Fakultät für Informatik und Mathematik 07

Bachelorarbeit

über das Thema

Microservices und technologische Heterogenität
Entwicklung einer sprachunabhängigen Microservice Framework Evaluationsmethode

Autor: René Zarwel
zarwel@hm.edu

Prüfer: Prof. Dr. Hammerschall

Abgabedatum: XX.XX.XX

I Kurzfassung

Insert Abstract Here

Abstract

Insert Abstract Here

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Listing-Verzeichnis

VI Abkürzungsverzeichnis

SEI Software Engineering Institute
SAAM Software Architecture Analysis Method
ATAM Architecture Tradeoff Analysis Method
SAEM Software Architecture Evaluation Model
GQM Goal Question Metrik
MFEM Microservice Framework Evaluation Method

1 Einleitung

TODO 1: Einführung + Motivation für Erstellung Methode

2 Microservices

TODO 2: Kurz Microservices Basics + Motivation einfügen und beschreiben

Die Auswahl des richtigen Frameworks für einen einzelnen Service hängt von vielen Faktoren ab. Damit neben den Kundenanforderungen nicht Anforderungen von der Microservice Architektur vergessen und objektiv in den Auswahlprozess mit einbezogen werden, sollte es eine Methode zur Bewertung des Frameworks geben. Dabei kann die durch das Framework vorgegebene Architektur und die bereits enthaltenen Funktionen bewertet werden. So kann, schon während der Entwurfsphase, eine fundierte Aussage über die zu erwartenden Risiken und Trade-Offs getroffen werden. Sollte z.B. eine wesentliche Funktion, wie eine Authentifizierung am Service, fehlen, müsste diese selbstständig entwickelt werden. Die kann, je nach Architektur, sehr aufwendig sein. Damit verbunden wäre ein hoher Zeitaufwand, der die Projektkosten immens steigern würde. Um eine passende Bewertungsmethode zu finden, können bereits etablierte Methoden zur Architekturbewertung herangezogen werden. Der Vorteil hierbei ist, dass diese bereits getestet, weiterentwickelt und in vielen Unternehmen verwendet werden. So ist der Aufwand für die Integration der neuen Methode sehr gering, da sie in bereits bestehende Bewertungsprozesse integriert werden kann. In diesem Zusammenhang stellt sich die grundsätzliche Frage, warum Architekturen in der professionellen Softwareentwicklung bewertet werden? Und ob es dafür bereits etablierte Methoden gibt?

3 Qualitätsbewertung von Softwarearchitektur

> „*You cannot control what you cannot measure*“
> - Tom DeMarco (Controlling Software Projects, 1982)

Die Aussage beschreibt sehr gut, dass ein IT-Projekt nur dann direkt auf die Projektziele zusteuern kann, wenn Messwerte erhoben werden. Das erlaubt einen Vergleich mit den bereits erreichten Zielen und trifft eine Aussage über den Projektstatus. Sollte der Status an gewissen Punkten zu stark von den Anforderungen abweichen, können rechtzeitig korrigierende Maßnahmen eingeleitet werden[7]. Gerade bei einem so entscheidendem Punkt, wie der Architektur, ist es wichtig frühzeitig Risiken und Qualitätsabweichungen festzustellen. Die Architektur ist ein sehr kritischer und wesentlicher Bestandteil im Entwicklungsprozess von einer Software. Ihrer Beschaffenheit nach, kann sie nur schwer und mit hohen Kosten verändert werden. Durch Zeit- und Kostendruck wird dies leider in der Praxis häufig erst in späten Entwicklungsphasen durchgeführt. In einem sogenannten

TODO 3: Ausschweifender einleiten, um keinen zu großen Cut zu haben.

„Audit“ oder „Review“ werden bereits produktiv laufende Systeme bewertet[7]. Sollten sich hier starke Abweichungen gegenüber den Anforderungen zeigen, kann sich dies zu einem großen Problem avancieren, hohe Kosten verursachen und Kunden verärgern. Aus diesem Grund ist die Qualität einer Architektur sehr entscheidend und sollte stichhaltig bewiesen sein. Um sie zu bestimmen und Restrisiken minimiert zu können, gibt es Methoden zur Qualitätsbewertung von Softwarearchitektur.

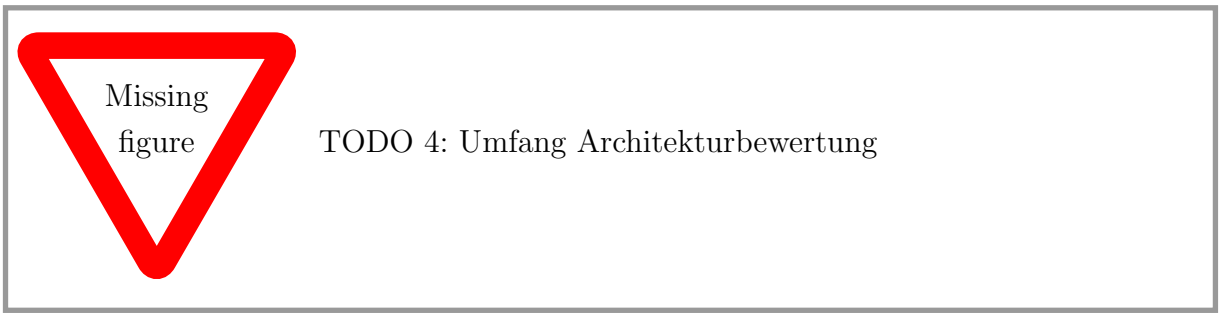

Abbildung 1: Architekturbewertung [7]

Die Bewertung kann sich dabei auf die im Softwareprojekt entstehenden Artefakte stützen. Beispielsweise sind Anforderungen, Architekturen, Diagramme, Quellcode und andere Dokumente zu nennen. Diese Artefakte können dabei quantitativ oder qualitativ bewertet werden. So kann z.B. der Quellcode mittels Metriken quantitativ, d.h. in reinen Zahlen, bewertet werden.Andere Artefakte entziehen sich dieser Bewertung und können nur auf ihre Güte hin, also qualitativ, bewertet werden. Zu letzterem zählt die Softwarearchitektur.

