htw saar

EmRoLab
Embedded Robotics Lab

Ausarbeitung

Software-Architektur

an der Hochschule für Technik und Wirtschaft des Saarlandes

im Studiengang Kommunikationsinformatik

der Fakultät für Ingenieurwissenschaften

Continuous Integration / Continuous Delivery and Deployment

vorgelegt von

Sayed Mustafa Sajadi

Mhd Yaman Aljazairi

Anas Alajaji

betreut und begutachtet von

Prof. Dr. Markus Esch

Saarbrücken, Tag. Monat Jahr

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, Tag. Monat Jahr

Sayed Mustafa Sajadi
Mhd Yaman Aljazairi
Anas Alajaji

Zusammenfassung

Kurze Zusammenfassung des Inhaltes in deutscher Sprache, der Umfang beträgt zwischen einer halben und einer ganzen DIN A4-Seite.

Orientieren Sie sich bei der Aufteilung bzw. dem Inhalt Ihrer Zusammenfassung an Kent Becks Artikel: `http://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html`.

Inhaltsverzeichnis

1 Einleitung

1.0.1 Motivation

Die Notwendigkeit von Software-Updates ist sehr wichtig, um die korrekte Funktion der eingebetteten Systeme fehlerfrei und problemlos zu gewährleisten, denn die Software-Bugs hatten katastrophale Folgen, insbesondere in den Bereichen Luft- und Raumfahrt, Automotive und Medizin. Zwischen Juni 1985 und Januar 1987 verabreichte ein computergesteuertes Therapiegerät namens "Therach-25" sechs Personen eine große Anzahl von Überdosierungen, die zu schweren Verletzungen und zum Tod führten. Der Software-Entwicklern sollten Funktionen zur Verfügung gestellt werden, die den Software-Entwicklungsprozess beschleunigen und Software-Updates über das Internet ermöglichen. Denn ein manuelles Update, zum Beispiel über USB-Sticks, zum einen verlangsamt der Übertragungsprozess und zum anderen kann während der Übertragung zu menschlichen Fehlern führen. Darüber hinaus ist dieser Prozess anfällig, wie Fiat Chrysler 2015 damit begonnen hat, einen Hotfix für Millionen von Fahrzeugen per Post über einen USB-Stick zu verteilen, der es Hackern ermöglicht, Briefe und USB-Sticks zu fälschen.

1.0.2 Aufgabenstellung und Zielsetzung

In dieser Arbeit soll die verschiedenen Release-Management-Methoden erklärt und insbesondere die Methode Continuous Integration, Delivery and Deployment (CICD) herausgearbeitet und die verschiedenen dabei eingesetzten Tools sowie deren Unterschiede erläutert werden. Als nächstes wird auf die Frage eingegangen, welche CI/CD-Auswirkungen die Softwarearchitektur und die Entwicklungsprozesse von haben. Darüber hinaus wird die Anwendung dieser Methode anhand eines Fallbei-

spiels näher untersucht, in dem der Fall der Anwendung eines Software-Updates eines Endgeräts über das Internet tatsächlich betrachtet und unter Verwendung einer Container-Technologie namens Docker umgesetzt wird.

2 Grundlagen

3 Release Management Methoden

4 Continuous Integration und Continuous Delivery

5 Continuous Integration und Continuous Delivery

6 Fallstudie

6.1 Anforderungen

6.1.1 Funktionale Anforderungen

6.1.2 Nicht-Funktionale Anforderungen

6.2 Konzept

6.2.1 Geeignete Software-Update-Strategie der Ampelanlage

6.2.2 Software Update-Architektur der Ampelanlage

6.2.3 Auswahl der geeigneten Release-Managementmethode

6.2.4 Auswahl der geeigneten Release-Strategie

6.3 Entwicklungsvorgang

6.4 Implementierung

6.5 Aufbau der experimentellen Umgebung

6.6 Implementierung des Release-Management-Prozesses

7 Evaluation

8 Zusammenfassung

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

Anhang