (zitiert auf S. 37).

[AAQ+23] A. Alshammari, A. Almadhor, S. N. Qasem, J. H. Alkhateeb, K. Amjad. „High-performance computing-enabled probabilistic framework for migration from monolithic to microservices architecture using genetic algorithms“. In: *Soft Computing* (Okt. 2023). ISSN: 1433-7479. DOI: 10.1007/s00500-023-09336-w. URL: https://doi.org/10.1007/s00500-023-09336-w (zitiert auf S. 17).

[ACC+22] W. K. G. Assunção, T. E. Colanzi, L. Carvalho, A. Garcia, J. A. Pereira, M. J. de Lima, C. Lucena. „Analysis of a many-objective optimization approach for identifying microservices from legacy systems“. In: *Empirical Software Engineering* 27.2 (Feb. 2022), S. 51. ISSN: 1573-7616. DOI: 10.1007/s10664-021-10049-7. URL: https://doi.org/10.1007/s10664-021-10049-7 (zitiert auf S. 21, 22, 46, 47, 49, 51, 102, 103).

[ALFT21] F. Auer, V. Lenarduzzi, M. Felderer, D. Taibi. „From monolithic systems to Microservices: An assessment framework“. In: *Information and Software Technology* 137 (2021), S. 106600. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2021.106600. URL: https://www.sciencedirect.com/science/article/pii/S0950584921000793 (zitiert auf S. 37).

[BCK12] L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice: Software Architect Practice_c3*. Addison-Wesley, 2012 (zitiert auf S. 36, 37).

[BCK98] L. Bass, P. Clements, R. Kazman. *Software architecture in practice*. Addison-Wesley Publishing Co., 1998 (zitiert auf S. 24).

[CS15] D. T. Campbell, J. C. Stanley. *Experimental and quasi-experimental designs for research*. Ravenio books, 2015 (zitiert auf S. 71, 74).

[DBFP21] A. A. C. De Alwis, A. Barros, C. Fidge, A. Polyvyanyy. „Microservice Remodularisation of Monolithic Enterprise Systems for Embedding in Industrial IoT Networks“. In: *Advanced Information Systems Engineering*. Hrsg. von M. La Rosa, S. Sadiq, E. Teniente. Cham: Springer International Publishing, 2021, S. 432–448. ISBN: 978-3-030-79382-1 (zitiert auf S. 21, 22, 46, 47, 49, 51, 101, 102).

[DEF+21] M. Daoud, A. El Mezouari, N. Faci, D. Benslimane, Z. Maamar, A. El Fazziki. „A multi-model based microservices identification approach“. In: *Journal of Systems Architecture* 118 (2021), S. 102200. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2021.102200. URL: https://www.sciencedirect.com/science/article/pii/S1383762121001442 (zitiert auf S. 21, 22, 46, 47, 51–54, 60, 98, 100, 102, 103, 105).

[DGL+17] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, L. Safina. „Microservices: Yesterday, Today, and Tomorrow“. In: *Present and Ulterior Software Engineering*. Hrsg. von M. Mazzara, B. Meyer. Cham: Springer International Publishing, 2017, S. 195–216. ISBN: 978-3-319-67425-4. DOI: 10.1007/978-3-319-67425-4_12. URL: https://doi.org/10.1007/978-3-319-67425-4_12 (zitiert auf S. 15).

[DJ14] K. Deb, H. Jain. „An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints“. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), S. 577–601. DOI: 10.1109/TEVC.2013.2281535 (zitiert auf S. 50).

[FBH+22] J. Fritzsch, J. Bogner, M. Haug, S. Wagner, A. Zimmermann. *Towards an Architecture-centric Methodology for Migrating to Microservices*. 2022. arXiv: 2207.00507 [cs.SE] (zitiert auf S. 13, 17, 18, 20).

[FBWZ19] J. Fritzsch, J. Bogner, S. Wagner, A. Zimmermann. „Microservices Migration in Industry: Intentions, Strategies, and Challenges“. In: *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Sep. 2019. DOI: 10.1109/icsme.2019.00081. URL: https://doi.org/10.1109%2Ficsme.2019.00081 (zitiert auf S. 13).

[FBZW19] J. Fritzsch, J. Bogner, A. Zimmermann, S. Wagner. „From Monolith to Microservices: A Classification of Refactoring Approaches“. In: *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Hrsg. von J.-M. Bruel, M. Mazzara, B. Meyer. Cham: Springer International Publishing, 2019, S. 128–141. ISBN: 978-3-030-06019-0 (zitiert auf S. 20).

[FCBW23] J. Fritzsch, F. Correia, J. Bogner, S. Wagner. *Tools for Refactoring to Microservices: A Preliminary Usability Report*. 2023. arXiv: 2311.04798 [cs.SE] (zitiert auf S. 54).

[FFC21] F. Freitas, A. Ferreira, J. Cunha. „Refactoring Java Monoliths into Executable Microservice-Based Applications“. In: *Proceedings of the 25th Brazilian Symposium on Programming Languages*. SBLP '21. Joinville, Brazil: Association for Computing Machinery, 2021, S. 100–107. ISBN: 9781450390620. DOI: 10.1145/3475061.3475086. URL: https://doi.org/10.1145/3475061.3475086 (zitiert auf S. 21, 46–48, 50, 100, 102).

[FHK22] J. Fritzsch, T. Haller, D. Koch. *GitHub repository of the Architecture Refactoring Helper*. 2022. URL: https://github.com/jfr609/architecture-refactoring-helper (zitiert auf S. 17).

[FQA+23] G. Filippone, N. Qaisar Mehmood, M. Autili, F. Rossi, M. Tivoli. „From monolithic to microservice architecture: an automated approach based on graph clustering and combinatorial optimization“. In: *2023 IEEE 20th International Conference on Software Architecture (ICSA)*. 2023, S. 47–57. DOI: 10.1109/ICSA56044.2023.00013 (zitiert auf S. 22).

[GIT23] G. Garbi, E. Incerto, M. Tribastone. „µP: A Development Framework for Predicting Performance of Microservices by Design“. In: *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*. 2023, S. 178–188. DOI: 10.1109/CLOUD60044.2023.00029 (zitiert auf S. 21).

[HA05] S. Hove, B. Anda. „Experiences from conducting semi-structured interviews in empirical software engineering research“. In: *11th IEEE International Software Metrics Symposium (METRICS'05)*. 2005, 10 pp.–23. DOI: 10.1109/METRICS.2005.24 (zitiert auf S. 29, 30).

[Hal22] T. Haller. *Design, Implementation and Evaluation of an Application for Guiding Architectural Refactoring to Microservices*. 2022. DOI: 10.18419/opus-12272 (zitiert auf S. 17, 20).

[HLT18] S. Habibullah, X. Liu, Z. Tan. „An approach to evolving legacy enterprise system to microservice-based architecture through feature-driven evolution rules“. In: *International Journal of Computer Theory and Engineering* 10.5 (2018), S. 164–169. DOI: 10.7763/IJCTE.2018.V10.1219 (zitiert auf S. 45–47, 50, 98, 100).

[Int00] International Organization for Standardization/International Electrotechnical Commission. *ISO/IEC FDIS 9126-1:2000(E), Software engineering - Product quality - Part 1: Quality model*. Standard. 2000 (zitiert auf S. 25, 26).

[Int11] International Organization for Standardization/International Electrotechnical Commission. *ISO/IEC 25010:2011, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. Standard. 2011 (zitiert auf S. 20, 26, 36, 37).

[KKC00] R. Kazman, M. Klein, P. Clements. *ATAM: Method for Architecture Evaluation*. Techn. Ber. CMU/SEI-2000-TR-004. Accessed: 2023-Oct-16. Aug. 2000. URL: https://insights.sei.cmu.edu/library/atam-method-for-architecture-evaluation/ (zitiert auf S. 24, 26, 27, 39).

[Kno23] M. Knodel. *Migration monolithischer Anwendungen in Microservices-basierte Architekturen: Fallstudie einer Service/Sales-Applikation*. 2023. DOI: 10.18419/opus-13687 (zitiert auf S. 13, 17, 20, 21, 24, 28, 32, 74).

[Koc22] D. Koch. *Migrating Monolithic Architecturesto Microservices: A Study onSoftware Quality Attributes*. 2022. DOI: 10.18419/opus-12676 (zitiert auf S. 17, 20, 26, 36, 37).