TODO 5: Ausführliche

Qualitätsmerkmale von Softwarearchitektur

Für die Bewertung werden bestimmte Anforderungen benötigt. Sie stellen das Architekturziel dar und charakterisieren die gewünschten Eigenschaften eines Systems[7]. Bei der Softwarearchitektur können eine Vielzahl an Qualitätsmerkmale untersucht und bewertet werden[7, 6]:

Funktionalität Mit der Funktionalität wird bestimmt, ob die funktionalen Anforderungen nicht nur vollständig erfüllt werden, sondern auch richtig und angemessen umgesetzt sind.

Zuverlässigkeit Aus der Zuverlässigkeit ergibt sich die Fähigkeit eines

	Systems, die Funktion innerhalb einer Zeitspanne umzusetzen und dabei auf etwaige Fehler zu reagieren bzw. diese zu verhindern.
Leistung	Die Leistung (Performance) gibt an, wie schnell das System reagiert und wie viele Eingaben pro Zeiteinheit verarbeitet werden können. Diese Anforderungen lassen sich meist über Benchmarks testen.
Flexibilität	Eine Architektur ist flexibel, wenn sie angepasst und erweitert werden kann, ohne dabei das gesamte Konstrukt zu zerstören.
Übertragbarkeit	Durch die Übertragbarkeit lässt sich das System auch unter verschiedensten Voraussetzungen betreiben. (Hard- und Softwareumgebungen)
Unterteilbarkeit	Die Unterteilbarkeit ermöglicht eine einfache Aufteilung in Teilsysteme. So können einzelne Teile unabhängig entwickelt oder sogar betrieben werden.
Konzeptuelle Integrität	Das Konzept sollte sich auf allen Ebenen widerspiegeln und sich durch ein einheitlichen Design präsentieren. Ähnliche Dinge sollen dabei in ähnlicher Art und Weise gelöst werden.
Machbarkeit	Es muss möglich sein, das System mit vorhandenen Ressourcen (Technologie, Budget, usw.) umzusetzen.

Diese Aufzählung stellt hierbei eine Auswahl an verschiedenen Qualitätsmerkmalen dar und erhebt dabei keinen Anspruch auf Vollständigkeit.

Soll-Ist Vergleich

Mit den Anforderungen kann ein Soll-Ist Vergleich anhand der Artefakte durchgeführt werden. Jede einzelne Anforderung wird mit Plänen, Dokumentation oder Modellen verglichen und bewertet. D.h. es wird geprüft, ob das geplante System den Anforderungen entsprechen wird. Wichtig ist, dass die geforderten Ziele möglichst granular sind. Je detaillierter die Anforderungen vorliegen, desto geringer ist das Restrisiko einzelne Punkte zu vergessen oder unscharf zu bewerten.

Das Ergebnis dieser Prüfung kann folgendermaßen aussehen[7]:

Soll = Ist Das Soll wird erfüllt und die Architektur besitzt alle geforderten Eigenschaften.

Soll ≈ Ist Das Soll wird nur teilweise erreicht und es werden Kompromisse geschlossen. (Verbesserung einzelner Anforderungen)

Soll ≠ Ist Das Soll wird nicht erfüllt und es ergibt sich ein Risiko für das System

Die Architekturbewertung ist somit ein relativer Vergleich in Hinblick auf spezifische Kriterien und liefert keine absolute Aussage über die Qualität. Vielmehr identifizieren sich aus einer Architekturbewertung Risiken, die die Architekturentscheidungen in der Entwurfsphase mit sich gebracht haben.

Diesen Ansatz verfolgt auch Software Architecture Analysis Method (SAAM). SAAM ist die erste am Software Engineering Institute (SEI), einem Wegbereiter für die Architekturbewertung, entwickelte Methode.Viele der heute bekannten Methoden basieren auf der SAAM und erweitern diese um spezielle Sichten oder fokussieren sich auf ein spezielles Qualitätsmerkmal. Mit der ATAM wurde die SAAM von der SEI weiterentwickelt und stellt heute die führende Methode zur Architekturbewertung dar.[3]

TODO 6: Umformulieren

3.1 Architecture Tradeoff Analysis Method (ATAM)

Im Folgenden Abschnitt möchte ich ATAM genauer Vorstellen. Das Ziel dieser Bewertungsmethode ist nicht eine exakte Vorhersage über die zu erwartende Qualität der Software zu treffen. Das ist nahezu unmöglich bei der frühen Entwurfsphase, da noch nicht genügend Informationen vorliegen. Vielmehr soll der Blick auf einzelne Qualitätsmerkmale und dessen Abhängigkeit zu gewissen Architektur-Entwurfsentscheidungen geschärft werden[6]. Mit diesem Wissen können einzelne Entscheidungen überdacht, genauer modelliert oder angepasst werden, um das gewünschte Qualitätsziel zu erreichen. Darüber hinaus wird auch die Dokumentation der Architektur verbessert, da alle Qualitätsaspekte genauer untersucht werden.

