Fakultät für Informatik und Mathematik 07

Bacheloararbeit

über das Thema

Generative Testerstellung für Microservice-Architekturenit

Autor: Fabian Wilms
holtkoet@hm.edu

Prüfer: Prof. Dr. Ulrike Hammerschall

Abgabedatum: xx.xx.2017

I Kurzfassung

Viele Softwareprojekte scheitern an fehlerhafter Kalkulation. Die Kosten werden meist viel zu gering angesetzt und sorgen für unzufriedene Kunden, geringe Erträge und fehlende Folgeaufträge. Doch warum ist das so?

Der größte Kostenfaktor sind Fehler im laufenden System. Diese Fehler zu beseitigen ist häufig eine große Herausforderung die viel Zeit, Personal und somit Geld kostet. Es gilt also, vor Auslieferung der Software eine entsprechende Qualität gewährleisten zu können.

Aus diesem Grund ist eine umfassende Testabdeckung ein sehr wichtiger Bestandteil in der Definition of Done der Landeshauptstadt München. Unit- und Integrations Tests sind fest in viele Entwicklungsprozesse integriert und finden besonders auch in agilen Entwicklerteams großen Zuspruch. Allerdings sind auch Tests finanzielle, wie auch zeitliche Kostenfaktoren. Dennoch wird für Tests, trotz des Bewusstseins für die damit entstehenden Probleme, häufig viel zu wenig zeit eingeplant.

Um diesen problemen entgegen zu wirken wurde bei der LHM im vergangenen Jahr das Softwareprojekt Barrakuda entwickelt, welches anhand eines nach Domain Driven Design (DDD) aufgebauten Domänen-Modells Microservice Architekturen generieren kann. Die Idee: Software anhand von simplen Modellen generieren, um möglichst viel Zeit beim Entwickeln von "Boiler PlateCode einzsuparen, und die Software-Qualität nicht nur durch ersparte Entwicklungszeit, sondern auch durch homogene Projektstrukturen in allen generierten Systemen, und damit erhöhte Wartbarkeit, zu verbessern.

Im Rahmen dieser Bachelor-Arbeit soll die These geprüft werden, dass sich aus einem soclhen Modell genügend Informationen zum Testen eines Microservice-Frameworks extrahieren lassen, um eine ausreichende Testabdeckung für die generierten Services zu bieten.

Hierzu ist ein tieferes Verständnis der zu Generierung eingesetzten Sprache Barrakuda, welche von Martin Kurz in seiner Masterarbeit „Verbesserung des Softwareentwicklungsprozesses der Landeshauptstadt München durch modellgetriebene Softwareentwicklung“ entwickelt wurde, nötig. Diese wird in einem weiteren Abschnitt erklärt und anhand von Beispielen demonstriert. Um die Verbindung von Barrakuda zum DDD hervorzuheben, wird dies im weiteren Vorgehen näher beleuchtet.

Im zweiten Teilabschnitt werden dann die gängigsten Methoden und Praktiken, sowie Frameworks, die beim Testen von Microservice-Architekturen eingesetzt werden, identifiziert und weitere Anforderungsanalysen für diese durchgeführt, um die benötigte Bandbreite

der generierten Tests abzugrenzen. Vorstellbare Testarten isnd Integrationstests, Unit-Tests und Smoke Tests.

Der letzte Schritt der Vorbereitung soll dann aus den zusammengetragenen Informationen festgestellen, für welches Framework ein generativer Ansatz zum Erstellen von Testfällen am sinnvollsten ist. Außerdem sollen eventuell benötigte Erweiterungen und/oder Änderungen von Barrakuda festgelegt werden, die für den generativen Ansatz von Nöten sind.

Im praktischen Teil der Arbeit werden dann zunächst einige Referenzimplementierungen umgesetzt, anhand derer im nächsten Teilschritt die Generierung der Tests, unter Zuhilfenahme von Barrakuda und den im vorherigen Teil festgelegten Änderungen, implementiert werden.

Abstract

Das ganze auf Englisch.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Listing-Verzeichnis

VI Abkürzungsverzeichnis

1 Einführung und Motivation

Softwarequalität, Tests, Zeit, kosten etc.

> *If the software coders don't catch their omission until final system testing–or worse, until after the system has been rolled out–the costs incurred to correct the error will likely be many times greater than if they'd caught the mistake while they were still working on the initial [...] process.*[1]

2 Architektur-Generierung für Software-Projekte

2.1 Barrakuda

2.1.1 Idee

2.1.2 Einführung

2.1.3 Verbindung zum Domain-driven Design

Domain-driven Design in Microservices

Umsetzung

[1] Charette, Robert N.: Why Software Fails. ⟨URL: `http://spectrum.ieee.org/computing/software/why-software-fails`⟩.

Charette, Robert N.: Why Software Fails. ⟨URL: `http://spectrum.ieee.org/computing/software/why-software-fails`⟩

Anhang

A Code-Fragmente

Viel Beispiel-Code