[LZJ+21] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen, M. A. Babar. „Understanding and addressing quality attributes of microservices architecture: A Systematic literature review“. In: *Information and Software Technology* 131 (2021), S. 106449. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2020.106449. URL: https://www.sciencedirect.com/science/article/pii/S0950584920301993 (zitiert auf S. 36).

[MBC23] H. Medeiros, T. Batista, E. Cavalcante. „On a Metaprocess for Microservice Migration“. In: *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. SBES '23. Campo Grande, Brazil: Association for Computing Machinery, 2023, S. 116–121. ISBN: 9798400707872. DOI: 10.1145/3613372.3613382. URL: https://doi.org/10.1145/3613372.3613382 (zitiert auf S. 21).

[MCF+20] T. Matias, F. F. Correia, J. Fritzsch, J. Bogner, H. S. Ferreira, A. Restivo. „Determining Microservice Boundaries: A Case Study Using Static and Dynamic Software Analysis“. In: *Software Architecture*. Hrsg. von A. Jansen, I. Malavolta, H. Muccini, I. Ozkaya, O. Zimmermann. Cham: Springer International Publishing, 2020, S. 315–332. ISBN: 978-3-030-58923-3. DOI: 10.1007/978-3-030-58923-3_21 (zitiert auf S. 21, 22, 46, 47, 49, 51, 101, 102).

[MF19] P. Mayring, T. Fenzl. „Qualitative Inhaltsanalyse“. In: *Handbuch Methoden der empirischen Sozialforschung*. Hrsg. von N. Baur, J. Blasius. Wiesbaden: Springer Fachmedien Wiesbaden, 2019, S. 633–648. ISBN: 978-3-658-21308-4. DOI: 10.1007/978-3-658-21308-4_42. URL: https://doi.org/10.1007/978-3-658-21308-4_42 (zitiert auf S. 30–32).

[ML14] F. Murtagh, P. Legendre. „Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?“ In: *Journal of Classification* 31.3 (Okt. 2014), S. 274–295. ISSN: 1432-1343. DOI: 10.1007/s00357-014-9161-z. URL: http://dx.doi.org/10.1007/s00357-014-9161-z (zitiert auf S. 47).

[MM22] M. Milić, D. Makajić-Nikolić. „Development of a Quality-Based Model for Software Architecture Optimization: A Case Study of Monolith and Microservice Architectures“. In: *Symmetry* 14.9 (2022). ISSN: 2073-8994. DOI: 10.3390/sym14091824. URL: https://www.mdpi.com/2073-8994/14/9/1824 (zitiert auf S. 37).

[MN09] M. Meuser, U. Nagel. „Das Experteninterview — konzeptionelle Grundlagen und methodische Anlage“. In: *Methoden der vergleichenden Politik- und Sozialwissenschaft: Neue Entwicklungen und Anwendungen*. Hrsg. von S. Pickel, G. Pickel, H.-J. Lauth, D. Jahn. Wiesbaden: VS Verlag für Sozialwissenschaften, 2009, S. 465–479. ISBN: 978-3-531-91826-6. DOI: 10.1007/978-3-531-91826-6_23. URL: https://doi.org/10.1007/978-3-531-91826-6_23 (zitiert auf S. 29).

[OMG18] OMG. *Essence – Kernel and Language for Software Engineering Methods, Version 1.2*. OMG, USA, 2018. URL: https://www.omg.org/spec/Essence/1.2/About-Essence (zitiert auf S. 21).

[RH09] P. Runeson, M. Höst. „Guidelines for conducting and reporting case study research in software engineering“. In: *Empirical Software Engineering* 14.2 (Apr. 2009), S. 131–164. ISSN: 1573-7616. DOI: 10.1007/s10664-008-9102-8. URL: https://doi.org/10.1007/s10664-008-9102-8 (zitiert auf S. 29, 30, 71, 72, 74).

[Sea08] C. B. Seaman. „Qualitative Methods“. In: *Guide to Advanced Empirical Software Engineering*. Hrsg. von F. Shull, J. Singer, D. I. K. Sjøberg. London: Springer London, 2008, S. 35–62. ISBN: 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_2. URL: https://doi.org/10.1007/978-1-84800-044-5_2 (zitiert auf S. 28–30, 75).

[Sin16] A. Singleton. „The Economics of Microservices“. In: *IEEE Cloud Computing* 3.5 (2016), S. 16–20. DOI: 10.1109/MCC.2016.109 (zitiert auf S. 16).

[SM07] M. Svahnberg, F. Mårtensson. „Six years of evaluating software architectures in student projects“. In: *Journal of Systems and Software* 80.11 (2007), S. 1893–1901. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2007.01.050. URL: https://www.sciencedirect.com/science/article/pii/S016412120700060X (zitiert auf S. 23–27, 37).

[SMNB21] J. Soldani, G. Muntoni, D. Neri, A. Brogi. „The µTOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures“. In: *Software: Practice and Experience* 51.7 (2021), S. 1591–1621. DOI: https://doi.org/10.1002/spe.2974. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2974. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2974 (zitiert auf S. 21).

[SS11] K. Schwaber, J. Sutherland. „The scrum guide“. In: *Scrum Alliance* 21.1 (2011), S. 1–38 (zitiert auf S. 29, 35).

[SSB+20] A. Selmadji, A.-D. Seriai, H. L. Bouziane, R. Oumarou Mahamane, P. Zaragoza, C. Dony. „From Monolithic Architecture Style to Microservice one Based on a Semi-Automatic Approach“. In: *2020 IEEE International Conference on Software Architecture (ICSA)*. 2020, S. 157–168. DOI: 10.1109/ICSA47634.2020.00023 (zitiert auf S. 46, 47, 50, 51, 103).

[SZ12] S. X. Sun, J. Zhao. „A decomposition-based approach for service composition with global QoS guarantees“. In: *Information Sciences* 199 (2012), S. 138–153. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2012.02.061. URL: https://www.sciencedirect.com/science/article/pii/S0020025512001892 (zitiert auf S. 37).

[TLP17] D. Taibi, V. Lenarduzzi, C. Pahl. „Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation“. In: *IEEE Cloud Computing* 4.5 (2017), S. 22–32. DOI: 10.1109/MCC.2017.4250931 (zitiert auf S. 13, 16).

[TS19] D. Taibi, K. Systä. *A Decomposition and Metric-Based Evaluation Framework for Microservices*. 2019. arXiv: 1908.08513 [cs.SE] (zitiert auf S. 16).

[VF23] V. Velepucha, P. Flores. „A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges“. In: *IEEE Access* 11 (2023), S. 88339–88358. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3305687 (zitiert auf S. 15, 16).

[VPAG21] F. H. Vera-Rivera, E. Puerto, H. Astudillo, C. M. Gaona. „Microservices Backlog–A Genetic Programming Technique for Identification and Evaluation of Microservices From User Stories“. In: *IEEE Access* 9 (2021), S. 117178–117203. DOI: 10.1109/ACCESS.2021.3106342 (zitiert auf S. 21, 22, 46–48, 51–54, 60, 99, 100, 102–104).

Alle URLs wurden zuletzt am 10. März 2024 geprüft.

A. Leitfaden Experteninterviews

Dieser Leitfaden beschreibt den Plan, nach dem in den Experteninterviews vorgegangen wird.

A.1. Übersicht

Eine Übersicht über den Rahmen der Experteninterviews ist in Tabelle A.1 gegeben sowie ein Zeitplan in Tabelle A.2.