Das Ziel von ATAM ist eine Dokumentation von Risiken, Empfindlichkeiten und Kompromisse, die durch eine genauere Analyse der Architektur[6] ermittelt werden konnten. Risiken stellen nicht getroffene Architekturentscheidungen oder Eigenschaften der Architektur,die nicht vollständig verstanden wurden, dar. So kann z.B. der verwendete Datenbanktyp noch ungeklärt oder die Auswirkungen einer zentral geführten Komponente unklar sein. Empfindlichkeiten sind starke Abhängigkeiten messbarer Qualitätsmerkmale. Diese beziehen sich auf die Auslegung von einzelnen Komponenten der Architektur. Ein

Beispiel für Empfindlichkeiten ist ein Engpass zwischen zwei Modulen. Dieser kann für den Durchsatz des gesamten Systems verantwortlich sein. Zusätzlich deckt der Kompromiss Empfindlichkeiten auf, die sich entgegen wirken. Die Folge aus einer breiteren Auslegung des Engpasses kann unter Umständen die Stabilität des Systems verschlechtern. Mit den Risiken, Empfindlichkeiten und Kompromissen kann die Architektur mit gezielteren Analysen, Erstellung von Prototypen und weiteren Entwürfen stark verbessert werden.

Die Grundvoraussetzung zum Erreichen dieses Zieles ist eine detaillierte Definition der Qualitätsmerkmale und eine genaue Spezifikation der Architektur sowie dessen zugrunde liegenden Entscheidungen[6]. In der Praxis ist es leider nicht unüblich, dass Ziele und Architekturdetails noch unklar oder mehrdeutig sind. Aus diesem Grund ist ein wichtiges Ziel von ATAM auch dies genauer zu definieren und eindeutig festzuhalten. Um die Erreichung sämtlicher Ziele zu unterstützen, ist ATAM in mehrere Phasen aufgeteilt. Bild 2 gibt hierzu einen Überblick.

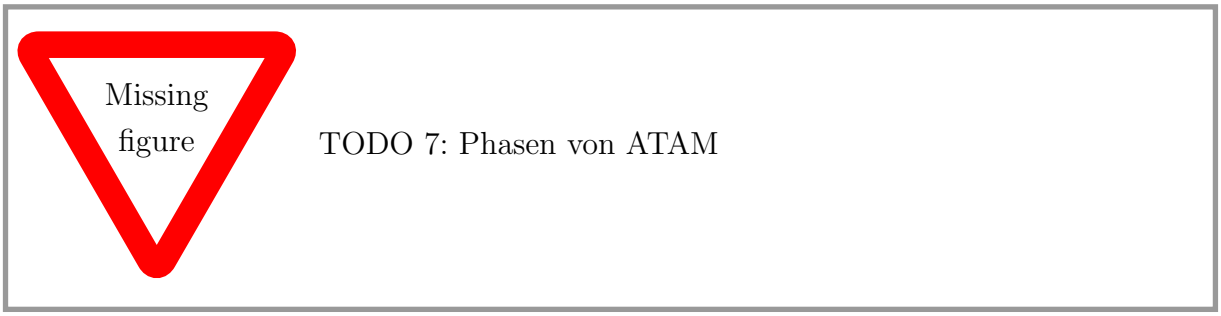

Abbildung 2: Phasen der Architekturbewertung nach ATAM [7]

Vorbereitung

Das Fundament der Bewertung stellen die Qualitätsanforderungen dar. Diese können nur lückenlos bestimmt werden, wenn alle maßgeblich vom Projekt betroffenen Personen involviert sind. In der Vorbereitungs-Phase müssen alle wichtigen Stakeholder identifiziert werden. Neben dem Kunden bzw. Auftraggeber selbst, kann dies z.B. der Benutzer, Administrator oder Tester sein. Dabei müssen jene Personen nicht direkt in den Prozess miteinbezogen werden. Es reicht schon einen Vertreter, meist aus dem Management, zu bestimmen oder die Wünsche und Ziele vorher genau zu ermitteln. In der Regel werden nur wenige Stakeholder, meist Personen aus dem Management und der Projektleitung[7], direkt zur Architekturbewertung eingeladen. Zu viele Personen würden den Prozess, durch

Diskussionen und Abschweifungen, nur verlangsamen und ineffektiv gestalten.

Bewertungsmethode (ATAM) vorstellen

Im ersten Schritt soll die Bewertungsmethode vorgestellt und dessen Ziele verdeutlicht werden. Nicht jeder Stakeholder ist regelmäßig in eine Architekturbewertung involviert. So muss klar gestellt werden, welche Bedeutung das Architektur- und Qualitätsziel hat[7]. Des Weiteren sollte hervorgehoben werden, dass es bei der qualitativen Architekturbewertung um das Aufdecken von Risiken sowie um mögliche Maßnahmen geht. Es sollte nicht Ziel sein Noten für die Architektur zu vergeben. Insbesondere sollte die Präsentation folgende Punkte enthalten[6]:

- Eine Kurze Beschreibung von ATAM und dessen Ablauf
- Methoden zur Analyse sollten erklärt werden
- Das Ziel und die Ergebnisse der Evaluation

Qualitätsziel vorstellen

Die Qualitätsanforderungen werden wesentlich vom Auftraggeber bestimmt. Aus diesem Grund sollte er auch diese vorstellen und dabei genau erläutern, was die Gründe für die Entwicklung sind und wie das System in die fachlichen Unternehmensprozesse eingeordnet werden soll. Selbst wenn die Ziele bereits in einem Anforderungsdokument erfasst wurden, ist es wichtig, die aktuelle Sichtweise des Auftraggebers zu begreifen. Zudem sind Zielformulierungen aus den Anforderungsdokumenten, meist von Systemanalytikern, gründlich gefiltert worden[7]. So können Teilnehmer Rückfragen stellen und Aha-Erlebnisse auslösen.

