



Fakultät für Informatik und Mathematik 07

Bacheloararbeit

über das Thema

Generative Testerstellung für Microservice-Architekturen

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 Plate Code" einzusparen, 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 solchen 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 zur 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 sind Integrationstests, Unit-Tests und Smoke Tests.

Der letzte Schritt der Vorbereitung soll dann aus den zusammengetragenen Informationen feststellen, 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

I	Kurzfassung	I
II	Inhaltsverzeichnis	III
III	Abbildungsverzeichnis	IV
IV	Tabellenverzeichnis	V
V	Listing-Verzeichnis	V
VI	Abkürzungsverzeichnis	VI
1	Einleitung	1
1.1	Problem	1
1.2	Tabellen	1
2	Kapitel	1
2.1	Unterkapitel	2
2.2	Unterkapitel	2
3	Quellenverzeichnis	3
	Anhang	I
A	Code-Fragmente	I

III Abbildungsverzeichnis

Abb. 1	OSGi Architektur	1
--------	----------------------------	---

IV Tabellenverzeichnis

Tab. 1	Beispieltabelle	1
--------	---------------------------	---

V Listing-Verzeichnis

VI Abkürzungsverzeichnis

1 Einleitung

Dieses Kapitel enthält Beispiele zum Einfügen von Abbildungen, Tabellen, etc.

1.1 Problem

Zum Einfügen eines Bildes, siehe Abbildung 1, wird die *minipage*-Umgebung genutzt, da die Bilder so gut positioniert werden können.

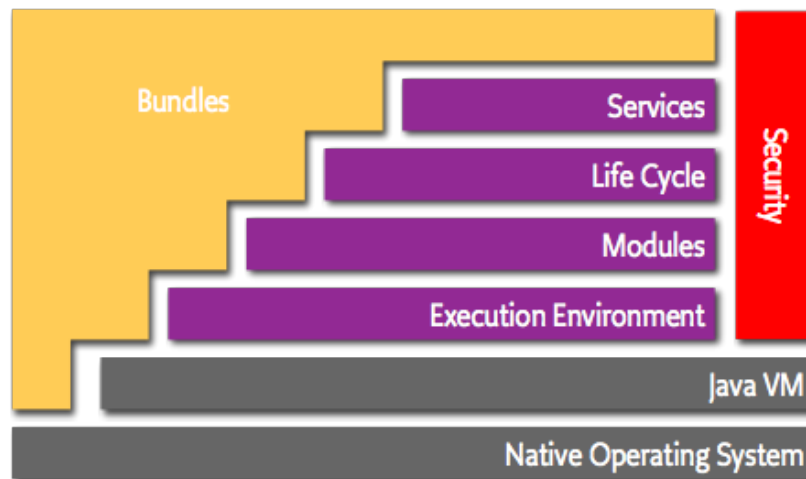


Abbildung 1: OSGi Architektur¹

2 Kapitel

Lorem ipsum dolor sit amet.

2.1 Unterkapitel

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

¹Quelle: <http://www.osgi.org/Technology/WhatIsOSGi>

2.2 Unterkapitel

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

3 Quellenverzeichnis

Anhang

A Code-Fragmente

Viel Beispiel-Code