Ziel	Bewertung der Nützlichkeit des MMF/ARH bei Migration zu Microservices im Spezialfall *jadice flow*
Art des Interviews	Semi-Strukturiert, Einzelinterviews
Interviewte	Product Owner, Softwarearchitekt, Softwareentwickler
Interviewer	Axel Herrmann
Dauer	45 - 60 min
Datum	26.02.2024 und 29.02.2024
Ort	Online-Meeting

Tabelle A.1.: Übersicht über die Experteninterviews

Schritt	**Ziel**	**Dauer**
Einführung	Begrüßung, Zustimmung zur Verarbeitung und Aufzeichnung, Erfassung grundlegender Daten	5 min
Verständnisfragen zu MMF/ARH	Verständnisprobleme eliminieren, damit spätere Fragen auf vergleichbarem Wissensstand durchgeführt werden können	10 min
Allgemeine Vorgehensweise eines/des MMF	Sehr abstrakte Bewertung MMF/ARH	10 min
Evaluation eines potentiellen Refactorings von *jadice flow* mit MMF/ARH	Hauptteil. Bewertung der in dieser Arbeit durchgeführten Anwendung des ARH auf *jadice flow*	20 min

Tabelle A.2.: Zeitplan der Experteninterviews

A.2. Einführung

Zu Beginn oder je nach Betrachtung vor Beginn der Experteninterviews werden einige Formalitäten geklärt. Dabei wird nach einem klaren, strukturierten Plan vorgegangen:

1. Zustimmung zur Aufzeichnung des Interviews und zur Verarbeitung der Daten
2. Übersicht über die Präambel
3. Start der Aufzeichnung
4. Dem Interviewten seine Rolle nennen (da einzelne Interviewte teilweise mehrere Rollen innehaben)
5. Professionelle Erfahrung in der Informatik (in Jahren)
6. Erfahrung mit Microservices (in Jahren)
7. Arbeit an dem Produkt (in Jahren, *jadice server* zählt als Vorgänger ebenfalls zu *jadice flow*)

A.3. Evaluation

Nach der Einleitung, in der keine inhaltlichen Fragen gestellt werden, kann der Hauptteil der Experteninterviews beginnen.

A.3.1. Verständnisfragen zu MMF/ARH

Da der Informationsbogen relativ komplexe Konstrukte enthält, ist die Erwartung nicht, dass jeder Teilnehmer zu Beginn des Interviews alles darauf verstanden hat. Deswegen werden wir anfangs etwas über die enthaltenen Informationen sprechen, um mögliche Lücken zu schließen.

1. Hatten Sie Probleme, Teile des Informationsmaterials zu verstehen, soll ich bestimmte Details genauer erläutern?
2. Falls nicht durch Besprechung der Probleme bereits geschehen:
    - Kurze Zusammenfassung über MMF/ARH
    - Kurze Zusammenfassung über die zwei enthaltenen Migrationsverfahren

A.3.2. Evaluation des MMF/ARH allgemein

Versetzen Sie sich zurück in die Zeit, in der die Planung der Migration von *jadice server* zu einer Microservices-Architektur bevorstand, aus der dann *jadice flow* entstanden ist. Sie suchen nach einem Tool oder einer Software in einer beliebigen Form, das oder die Sie bei der Migration zu Microservices unterstützen soll.

1. Welche Aufgaben könnten Sie sich vorstellen, bei denen Sie ein solches Tool sinnvoll unterstützen könnte?
2. Falls noch nicht in der Antwort erläutert: Welche Qualitäten soll das Tool mit sich bringen?
3. Finden Sie die allgemeine Funktions- und Vorgehensweise des MMF/ARH, **bezogen auf die einzelnen Phasen**, sinnvoll?
4. Sehen Sie **Probleme** bei der Funktionsweise des Frameworks?

A.3.3. Evaluation der Anwendung des MMF/ARH auf *jadice flow*

Sie kennen zwei der vorgeschlagenen Migrationsmethoden. Nun soll eine potentielle Anwendung dieser Methoden mit *jadice flow* bewertet werden.

1. Jeweils für Methode 1 und 2:
   - **Allgemein:** Wie nützlich schätzen sie diese Methode bei einer potentiellen Anwendung mit *jadice flow* ein?
   - Falls nicht bereits erwähnt: Wie passend wäre Vorgehensweise der Methode auf einer **abstrakten Ebene** (Art der Eingabe, Bestimmung der Metriken, Extraktion von Microservices)?
   - Falls nicht bereits erwähnt: Auf einer **weniger abstrakten** Ebene: Welche konkreten Probleme könnten Sie sich bei der Anwendung mit *jadice flow* vorstellen?
   - Denken Sie, die **Granularität** der Microservices von *jadice flow* könnte potentiell verbessert werden durch die Anwendung dieser Methode?
2. Jeweils für Best Practices und Patterns
   - Finden Sie die **Sortierung passend** zu den QAs *Scalability*, *Maintainability*, *Performance* und *Portability*?
   - Wie viele der gezeigten Best Practices/Patterns sind in *jadice flow* **bereits angewendet**/teilweise angewendet?
   - Wie viele der gezeigten Best Practices/Patterns würden Sie als **potentiell sinnvoll** für die Implementierung einschätzen?
   - Sind in der Liste Best Practices/Patterns enthalten, die Sie als potentiell geeignet für die **Optimierung des Kommunikationsmodells bzw. des Netzwerk-Overheads** von *jadice flow* einschätzen würden?

3. Würden Sie die Verwendung des MMF/ARH für *jadice flow* zum jetzigen Zeitpunkt als sinnvoll und potentiell gewinnbringend betrachten?
4. Hätten Sie die Verwendung des MMF/ARH für die damalige Migration von *jadice server* zu Microservices als sinnvoll und potentiell gewinnbringend betrachtet?

B. Präambel Experteninterviews

In diesem Dokument werden die allgemeinen Richtlinien und ethischen Erwägungen für die Experteninterviews dieser Fallstudie beschrieben.

B.1. Ziel der Studie

In der Thesis wird das toolunterstützte Refactoring von Microservices mit dem Tool *Architecture Refactoring Helper* nach dem *Microservices Migration Framework* erforscht. Ziel der Fallstudie ist es, diese am Produkt *jadice flow* der Firma *levigo solutions* zu testen und diesen Test zu evaluieren. Damit soll auf der einen Seite eine potentielle Überarbeitung von *jadice flow* angestoßen werden und auf der anderen Seite Erkenntnisse über die Effektivität des Frameworks und Tools gewonnen werden, die in Zukunft bei der Weiterentwicklung dieser helfen sollen.

Das Framework hat allgemein das Ziel, die Lücke zwischen Industrie und Forschung zu verringern, indem wissenschaftliche Methoden für die Migration zu Microservices-Architekturen in einem webbasierten Tool kategorisiert und so für die Industrie zugänglich gemacht werden.

B.2. Ziel der Experteninterviews

In den Interviews soll eine Evaluation der Anwendung des Frameworks und Tools auf *jadice flow* in dieser Arbeit stattfinden. Dabei wird sowohl eine allgemeine Bewertung des Konzepts des Frameworks, als auch der speziellen Anwendung des Tools mit *jadice flow*, die in dieser Arbeit stattgefunden hat, durchgeführt.

B.3. Vertraulichkeit und Anonymität

Alles, was in den Interviews gesagt wird, wird streng vertraulich behandelt und ohne die Erlaubnis der Teilnehmer an niemanden außerhalb unseres Forschungsteams weitergegeben. Die Ergebnisse werden vor der Veröffentlichung anonymisiert. Aufgrund der geringen Teilnehmerzahl und der Bekanntgabe der Rollen könnten jedoch Aussagen von einem Personenkreis, der die Personen hinter den Rollen kennt, auf Teilnehmer zurückgeführt werden. Auch nach dem Interview haben die Teilnehmer die Möglichkeit, Aussagen vor der Auswertung auszuschließen oder zu schwärzen. Die vollständigen Interviewtranskripte werden NICHT veröffentlicht.

B.4. Interviewprozess und Analyse

Die Befragten werden über das allgemeine Thema der Studie sowie speziell für das Interview relevante Details informiert, so dass sie damit vertraut sind und sich vorab informieren können. Die Dauer des Interviews wird auf 45 Minuten geschätzt. Sie werden um Erlaubnis gebeten, das Interview für die Transkription aufzuzeichnen. Nach der Transkription wird die Tonaufnahme vernichtet und das Transkript wird die einzige Grundlage für die qualitative Inhaltsanalyse sein.

C. Informationsbogen Experteninterviews

Dieser Informationsbogen wurde erstellt, sowohl um Zeit in den eigentlichen Experteninterviews zu sparen, als auch, damit alle Experten bei der Befragung möglichst auf dem gleichen Wissensstand sind. Es wird ein Überblick über das Microservices Migration Framework (MMF) und den Architecture Refactoring Helper (ARH) gegeben. Außerdem werden zwei Migrationsverfahren, die in den Interviews behandelt werden, vorgestellt. Der Bogen wird allen Experten im Vorhinein ausgehändigt, es wird aber kein vollständiges Verständnis der Informationen erwartet. Im Vorfeld des Interviews werden die enthaltenen Informationen nochmals kurz zusammen besprochen, um mögliche Unklarheiten und Verständnisprobleme zu beseitigen.