Architektur erläutern

In dieser Phase wird vom Softwarearchitekten die Architektur vorgestellt. Dabei wird nicht nur das eigentliche System erläutert, sondern es sollte auch der gesamte Kontext mit einbezogen werden[7]. Es beinhaltet Nachbarsysteme oder Plattformen mit denen das System in Verbindung steht. Eine angemessene Detailtiefe spielt dabei auch eine Rolle. Was angemessen ist, hängt in erster Linie von den vorhandenen Informationen (Dokumente, Diagramme, usw.) und der verfügbaren Zeit ab. Dies stellt einen wesentlichen

Punkt in der Bewertung dar[6], da nur die bereitgestellten Informationen in die Analyse mit einbezogen werden können. Je mehr vorhanden ist, desto tiefer und detaillierter kann die Bewertung durchgeführt werden. Wichtige und benötigte Dokumente oder Entscheidungen zur Architektur sollten unbedingt vor der Evaluation erstellt worden sein. Die Präsentation der Architektur sollte hier folgende Informationen umfassen[6, 7]:

- Bausteine der oberen Abstraktionsebene
- Ausgewählte Laufzeitsichten wichtiger Use-Cases
- Technische Einschränkungen, wie Betriebssystem, Hardware oder Middleware
- Weitere Systeme mit denen dieses zusammenhängt

Architekturansätze identifizieren

Aus der vorangegangenen Information können nun Architekturentscheidungen identifiziert werden, die zur Erfüllung spezieller Qualitätsanforderungen dienen. Es ist zu klären, wie die Architektur, strukturell oder konzeptionell, die wesentlichen Probleme oder Herausforderungen löst. Die treibenden und prägenden Architekturansätze werden dabei von den Architekten hervorgehoben und dienen der Ergänzung des vorangegangen Überblicks.[7] Die Erfüllung der kritischen Anforderungen von der Architektur wird so sichergestellt.

Bildung eines Qualitätsbaumes (Quality Utility Tree)

In dieser Phase werden alle Stakeholder aktiv. Sie identifizieren und verfeinern die wichtigsten Qualitätsanforderungen des Systems. Wichtig dabei ist den Fokus nicht auf die wesentlichen Anforderungen zu verlieren und sich in Feinheiten zu verstricken[6]. Damit dies gelingen kann, wird ein Qualitätsbaum erstellt. So werden die wesentlich geforderten Anforderungen, z.B. in einem Brainstorming, erfasst und anschließend in Baum-Form angeordnet. Die globalen Merkmale stehen auf der linken Seite und werden nach rechts hin verfeinert. Bild 3 stellt ein Beispiel für einen solchen Qualitätsbaum dar.

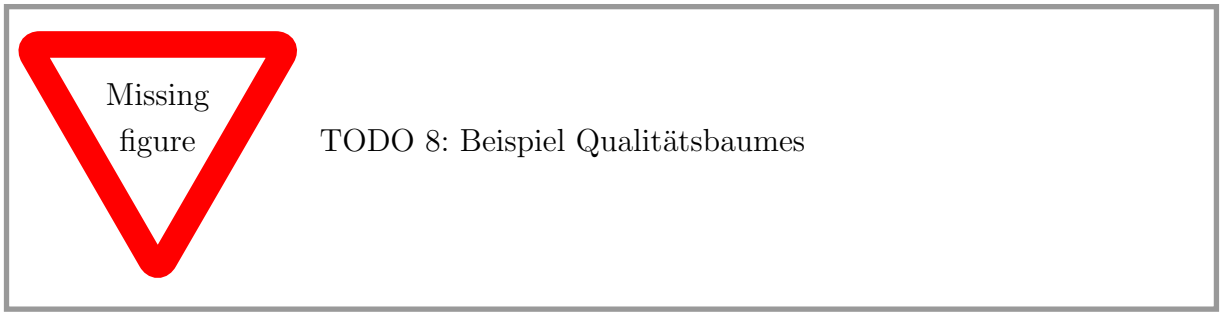

Abbildung 3: Beispiel eines Qualitätsbaums.

Anschließend werden zu den Qualitätsanforderungen Szenarien erstellt. Diese bilden einen wichtigen Bestandteil der Bewertungsmethode. Sie dienen dazu, die Anforderungen genauer zu definieren und den Stakeholdern somit näher zu bringen. Dabei muss die Formulierung eines Szenarios möglichst konkret sein und ein Beispiel darstellen, wie sich das System in einer bestimmten Situation verhält. Ein Szenario sollte folgende Informationen umfassen [7]:

Auslöser	Welcher spezifischer Stimulus löst das Szenario aus.
Quelle	Woher stammt der Auslöser. Z.B. intern, extern, Benutzer, Administrator usw.
Umgebung	Welcher Bestandteil des Systems ist betroffen.
Antwort	Wie reagiert das System auf den Auslöser.
Metrik	Wie kann die Antwort gemessen oder bewertet werden.

Die Szenarien sind nicht nur eine genauere Definition der Anforderungen. Auch helfen sie den Projektbeteiligten dabei, die Qualitätsmerkmale genauer zu verstehen, da sie weniger abstrakt sind und konkrete Anwendungsfälle enthalten. Ein, um Szenarien erweiterter Qualitätsbaum, ist in Bild 4 dargestellt.