C.1. MMF/ARH

Die Abteilung *Empirical Software Engineering* des *Institute of Software Engineering* der Universität Stuttgart arbeitet seit einigen Jahren an einem Microservices Migration Framework und damit verbunden auch an der Umsetzung dessen in einem webbasierten Tool, dem ARH. Dieses soll Entwickler dabei anleiten und unterstützen, Migrationen von Monolithen zu Microservices durchzuführen. Das Framework unterteilt die Durchführung in drei Phasen:

- **Phase 1: System Comprehension:** In Phase 1 wird das Verständnis des Systems, das migriert werden soll, angestrebt. Mit den Stakeholdern werden dabei strategische Ziele für Produkt und Unternehmen definiert, sowie Qualitätsattribut (QA) und gewünschte Szenarien identifiziert. Diese QAs und Szenarien sind besonders wichtig für die späteren Phasen, in denen sie als Suchkriterium verwendet werden. Außerdem sollen sich dadurch mögliche Treiber für die neue Microservices-Architektur herauskristallisieren. Architekten sollen dann mithilfe dieser Informationen eine Entscheidung für oder gegen die Migration zu Microservices fällen.
- **Phase 2: Strategy Definition:** Die hauptsächliche Aufgabe in dieser Phase ist die Entscheidung, welche Migrationsstrategie benutzt werden soll, um das System zu modernisieren. Dafür kann zwischen (momentan) 115 verschiedenen Methoden gewählt werden, die aus akademischen Publikationen stammen. Das Tool kann Entwickler dabei unterstützen, indem es basierend auf den in Phase 1 eingegebenen Qualitätsattributen Methoden empfiehlt. Einige dieser Methoden beinhalten die Nutzung von Tools, welche auch gesondert durchsucht werden können. Die verschiedenen Migrationsmethoden und Tools können von Admins importiert, exportiert und bearbeitet werden, wodurch die Erweiterung in zukünftigen Arbeiten vereinfacht möglich ist.

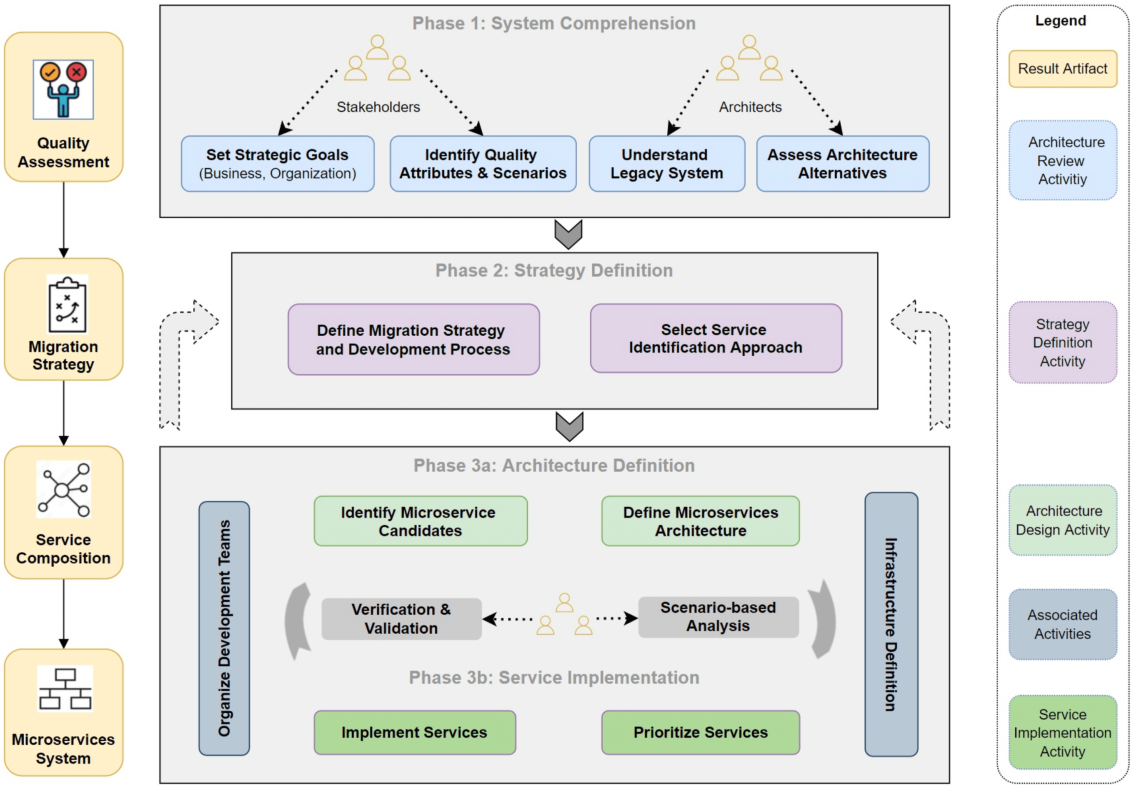

Abbildung C.1.: Übersicht über das MMF des ESE.

- **Phase 3a: Architecture Definition:** Phase 3 ist in zwei Teile aufgeteilt. Im ersten Abschnitt wird die neue Architektur definiert. Hier wird die in Phase 2 gewählte Migrationsmethode durchgeführt, wodurch eine Aufteilung in Microservices entsteht. Das Tool unterstützt Entwickler in dieser Phase durch das Vorschlagen von geeigneten Best Practices und Patterns, deren Implementierung potentiell wertvoll für die in Phase 1 definierten QAs wäre.
- **Phase 3b: Service Implementation:** Phase 3a und 3b sind eng verbunden, denn durch Erkenntnisse in der Implementierung kann die Planung häufig noch mehrmals überarbeitet werden und dadurch eine neue Implementierung begonnen werden. In Phase 3b startet dann ein Zyklus von Implementierungen der in Phase 3a definierten Services. Bei Implementierung selbst kann das Tool Entwickler nicht unterstützen. Ist diese Phase abgeschlossen, ist eine erste Version des migrierten Systems fertig.

C.2. Migrationsverfahren

In dieser Thesis wurde in Phase 2 des MMF eine Suche nach Migrationsverfahren durchgeführt, bei der im Voraus gezielt auf *jadice flow* abgestimmte Filter konfiguriert wurden. Dazu gehören die in Phase 1 des Frameworks erhobenen Szenarien/QAs sowie zusätzlich definierte Filter. Insgesamt acht dieser Suchergebnisse wurden im Rahmen dieser Thesis analysiert und auf ihre Eignung für

jadice flow überprüft. Davon wurden zwei als potentiell für *jadice flow* geeignet bewertet. In den Interviews sollen Sie ebenfalls eine Einschätzung über die Eignung dieser abgeben. Deshalb wird deren Funktionsweise im Folgenden beschrieben.

C.2.1. Migrationsverfahren 1

Dieses Verfahren besteht aus drei übergeordneten Schritten (Übersicht in Abbildung C.2).

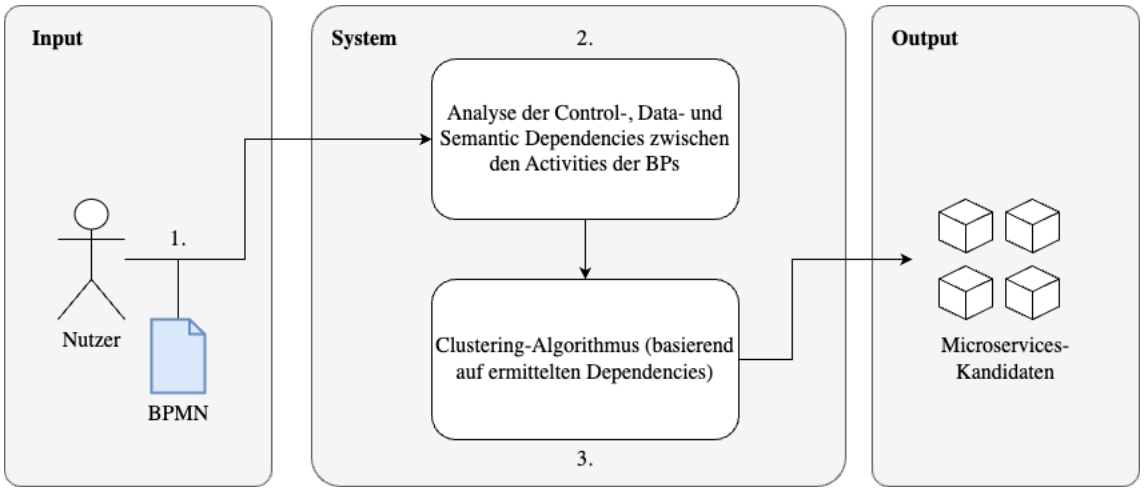

Abbildung C.2.: Funktionsweise des ersten Migrationsverfahrens

Im ersten Schritt werden die Eingaben für die Methode in Form von Business Processes (BPs) gesammelt und logisch verbunden, beispielsweise in Form eines BPMN-Diagramms (Beispielhafte Eingabe für *jadice flow* in Abbildung C.3).