TODO 9: Qualitätsbaum und Szenarien

Abbildung 4: Qualitätsbaum mit Szenarien erweitert

Zuletzt werden den Szenarien Prioritäten zugeordnet, so dass der Fokus zuerst auf kritische Anforderungen steht. Auch ist der Bewertungs-Workshop meist zeitlich begrenzt und somit wird sichergestellt, dass die Nichterfüllung essentieller Qualitätsmerkmale schnell zu einem Abbruch führt. Es kann durch eine einfache Skala mit wenigen Prioritäten erreicht werden, z.B. A = hoch, B = mittel und C = weniger wichtig.

Architekturansätze analysieren

Anhand der Prioritäten kann nun die eigentliche Analyse beginnen. In kleineren Gruppen, zusammen mit den Architekten oder Entwicklern, werden die Szenarien genauer untersucht und zugehörige Architekturansätze erläutert. So kann z.B. mittels eines Walkthrough genau aufgezeigt werden, wie einzelne Komponenten zur Erreichung des Ziels interagieren und welche Entwurfsentscheidungen unterstützen[7]. Dabei sollten folgende Fragen geklärt werden:

- Wie wird das Szenario in der Architektur umgesetzt?
- Warum wurde dieser Architekturansatz gewählt?
- Gibt es Kompromisse, die gemacht wurden?
- Werden andere Qualitätsmerkmale davon beeinflusst?
- Gibt es Risiken die damit verbunden sind?
- Wird die Entscheidung von Analysen, Untersuchungen oder Prototypen unterstützt?

Die Analyse ist dabei nicht auf diese Fragen beschränkt. Sie bieten lediglich einen Startpunkt für eine Diskussion, um potentielle Risiken, Empfindlichkeiten oder Kompromisse

zu finden. An dieser Stelle gibt es leider kein Patentrezept oder eine spezifisches Vorgehen, um das Erfüllen des Qualitätsmerkmales zu bestimmen. Viel mehr wird von den Teilnehmern ein analytisches und systematisches Denken verlangt. Im Fokus muss hierbei stehen, dass eine Verbindung zwischen der Architekturentscheidung und der Qualitätsanforderung geschaffen wird, die sie erfüllen soll.

Ergebnis präsentieren

Am Ende wird das Ergebnis den Stakeholdern präsentiert. Es muss nicht zwangsläufig nur durch eine Präsentation dargestellt, sondern kann auch um einen detaillierten Berichte ergänzt werden. Sämtliche Phasen der Bewertung und dessen Ergebnisse können so noch einmal rekapituliert werden und bilden die Basis zur Definition von eventuell benötigten Maßnahmen.

Die durch ATAM gefundenen Risiken und die daraus resultierenden Maßnahmen können auch Änderungen an der Architektur darstellen. Daher bietet sich ATAM auch zum iterativen Einsatz an. Mit ersten Entwürfen einer komplexen Architektur kann überprüft werden, ob die Entscheidungen den richtigen Weg einschlagen und wichtige Anforderungen von Beginn an mit einbezogen werden. So bildet sich Schrittweise eine detaillierte Architektur unter gut dokumentierter Berücksichtigung der Qualitätsanforderungen.

Einen anderen Ansatz verfolgt die SAEM Methode. Diese betrachtet die Softwarearchitektur als finales Produkt der Entwurfsphase und versucht mittels dem ISO/IEC 9126-1[1] Qualitätsmodell die Qualität der Architektur festzustellen.

3.2 Software Architecture Evaluation Model (SAEM)

Im Gegensatz zu ATAM ist SAEM weniger bekannt und erprobt[1]. Sie zielt darauf ab die Qualität der Softwarearchitektur selbst festzustellen. Nicht nur weil Fehler in der Architektur schwer zu beheben sind, sondern diese auch direkt einen Einfluss auf die Qualität und Kapazitäten des Systems haben[5]. Dies trifft besonders auf die nicht-funktionalen Anforderungen, wie Wartbarkeit, Skalierbarkeit oder Effizienz, zu.

Wie beim finalen System ist die direkte Messung am Produkt eine objektive und sehr effektive Evaluation der Qualität. Um dies zu erreichen, müssen Qualitätsanforderungen definiert und ein Prozess zur Überprüfung dieser geplant, implementiert und kontrolliert

[1] Die ISO/IEC 9126-1 wurde 2011 durch die ISO/IEC 25010 ersetzt. Die neue Norm bietet in diesem Zusammenhang jedoch keinen Mehrwert.

werden[5]. SAEM versucht dies mittels eines Qualitätsmodells auf die Softwarearchitektur abzubilden und stellt Methoden sowie Metriken für die Evaluation bereit.

Spezifikation Qualität

Das Qualitätsmodell von SAEM baut dabei auf das Modell der ISO/IEC 9126-1 auf. Jenes teilt die Qualität in 6 Kategorien und weitere Unterkategorien auf[4]. Anhand derer kann die Qualität der Softwarearchitektur spezifiziert werden, indem man diesen Architekturanforderungen zuordnet.

TODO 10: Qualitätsmodell der ISO/IEC 9126-1

Abbildung 5: Qualitätsmodell der ISO/IEC 9126-1 [4]

Hierzu wird im ersten Schritt die externe Qualität bestimmt. Diese basiert auf messbaren Metriken unter Nutzungsbedingungen und werden vom Nutzer sowie den Architekten definiert[5]. So sollten, für alle Qualitätskategorien, Anforderungen an die Architektur ausgewählt und optimale bzw. zulässige Werte bestimmt werden. Ein Beispiel hierfür ist die Reaktionszeit auf Anfragen. Mittels der externen Qualität können die Architekten die interne Qualität bestimmen. Diese sind Anforderungen, die zur Erreichung und Zufriedenstellung des Ziels benötigt werden. Im Unterschied zu externen Anforderungen werden diese nicht durch die Nutzung bestimmt, sondern beziehen sich auf intrinsische Eigenschaften, wie z.B. Modularität oder Komplexität [5]. Die Bestimmung dieser basiert meist auf Expertenwissen oder Firmen interne Richtlinien.