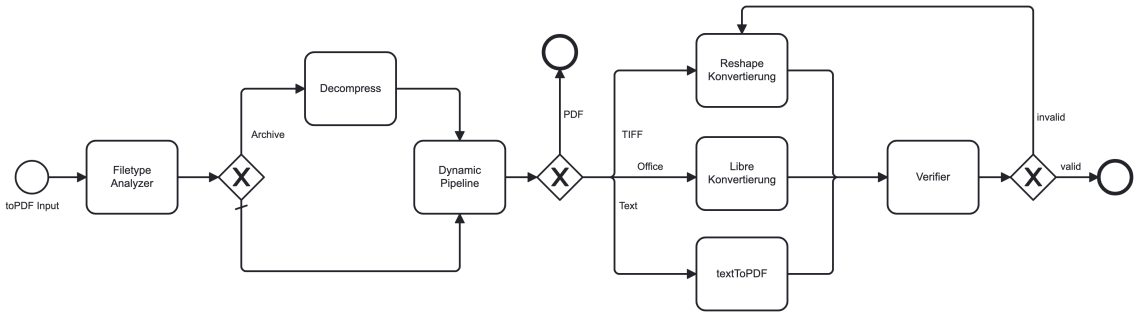

Abbildung C.3.: Mögliche Eingabe für Migrationsverfahren 1

Für die Bestandteile dieser Eingabe (genannt Aktivitäten) werden im zweiten Schritt Abhängigkeiten in drei verschiedenen Formen analysiert: *control dependencies*, *data dependencies* und *semantic dependencies*.

- Eine *control dependency* beschreibt eine Abhängigkeit zweier Aktivitäten im Kontrollfluss, also dass eine der Aktivitäten mit hoher Wahrscheinlichkeit auf die andere folgt.

- Eine *data dependency* bildet einen Datenfluss zwischen den Ein- oder Ausgaben zweier Aktivitäten ab.
- Eine *semantic dependency* gibt die Verwandtschaft des Zwecks zweier Aktivitäten an. Die Messung solcher Abhängigkeiten kann anhand der Ähnlichkeit der Namen der Aktivitäten erfolgen.

Schließlich wird die *Machine Learning*-Technik *Collaborative Clustering* verwendet, um die Menge von Aktivitäten in Gruppen (genannt *Cluster*) zu unterteilen. Dabei werden die beschriebenen Dependencies verwendet, um die Gruppierung anhand dieser zu optimieren. Das bedeutet, dass Abhängigkeiten zwischen Aktivitäten des gleichen Clusters stark sein und clusterübergreifend geringer sein sollten. Entstandene Gruppen sind Vorschläge für eine Aufteilung in Microservices.

C.2.2. Migrationsverfahren 2

In diesem Verfahren wird das Tool *Microservice Backlog (MB)* vorgestellt, das ein semiautomatisches Modell zur Bewertung der Granularität einer MSA anbietet (Übersicht in Abbildung C.4).

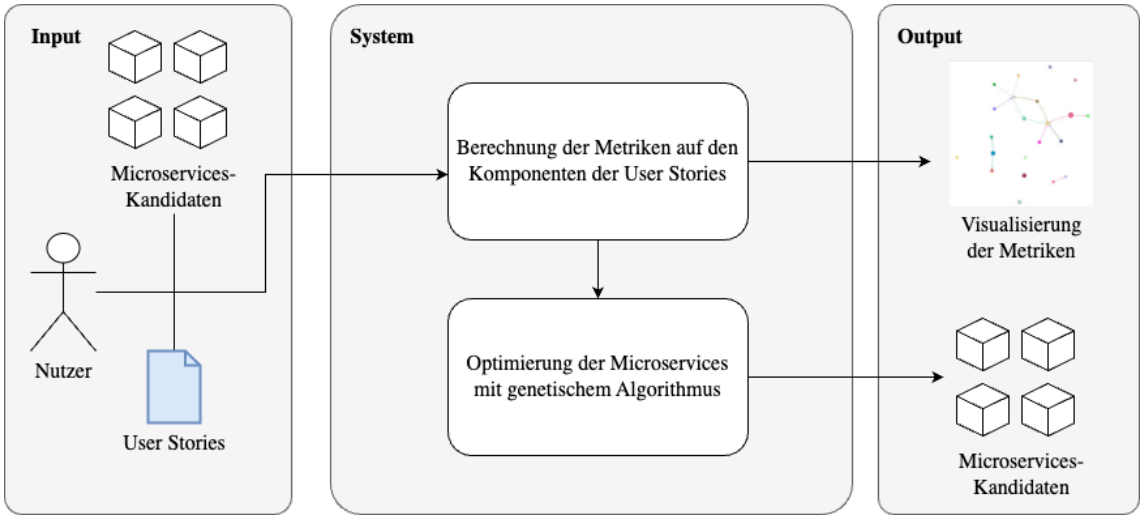

Abbildung C.4.: Funktionsweise des zweiten Migrationsverfahrens

Als Eingabe werden *User Stories* aus dem *Product Backlog* oder der Release-Planung verwendet. Diese haben im Vergleich zum ersten Verfahren eine etwas andere Form, aber die Eingaben sind trotzdem mit denen des ersten Verfahrens vergleichbar; mit den User Stories werden auch übliche Workflows beschrieben. Deswegen wird hier keine beispielhafte Eingabe vorgestellt. Außerdem kann MB die Metriken *Coupling*, *Kohäsion*, *Granularität*, *semantische Ähnlichkeit* und *Komplexität* für das Gesamtsystem und für einzelne Microservices berechnen. Diese können ebenfalls visualisiert und so übersichtlich betrachtet werden. Anhand dieser Metriken könnte eine manuelle Überarbeitung der Serviceaufteilung angestoßen werden oder der genetische Algorithmus genutzt werden, der ebenfalls in dem Verfahren vorgestellt wird, um eine Verbesserung der Metriken zu erreichen.

C.3. Best Practices und Patterns

Neben der Wahl von Migrationsmethoden unterstützt der ARH Entwickler außerdem bei der Wahl zu implementierender Best Practices und Patterns auf Basis der in Phase 1 konfigurierten QAs. Diese waren *Scalability*, *Maintainability*, *Performance* und *Portability*. Auf deren Basis wurden für *jadice flow* die in Tabelle C.1 sichtbaren Best Practices und Patterns vorgeschlagen. Auch über diese wird von Ihnen eine Bewertung im Interview abgefragt.

Pattern	**Matches**
Service Registry	11/17
CQRS	10/17
Pipes and Filters	10/17
Asynchronous Messaging	9/17
Gateway Offloading	9/17
Load Balancer	9/17
Ambassador	7/17
Backend for Frontend	7/17
Container	7/17
Database is the Service	7/17
Scalable Store	7/17
API Gateway	6/17
Circuit Breaker	6/17
Deploy clusters and orchestrate containers	6/17
Edge Server	6/17
External Load Balancer	6/17
Gateway Aggregation	6/17
Internal Load Balancer	6/17
Local Database Proxy	6/17
Local Sharding-based Router	6/17

Best Practices	**Matches**
Isolated State	11/17
Dynamic Scheduling	9/17
Infrastructure Abstraction	8/17
Use Infrastructure as Code	8/17
Acyclic Calls	7/17
Automated Infrastructure	6/17
Automated Monitoring	6/17
Operation Outsourcing	6/17
Cloud Vendor Abstraction	5/17
Immutable Artifacts	5/17
Standardized Deployment Unit	5/17
Built-in Autoscaling	4/17
Persistent Communication	4/17
API-based Communication	3/17
Automated Restarts	3/17
Autonomous Fault Handling	3/17
Coarse-Grained Microservices	3/17
Communication Indirection	3/17
Guarded Ingress	3/17
Loose Coupling	3/17

Tabelle C.1.: Ergebnisse der Suche mit ARH nach Best Practices und Patterns, nach Matches sortiert. Die Matches entsprechen den im ARH angezeigten Matches.

D. Feldnotizen

Tabelle D.1.: F01_P2_11_20_2023_FilterAuswahl

<table>
<tr><th>Eintragstyp</th><th>Inhalt</th></tr>
<tr><td>Feldnotitz Nr.</td><td>1</td></tr>
<tr><td>Datum, Uhrzeit</td><td>20.11.2023, 10:00-10:30</td></tr>
<tr><td>Ort</td><td>Holzgerlingen, Büro levigo</td></tr>
<tr><td>Beteiligte Personen</td><td>Axel Herrmann, Product Owner</td></tr>
<tr><td>Phase & Schritt des MMF</td><td>2, Einstellen der Filter des ARH</td></tr>
<tr><td>Aktionen und Entscheidungen</td><td>Es wurden Filter für die Suche nach Refactoring-Methoden mit dem ARH eingestellt. Dabei wurden die SPs als am wichtigsten betrachtet, weswegen dort einige Eigenschaften inkludiert wurden. Bei den anderen Kategorien wurde versucht, nur die wichtigsten Filter zu verwenden, da mit zu vielen Filtern eine geringere Aussagekraft dieser vermutet wird. Vor allem Process Preferences und Usability Preferences, also Eigenschaften, die die interne Funktionsweise der Methode betreffen, wurden als relativ unwichtig und vor allem auch unabhängig vom betreffenden System eingestuft und demnach nur ein einziger dieser Filter verwendet. Am wichtigsten neben den SPs wurden Input Preferences und Output Preferences eingestuft, da diese ebenfalls stark vom System abhängen und Ausgangslage und gewünschtes Ergebnis betreffen. Daraus resultieren folgenden Filter:
<table>
<tr><th>Kategorie</th><th>Filter</th></tr>
<tr><td>Quality Preferences, System Properties</td><td>Autonomy, Complexity, Granularity, Isolation, Technology Heterogenity</td></tr>
<tr><td>Input preferences, Domain Artifacts</td><td>Human Expertise, Version Control System</td></tr>
<tr><td>Input preferences, Executables</td><td>API/Interface, Source Code (Java)</td></tr>
<tr><td>Process preferences, Process Strategy</td><td>Refactor</td></tr>
<tr><td>Output preferences, Representation</td><td>List of services, Splitting recommendations</td></tr>
<tr><td>Usability Preferences</td><td>-</td></tr>
</table>
</td></tr>
<tr><td colspan="2">Wird auf der nächsten Seite fortgeführt</td></tr>
</table>

Tabelle D.1 – von der vorherigen Seite weitergeführt

Eintragstyp	**Inhalt**
Kommentare	Die Filter sind noch nicht zwingend final. Nach einer Durchsicht der Ergebnisse der Refactoring-Methoden kann eine weitere Anpassung der Filter erfolgen.
Gut gelaufen	Die Auswahl der Filter mit dem ARH ist simpel, bietet gute UX.
Schlecht gelaufen	Nicht alle Filterkriterien sind direkt verständlich und beinhalten oft noch keine Beschreibung. Beispiel: *Validation Methods* könnte auch so verstanden werden, dass es sich auf die Validierung des Systems nach Migration bezieht, bezieht sich aber darauf, wie die jeweiligen Methoden bereits validiert wurden.
Empfindungen	Bei manchen Filterkriterien unsicher bezüglich der Bedeutung, legt sich allerdings nach Diskussion darüber. Außerdem etwas unsicher, wie genau die Feldnotizen funktionieren werden, da es die erste Anwendung dieser ist.

Tabelle D.2.: F02_P2_11_22_2023_FilterAuswahl

Eintragstyp	**Inhalt**
Feldnotitz Nr.	2
Datum, Uhrzeit	22.11.2023, 14:00-14:10
Ort	Zuhause, Online-Meeting
Beteiligte Personen	Axel Herrmann, universitärer Betreuer
Phase & Schritt des MMF	2, Einstellen der Filter des ARH
Aktionen und Entscheidungen	Die Filter, die in der ersten Feldnotiz angelegt wurden, wurden hier besprochen und im Rahmen dessen der Filter *Source Code (No specification)* hinzugefügt. Des Weiteren wurde beschlossen, dass in einem erneuten Treffen mit dem PO die Filter in zwei Prioritäten unterteilt werden sollen.
Kommentare	
Gut gelaufen	Bis auf die eine Änderung wurden die Filter und Entscheidungen dafür für sinnvoll befunden.
Schlecht gelaufen	
Empfindungen	Zuversichtlich durch die Bestätigung durch den Betreuer.

Tabelle D.3.: F03_P2_11_27_2023_FilterPrio

Eintragstyp	**Inhalt**
Feldnotitz Nr.	3
Datum, Uhrzeit	27.11.2023, 11:00-11:10
	Wird auf der nächsten Seite fortgeführt

Tabelle D.3 – von der vorherigen Seite weitergeführt

<table>
<tr><th>Eintragstyp</th><th>Inhalt</th></tr>
<tr><td>Ort</td><td>Holzgerlingen, Büro levigo</td></tr>
<tr><td>Beteiligte Personen</td><td>Axel Herrmann, Product Owner</td></tr>
<tr><td>Phase & Schritt des MMF</td><td>2, Einstellen der Filter des ARH</td></tr>
<tr><td>Aktionen und Entscheidungen</td><td>Die Einstellung der Filter für die Suche nach Refactoring-Methoden mit dem ARH wurde fortgesetzt. Dabei sollten Filter in zwei Prioritäten entstehen, sodass später eine Suche mit allen Filtern und eine mit nur den wichtigsten Filtern durchgeführt werden kann. Größtenteils wurden die bereits ausgewählten Filter in die zwei Prioritäten unterteilt, aber in zwei Fällen auch neue, vorher nicht ausgewählte Filter zur geringeren Priorität hinzugefügt. Die neuen Filter sind:
<table>
<tr><th>Kategorie</th><th>Filter Prio 1</th><th>Filter Prio 2</th></tr>
<tr><td>Quality Preferences, System Properties</td><td>Autonomy, Granularity, Technology Heterogenity</td><td>Complexity, Isolation</td></tr>
<tr><td>Input Preferences, Domain Artifacts</td><td>Human Expertise</td><td>Version Control System</td></tr>
<tr><td>Input Preferences, Runtime Artifacts</td><td></td><td>Log Traces</td></tr>
<tr><td>Input Preferences, Executables</td><td>API/Interface, Source Code (No specification)</td><td>Source Code (Java)</td></tr>
<tr><td>Process Preferences, Process Strategy</td><td>Refactor</td><td></td></tr>
<tr><td>Output Preferences, Representation</td><td>List of services, Splitting recommendations</td><td>Guideline/ Workflow</td></tr>
<tr><td>Usability Preferences</td><td>-</td><td></td></tr>
</table>
</td></tr>
<tr><td>Kommentare</td><td>Die Filter sind vorerst finalisiert. Nach einer Durchsicht der Ergebnisse der Refactoring-Methoden könnten weitere Anpassung der Filter erfolgen.</td></tr>
<tr><td>Gut gelaufen</td><td>Es konnte sich bei jedem Filter schnell geeinigt werden und die Aufgabe war schnell erledigt.</td></tr>
<tr><td>Schlecht gelaufen</td><td></td></tr>
<tr><td>Empfindungen</td><td>Sicherheit bei der Aufgabe, da das Vorgehen nun klarer ist und alles mit dem Betreuer besprochen wurde. Außerdem mehr Sicherheit mit Dokumentation in Feldnotizen.</td></tr>
</table>

Tabelle D.4.: F04_P2_12_13_2023_Ergebnisbetrachtung_1

Eintragstyp	**Inhalt**
Feldnotitz Nr.	4
Datum, Uhrzeit	13.12.2023, 13:00-15:00
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des ersten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Habibullah et al. [HLT18] wurde durchgelesen und zusammengefasst.
Kommentare	Nach der Betrachtung der anderen Ergebnisse kann die Arbeit erneut und genauer betrachtet werden. Eine Bewertung hinsichtlich des Nutzens für *jadice flow* erfolgt später, wenn mit anderen Ergebnissen verglichen werden kann. Verbesserungsvorschlag: Die Methode scheint keinen spezifischen SIA zu enthalten. Das könnte in die Filter des ARH aufgenommen werden.
Gut gelaufen	
Schlecht gelaufen	Erster Eindruck: Die Methode wirkt sehr oberflächlich und wenig spezifisch und scheint sich daher weniger zu eignen.
Empfindungen	Frustration: Vom ARH mit Abstand am besten bewertetes Ergebnis unpassend.