Bestimmung von Metriken

Für die Messung der Qualitätsanforderungen müssen Metriken bestimmt werden. Hierzu kann das Bewertungsteam aus bekannten Softwaremetriken wählen oder systematisch

neue definieren. Wichtig ist nur, dass die Messung zielorientiert ist. D.h. die Metriken müssen mit einem Top-Down Ansatz bestimmt werden[2]. Somit soll nicht aus vorhanden Messungen ein Bezug zur Qualität aufgebaut, sondern mittels spezifischer Qualitätsanforderungen passende und aussagekräftige Metriken gefunden werden.

Eine sehr gute Vorlage hierzu ist die GQM Methode. Diese setzt die Qualitätsanforderungen an erste Stelle, welche das Ziel darstellen. Auf dessen Basis werden Fragen abgeleitet, die das Ziel wiedergeben. Dabei muss es pro Ziel nicht nur eine Frage geben. Zur Erreichung des Ziels kann eine Komposition aus verschieden Fragen nötig sein. Dabei hilft es, das Ziel aus mehreren Sichten zu betrachten. Nach [2] sind Teilkriterien zur Erfüllung der notwendigen Fragestellung:

TODO 11: Richtiges Format?

Sichtweise	Auftraggeber, Administrator, Entwickler usw.
Anwendungsbereich	Benötigte Technologien, angeschlossene Systeme, usw.
Zweck	Analyse, Kontrolle, Verständnis, usw.
Kontext	intern, extern

Anschließend werden für sämtliche Fragestellungen Metriken bestimmt, die diese am besten beantworten. Die damit erhobenen Daten können dabei objektiv (z.B. Unterstützung Standardformate) oder subjektiv (z.B. Skala für Zufriedenstellung) sein. Durch den Top-Down Ansatz stehen die erhobenen Daten immer im Zusammenhang mit den Zielen, womit die Interpretation leichter fällt.

Ein Beispiel für das Ergebnis von GQM wird in Bild 6 dargestellt.

TODO 12: Beispiele GQM

Abbildung 6: Beispiel für die Anwendung von GQM

Evaluation

An diesem Punkt ist die Evaluation denkbar einfach. Sie besteht aus dem Sammeln und Auswerten der Daten mittels der Metriken. Über die gewonnen Daten und die vorher bestimmten zulässigen Werte, wie das Vorhandensein spezifischer Funktionen, wird die Architekturqualität bestimmt. Es lässt sich auch eine Aussage über die zu erwartende Qualität des Endsystems treffen. An diesem Punkt wird der Unterschied zu ATAM wieder deutlich. SAEM versucht nicht ein tieferes Verständnis für die Architektur bei maßgeblichen Stakeholdern zu schaffen, sondern nur die Einhaltung spezifischer Anforderungen zu gewährleisten. Dies hängt dabei stark vom definiertem Qualitätsziel ab. Gerade bei der internen Qualität verlangt SAEM somit eine Firmen interne Qualitätsrichtlinie[1], die auf Erfahrung und Expertenwissen basiert.

Mit dem Wissen über verschiedene Ansätze zur Qualitätsbewertung von Softwarearchitektur kann nun versucht werden, eine Methode zu finden, die es zur Bewertung von Microservice Frameworks und die daraus vorgegebenen Architektur des Services nutzt.

4 Microservice Framework Evaluation Method (MFEM)

Die Bewertung, ob ein Framework für den produktiven Einsatz in einer Microservice Architektur geeignet ist, lässt sich nicht auf die Servicearchitektur beschränken. Aus diesem Grund ist MFEM keine reine Architekturbewertungs Methode. Sie betrachtet einerseits die durch das Framework vorgegebene Servicearchitektur und versucht dabei Risiken aufzudecken sowie ein tieferes Verständnis für die Architektur zu schaffen. Auf der anderen Seite wird das Framework als Produkt aufgefasst und versucht, eine Aussage über die Qualität dessen zu treffen. MFEM setzt dabei keine grundlegenden Kenntnisse über das Framework voraus, da es 2 Seiten der Analyse anbietet. Dies ist einerseits die Nutzung von bereits vorhandenem Experten Wissen, was die Phase der Analyse verkürzt. Andererseits werden die restlichen Daten über die Erstellung von Prototypen erfasst. Auf dieser Basis kann am Ende eine begründete Entscheidung für oder gegen den Einsatz eines Frameworks getroffen werden.

Der Ablauf von MFEM ist in Bild 7 schematisch dargestellt und startet mit der Analyse der vorhanden Microservice Architektur sowie der Identifizierung von Qualitätsanforderungen.

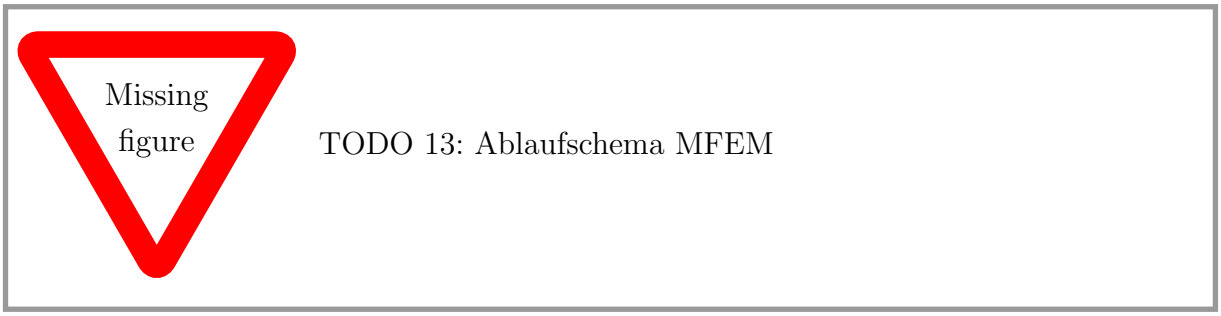

Abbildung 7: Schematische Darstellung vom Ablauf von MFEM

4.1 Architektur Analyse und Wahl der Anforderungen

Aus dem vorangegangenen Kapitel wird ersichtlich, dass eine Bewertung nicht nur frühzeitig erfolgen muss. Viel mehr ist es wichtig, dass diese auch auf detaillierten Qualitätsanforderungen basiert. Die Anforderungen werden dabei durch die maßgeblichen Stakeholder definiert. In einer Microservice Architektur sind dies die definierten Umstände eines Services. Diese beschränken sich dabei nicht auf die Kommunikation zwischen zwei einzelnen Services. Gerade die Komposition der Microservices über die Infrastruktur muss gewährleistet sein. Wobei Punkte wie Sicherheit und Wartbarkeit die Komplexität noch steigern. Damit ein Service dies erfüllt und somit in der Architektur funktionieren kann, müssen anhand der

Umstände, wie z.B. Service Discovery, Logging oder Tracing, Anforderungen definiert werden. Nur wenn das Framework eines Services diese unterstützt, kann garantiert werden, dass sich das Risiko zur Erfüllung des Qualitätszieles minimiert.

Externe Serviceanforderungen

Im ersten Schritt müssen demnach Anforderungen gefunden werden, die die Microservice Architektur an die einzelnen Services stellt. Das ist dabei stark von der Umsetzung dieser abhängig. Ein gutes Beispiel hierfür ist die Service Discovery. Dass sich Services gegenseitig finden können, ohne die Flexibilität der Architektur zu verlieren, darf eben diese nicht fehlen. Die Umsetzung kann dabei jedoch stark variieren. So kann ein zentraler Service, wie z.B. Netflix-Eureka[2] oder Consul[3] , eine Registrierungsstelle anbieten. Dort können sich sämtliche Service-Instanzen registrieren und auch andere Services finden. Diese müssen es aber auch nicht direkt unterstützen. Projekte wie Spring Cloud Sidecar[4] übernehmen dies für einzelne Services. Die Discovery kann aber auch auf die darüber liegende Abstraktions-Schicht hochgezogen werden. Innerhalb eines Kubernetes[5] Clusters übernimmt die Service Verwaltung diese Aufgabe selbst und stellt sie über Umgebungsvariablen oder DNS zur Verfügung.

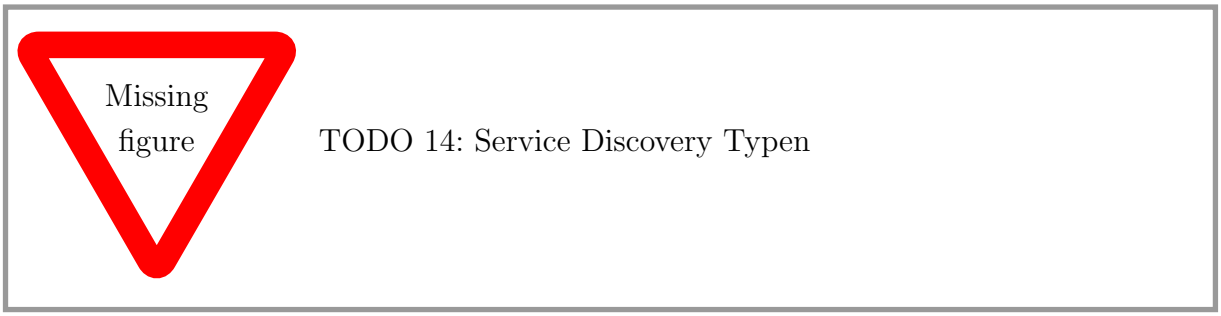

Abbildung 8: Beispiele für Ausprägungen der Service Discovery

Die spezifische Ausprägung der Architektur gibt somit Anforderungen vor, die es zu identifizieren gilt und im folgenden externe Serviceanforderungen genannt werden.

[2] https://github.com/Netflix/eureka
[3] https://www.consul.io
[4] http://projects.spring.io/spring-cloud/spring-cloud.html#_polyglot_support_with_sidecar
[5] http://kubernetes.io

Interne Serviceanforderungen

Aus den externen Serviceanforderungen lassen sich auch interne Anforderungen ableiten, die zur Erreichung und Zufriedenstellung dieser notwendig sind. Dies muss dabei nicht auf die durch das Framework vorgegebene Architektur des Services beschränkt sein. An dieser Stelle kann das Framework als Werkzeug aufgefasst werden. Es gilt somit nicht nur die Frage zu klären, ob ein Framework eine benötigte Funktion bereitstellt. Sondern ob es den Entwickler bei der Umsetzung bestmöglich unterstützt und dabei flexibel bleibt. Nach dem KISS-Prinzip[6] wird zum Beispiel die Funktion bevorzugt, die möglichst einfach und „dumm“ erscheint. So brauch es für die Umsetzung keine anspruchsvollen oder besonders cleveren Lösungen. Eine Einfache lässt sich nicht nur besser Lesen und Verstehen, es erhöht auch die Wartbarkeit.