Tabelle D.5.: F05_P2_12_20_2023_Ergebnisbetrachtung_2

Eintragstyp	**Inhalt**
Feldnotitz Nr.	5
Datum, Uhrzeit	20.12.2023, 15:00-17:30
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des zweiten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Daoud et al. [DEF+21] wurde durchgelesen und zusammengefasst.
Kommentare	Nach der Betrachtung der anderen Ergebnisse kann die Arbeit erneut und genauer betrachtet werden. Eine Bewertung hinsichtlich des Nutzens für *jadice flow* erfolgt später, wenn mit anderen Ergebnissen verglichen werden kann.
Gut gelaufen	Erster Eindruck: Die Methode wirkt sehr viel spezifischer als die erste und scheint relativ gut zu passen.
	Wird auf der nächsten Seite fortgeführt

Tabelle D.5 – von der vorherigen Seite weitergeführt

Eintragstyp	**Inhalt**
Schlecht gelaufen	
Empfindungen	Froh, einen relativ gut passenden Ansatz gefunden zu haben, nachdem der erste nicht gut war.

Tabelle D.6.: F06_P2_12_20_2023_Ergebnisbetrachtung_3

Eintragstyp	**Inhalt**
Feldnotitz Nr.	6
Datum, Uhrzeit	20.12.2023, 17:30-18:30
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des dritten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Vera-Rivera et al. [VPAG21] wurde durchgelesen und zusammengefasst.
Kommentare	Nach der Betrachtung der anderen Ergebnisse kann die Arbeit erneut und genauer betrachtet werden. Eine Bewertung hinsichtlich des Nutzens für *jadice flow* erfolgt später, wenn mit anderen Ergebnissen verglichen werden kann.
Gut gelaufen	Erster Eindruck: Die Methode ist bisher am ausführlichsten. Im Vergleich zur 2. berücksichtigt sie mehr Metriken und die Eingabe einer vorhandenen ARH ist auch passend. Die Visualisierung der Architektur mit Anzeige der Metriken wäre sehr angenehm.
Schlecht gelaufen	
Empfindungen	Froh, einen weiteren relativ gut passenden Ansatz gefunden zu haben.

Tabelle D.7.: F07_P2_12_21_2023_Ergebnisbetrachtung_4

Eintragstyp	**Inhalt**
Feldnotitz Nr.	7
Datum, Uhrzeit	21.12.2023, 08:30-09:30
Ort	Zuhause, Online-Meeting
Beteiligte Personen	Axel Herrmann, Product Owner
Phase & Schritt des MMF	2, Betrachtung der ersten drei Suchergebnisse
	Wird auf der nächsten Seite fortgeführt

Tabelle D.7 – von der vorherigen Seite weitergeführt

Eintragstyp	Inhalt
Aktionen und Entscheidungen	Die Arbeiten von Habibullah et al. [HLT18], Daoud et al. [DEF+21] und Vera-Rivera et al. [VPAG21] wurden dem Product Owner zusammengefasst und gemeinsam die Möglichkeit der Anwendung auf *jadice flow* diskutiert. Dabei wurde eine Ordnung der Ergebnisse hinsichtlich ihrer Nützlichkeit für diese Fallstudie vorgenommen, wobei Vera-Rivera et al. [VPAG21] mit Platz 1, Daoud et al. [DEF+21] mit Platz 2 und Daoud et al. [DEF+21] mit Platz 3 abschneidet.
Kommentare	Die aufgestellte Platzierung ist in Anbetracht dessen, dass erst 3 der 6 Ergebnisse betrachtet wurden, noch nicht vollständig.
Gut gelaufen	Alle der Methoden beinhalten eine nützliche Übersichts-Grafik, wodurch das Beschreiben der Methoden gut funktioniert. Bei der Platzierung der Methoden kann einfach und unkopliziert Konsens erreicht werden. Der erste Eindruck des Product Owners stimmt mit der des Autors überein.
Schlecht gelaufen	
Empfindungen	

Tabelle D.8.: F08_P2_01_10_2024_Ergebnisbetrachtung_5

Eintragstyp	Inhalt
Feldnotitz Nr.	8
Datum, Uhrzeit	10.01.2024, 10:00-10:30
Ort	Büro levigo
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des vierten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Freitas et al. [FFC21] wurde durchgelesen und zusammengefasst.
Kommentare	Die Betrachtung dieser Arbeit ist im Gegensatz zu den anderen Ergebnissen vermutlich final, da die Verwendung dieser Methode schnell ausgeschlossen werden konnte.
Gut gelaufen	
Schlecht gelaufen	Erster Eindruck: Die Ziele der Methode passen nicht zu unseren und sie ist technisch vermutlich auch nicht anwendbar.
Empfindungen	

Tabelle D.9.: F09_P2_01_10_2024_Ergebnisbetrachtung_6

Eintragstyp	**Inhalt**
Feldnotitz Nr.	9
Datum, Uhrzeit	10.01.2024, 11:00-11:30
Ort	Büro levigo
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des fünften Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Matias et al. [MCF+20] wurde durchgelesen und zusammengefasst.
Kommentare	Nach der Betrachtung der anderen Ergebnisse kann die Arbeit erneut und genauer betrachtet werden.
Gut gelaufen	Erster Eindruck: Das Tool kann nicht verwendet werden, die Methode kommt infrage.
Schlecht gelaufen	Erster Eindruck: Verwendung bringt vermutlich erhöhten Aufwand mit sich, da das fehlende Tool manuell nachgeahmt werden muss.
Empfindungen	

Tabelle D.10.: F10_P2_01_11_2024_Ergebnisbetrachtung_7

Eintragstyp	**Inhalt**
Feldnotitz Nr.	10
Datum, Uhrzeit	11.01.2024, 11:00-11:30
Ort	Büro levigo
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des sechsten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von De Alwis et al. [DBFP21] wurde durchgelesen und zusammengefasst.
Kommentare	Nach der Absprache mit dem PO kann die Arbeit erneut und genauer betrachtet werden.
Gut gelaufen	
Schlecht gelaufen	Erster Eindruck: Der Kontext von IIoT passt nicht zu *jadice flow*, ob die Methode trotzdem sinnvoll sein könnte, bleibt offen.
Empfindungen	

Tabelle D.11.: F11_P2_01_11_2024_Ergebnisbetrachtung_8

Eintragstyp	**Inhalt**
Feldnotitz Nr.	11
Datum, Uhrzeit	11.01.2024, 14:00-14:30
Ort	Büro levigo
Beteiligte Personen	Axel Herrmann, Product Owner
Phase & Schritt des MMF	2, Vergleich der Suchergebnisse
Aktionen und Entscheidungen	Die Arbeiten von Freitas et al. [FFC21], Matias et al. [MCF+20] und De Alwis et al. [DBFP21] wurden dem Product Owner zusammengefasst und gemeinsam die Möglichkeit der Anwendung auf *jadice flow* diskutiert. Dabei wurde eine Ordnung aller Ergebnisse hinsichtlich ihrer Nützlichkeit für diese Fallstudie vorgenommen, wobei immer noch Vera-Rivera et al. [VPAG21] mit Platz 1 und Daoud et al. [DEF+21] mit Platz 2 abschneiden.
Kommentare	
Gut gelaufen	Der erste Eindruck des Product Owners stimmt mit der des Autors überein.
Schlecht gelaufen	
Empfindungen	

Tabelle D.12.: F12_P2_01_17_2024_Ergebnisbetrachtung_9

Eintragstyp	**Inhalt**
Feldnotitz Nr.	12
Datum, Uhrzeit	17.01.2024, 11:00-11:30
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des siebten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Assunção et al. [ACC+22] wurde durchgelesen und zusammengefasst.
Kommentare	
Gut gelaufen	Erster Eindruck: Einsatz wäre prinzipiell möglich, aber
Schlecht gelaufen	Funktionen als atomare Einheiten scheinen allerdings sehr klein-granular und möglicherweise zu aufwendig.
Empfindungen	

Tabelle D.13.: F13_P2_01_18_2024_Ergebnisbetrachtung_10

Eintragstyp	**Inhalt**
Feldnotitz Nr.	13
Datum, Uhrzeit	18.01.2024, 10:30-11:00
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	2, Betrachtung des achten Suchergebnisses
Aktionen und Entscheidungen	Die Arbeit von Selmadji et al. [SSB+20] wurde durchgelesen und zusammengefasst.
Kommentare	
Gut gelaufen	Erster Eindruck: Einsatz wäre prinzipiell möglich, aber
Schlecht gelaufen	Klassen als atomare Einheiten scheinen allerdings sehr klein-granular und möglicherweise zu aufwendig.
Empfindungen	

Tabelle D.14.: F14_P2_01_18_2024_Ergebnisbetrachtung_11

Eintragstyp	**Inhalt**
Feldnotitz Nr.	14
Datum, Uhrzeit	18.01.2024, 11:00-11:30
Ort	Zuhause, Online-Meeting
Beteiligte Personen	Axel Herrmann, Product Owner
Phase & Schritt des MMF	2, Vergleich der Suchergebnisse
Aktionen und Entscheidungen	Die Arbeiten von Assunção et al. [ACC+22] und Selmadji et al. [SSB+20] wurden dem Product Owner zusammengefasst und gemeinsam die Möglichkeit der Anwendung auf *jadice flow* diskutiert. Da nun voraussichtlich alle potentiellen Methoden betrachtet wurden, wurde diskutiert, welche der Verfahren als erstes fokussiert werden sollte. Die ersten zwei Plätze belegen dabei Vera-Rivera et al. [VPAG21] und Daoud et al. [DEF+21].
Kommentare	
Gut gelaufen	Schnelles Übereinstimmen bei der Bewertung und Sortierung der Verfahren
Schlecht gelaufen	Nur oberfläches Wissen über die Verfahren reicht nicht aus, um final zu entscheiden, welcher Anstatz verwendet wird. Einige Verfahren gelten als semi-automatisch, allerdings sind keine der in den Verfahren beschriebenen Tools zu finden, was den zeitlichen Aufwand bei Einsatz der Verfahren schwerer einschätzbar macht.
Wird auf der nächsten Seite fortgeführt	

Tabelle D.14 – von der vorherigen Seite weitergeführt

Eintragstyp	**Inhalt**
Empfindungen	Unsicherheit, welches der Verfahren wirklich im zeitlichen Rahmen realistisch anwendbar ist

Tabelle D.15.: F15_P3_01_30_2024_Methodenanwendung_1

Eintragstyp	**Inhalt**
Feldnotitz Nr.	15
Datum, Uhrzeit	30.01.2024, 11:00-17:40
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	3, Anwendung der ersten Migrationsmethode
Aktionen und Entscheidungen	Das Ergebnis 3 [VPAG21] wurde in vorherigen Schritten als favorisierte Methode ausgewählt. Der Artikel dazu wurde genauer betrachtet und das beschriebene Tool Microservice Backlog (MB) gesucht. Es konnte keine Veröffentlichung des Tools gefunden werden, lediglich ein GitHub Projekt der Autoren, das den richtigen Namen trug. Das Projekt scheint veraltet zu sein, es dauerte einige Zeit, es überhaupt zum Laufen zu bringen. Auch nach erfolgreichem Start von MB in einem Docker Container konnte das Webtool nicht verwendet werden, da abgesehen von der Startseite jede Sub-Seite Fehler auslöste und nicht aufgerufen werden konnte. Da der manuelle Aufwand der Umsetzung dieser Methode als zu hoch bewertet wurde, wurde dieses Verfahren letztendlich verworfen.
Kommentare	Nach diesem favorisiertem Verfahren besteht noch bei einem weiteren Verfahren Hoffnung, dass es verwendet werden könnte. Dieses wird als nächstes betrachtet.
Gut gelaufen	
Schlecht gelaufen	Das beschriebene Tool ist nicht öffentlich verlinkt, das gefundene GitHub Projekt nicht benutzbar und die Methode manuell in dieser Thesis nicht umsetzbar.
Empfindungen	Frustration über das Fehlschlagen dieses Verfahrens.

Tabelle D.16.: F16_P3_02_07_2024_Methodenanwendung_2

Eintragstyp	**Inhalt**
Feldnotitz Nr.	16
Datum, Uhrzeit	07.02.2024, 10:00-14:00
Ort	Zuhause
	Wird auf der nächsten Seite fortgeführt

Tabelle D.16 – von der vorherigen Seite weitergeführt

Eintragstyp	Inhalt
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	3, Anwendung der zweiten Migrationsmethode
Aktionen und Entscheidungen	Der Algorithmus *cHAC* nach Ergebnis 2 [DEF+21] wurde im Detail betrachtet und nach Top-Down Methode angefangen umzusetzen. Die Implementierung wurde in Java vorgenommen. Für gewisse Details des Pseudo-Codes wurden Java-spezifische Konstrukte verwendet statt der im Algorithmus angegebenen. Beispielsweise wurde die Synchronisationsvariable @SIC des Algorithmus durch eine CyclicBarrier[1] umgesetzt.
Kommentare	
Gut gelaufen	Obwohl der Algorithmus relativ komplex ist, wurde er auf einer abstrakten Ebene verstanden und auf dieser auch umgesetzt.
Schlecht gelaufen	Der Algorithmus ist relativ kompliziert und die Beschreibung dazu fällt kurz aus. Viele Teile mussten sehr oft gelesen werden, bevor Verständnis erlangt werden konnte.
Empfindungen	Unsicher, ob der Algorithmus umgesetzt werden kann.

Tabelle D.17.: F17_P3_02_08_2024_Methodenanwendung_3

Eintragstyp	Inhalt
Feldnotitz Nr.	17
Datum, Uhrzeit	08.02.2024, 10:45-17:00
Ort	Zuhause
Beteiligte Personen	Axel Herrmann
Phase & Schritt des MMF	3, Anwendung der zweiten Migrationsmethode
Aktionen und Entscheidungen	Der Algorithmus *cHAC* nach Ergebnis 2 [DEF+21] wurde weiter versucht umzusetzen. Die grobe Struktur war duch den vorherigen Tag bereits vorhanden. Nun sollten die einzelnen Schritte, Funktionen und Formeln des Artikels umgesetzt werden. Dabei konnte kein Erfolg verzeichnet werden; einige Funktionen waren nicht bis ins Detail beschrieben und ließen Freiraum, einige Formeln definierten Eingabevariablen nicht genau und in einer Formel wurde auch ein Schreibfehler vermutet. Dadurch dass die Formeln lediglich genannt, nicht aber begründet oder beschrieben wurden, konnten Unklarheiten nicht geklärt und der Algorithmus so nicht umgesetzt werden.
	Wird auf der nächsten Seite fortgeführt

[1] https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CyclicBarrier.html

Tabelle D.17 – von der vorherigen Seite weitergeführt

Eintragstyp	**Inhalt**
Kommentare	
Gut gelaufen	
Schlecht gelaufen	Kein Erfolg: Arbeit am Algorithmus muss eingestellt werden.
Empfindungen	Frustration: Obwohl 2 Suchergebnisse vielversprechend klangen, konnte schlussendlich keines davon verwendet werden.

Tabelle D.18.: F18_P3_02_09_2024_Besprechung_Methodenanwendung

Eintragstyp	**Inhalt**
Feldnotitz Nr.	18
Datum, Uhrzeit	09.02.2024, 14.00-14:15
Ort	Zuhause, Online-Meeting
Beteiligte Personen	Axel Herrmann, Product Owner
Phase & Schritt des MMF	3, Besprechung Anwendung Migrationsverfahren
Aktionen und Entscheidungen	Die Ergebnisse der erfolgslosen Anwendung der Migrationsmethoden wurden besprochen und gemeinsam beschlossen, die Arbeit des Refactorings deswegen einzustellen. Die anderen Suchergebnisse des ARH wurden bereits in vorherigen Schritten als wesentlich weniger passend bewertet und deswegen und aus Zeitgründen keine weitere Anwendung in Betracht gezogen.
Kommentare	
Gut gelaufen	
Schlecht gelaufen	Kein Erfolg in dieser Phase: Arbeit am Refactoring muss eingestellt werden.
Empfindungen	Frustration: Obwohl 2 Suchergebnisse vielversprechend klangen, konnte schlussendlich keines davon verwendet werden.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Druck-Exemplaren überein.

Ort, Datum, Unterschrift