Damit das Bewertungsteam, bei der Findung von internen und externen Serviceanforderungen, nicht bei null anfangen muss, wird an dieser Stelle eine Orientierungshilfe angeboten. Hier wurde in einem Brainstorming versucht, eine Schnittmenge an Anforderungen zu finden, die für die meisten Microservice Architekturen gelten sollte. Die so gefundenen Anforderungen basieren dabei auf der Erfahrung der Anwendungsentwicklung innerhalb der Landeshauptstadt München. Diese hat sich in den letzten Jahren mit dem produktiven Einsatz von Microservices beschäftigt und diverse erfolgreiche Projekte damit umgesetzt.

TODO 15: anders Formulieren und ausschmücken

Für die Analyse und Wahl der Anforderungen wird hier ein Quality Utility Tree aufgebaut.

4.1.1 Entwicklung Quality Utility Tree

Der Quality Utility Tree bietet einen effizienten Top-Down Ansatz zur Identifizierung und Verfeinerung von Qualitätsanforderungen. Die mittels Brainstorming gefundenen Anforderungen werden in einzelne Kategorien aufgeteilt und bis zu den konkreten Zielen weiter verfeinert. Bei der Kategorisierung kann das Qualitätsmodell aus der ISO/IEC 9126-1 hergenommen werden. Wobei das Modell nur eine Hilfestellung ist und keine Vorgabe darstellt. Hier wurden die Kategorien Funktionalität, Performance, Benutzbarkeit, Sicherheit und Wartbarkeit definiert. Anschließend wurden diesen die Qualitätsanforderungen zugeordnet. Bild 9 zeigt einen Ausschnitt des gesamten Baumes, wobei der vollständige Baum im Anhang zu finden ist.

[6] `http://principles-wiki.net/principles:keep_it_simple_stupid`

TODO 16: MFEM Quality Utility Tree Beispiel

Abbildung 9: Ausschnitt aus dem Quality Utility Tree als Beispiel

4.2 Metriken definieren über GQM

Für die identifizierten Qualitätsanforderungen müssen möglichst aussagekräftig Metriken gefunden werden. Dies können dabei Messung an der Architektur sein, wie das Vorhandensein spezifischer Funktionen oder Komponenten sowie den Einsatz bewährter Pattern. Zusätzlich können auch Softwaremetriken genutzt werden, die sich direkt an einem Prototypen ermitteln lassen. Damit kann neben Messungen zur Performance auch z.B. der Aufwand zur Umsetzung bestimmter Funktionen bemessen werden. Um dabei stets das Ziel im Auge zu behalten, wird die bereits bekannte GQM Methode genutzt.

Eine um Metriken erweiterter Qualitätsbaum ist in Bild 10 dargestellt. Wie zuvor ist dies nur ein Ausschnitt, wobei der gesamte Baum im Anhang zu finden ist.

TODO 17: MFEM Qualitätsbaum erweitert um Metriken

Abbildung 10: Ein Ausschnitt aus dem Qualitätsbaum erweitert um Metriken

4.3 Analyse

4.3.1 Evaluationsmethoden

Subjektive Evaluation

Cognitive Walkthrough

Objektive Evaluation

Performance Messung

5 Evaluation der Methode an Beispielen

5.1 Spring Boot

5.1.1 Methodenanwendung

5.1.2 Auswertung

5.2 Go-KIT

5.2.1 Methodenanwendung

5.2.2 Auswertung

6 Vergleich und Ausblick

6.1 Methoden Anpassung/Erweiterung

6.2 Ausblick

7 Quellenverzeichnis

[1] L. Dobrica und E. Niemela. „A survey on software architecture analysis methods“. In: *IEEE Transactions on Software Engineering* 28.7 (Juli 2002), S. 638–653. ISSN: 0098-5589. DOI: 10.1109/TSE.2002.1019479.

[2] Sascha-Ulf Habenicht. *Das GQM-Paradigma*. Zugriff: 22.10.2016. URL: www.sascha.habenicht.name/publikationen/GQM-Paradigma.pdf.

[3] Software Engineering Institute. *Architecture Tradeoff Analysis Method*. Zugriff: 22.10.2016. URL: http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm.

[4] ISO/IEC. *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001.

[5] William L. de Oliveira und Juan A. de la Puente Juan C. Dueñas. „A Software Architecture Evaluation Model“. In: *Development and Evolution of Software Architectures for Product Families: Second International ESPRIT ARES Workshop Las Palmas de Gran Canaria, Spain February 26–27, 1998 Proceedings*. Springer Berlin Heidelberg, 1998, S. 148–157. ISBN: 978-3-540-68383-4.

[6] Rick Kazman und Mark Klein Paul Clements. *ATAM: Method for Architecture Evaluation*. Zugriff: 22.10.2016. URL: http://www.sei.cmu.edu/reports/00tr004.pdf.

[7] Gernot Starke. *Effektive Softwarearchitekturen - Ein praktischer Leitfaden*. M: Carl Hanser Verlag GmbH Co KG, 2015. ISBN: 978-3-446-44406-5.

Todo list

Anhang

A MFEM - Vollständiger Quality Utility Tree

TODO 18: Quality Utility Tree vollständig mit allen Kategorien (mit GQM)

B CODE

TODO 19: Nothing Here