WILHELM BÜCHNER HOCHSCHULE
Mobile University of Technology

Konzeption und Implementierung einer modularen Testsuite zur Automatisierung von Softwaretests

Test

Fachbereich Informatik

B-FMP02XX

Autor	Marvin Böck
Matrikelnummer	1892597
Anschrift	Solweg 5, 78647 Trossingen
Abgabedatum	07. Juli 2022

Erklärung an Eidesstatt

Hiermit erkläre ich, dass ich die vorliegende Abschlussarbeit mit dem Titel: "*Konzeption und Implementierung einer modularen Testsuite zur Automatisierung von Softwaretests*"
eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.
Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

.....	.....
Ort, Datum	Unterschrift

Sperrvermerk

Der Inhalt der Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung der SICK STEGMANN GmbH vorliegt

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Kozeptionierung sowie Proof of Concept Implementierung einer modularen Testsuite für Softwaretests. Grund für die Entwicklung der Suite ist die Portierung eines Bestandsprodukts auf eine neue Mikrocontroller Generation und der damit verbundener erhöhter Testaufwand. Bei der Firma SICK STEGAMNN GmbH ist seit kurzem ein Framework zum automatischen Testen von Software im Einsatz welches die Möglichkeit bietet, entsprechende Softwaretests komfortabel abzudecken. Um die Generierung der Testfälle für das Portierungsprojekt zu erleichtern, soll im Rahmen dieser Arbeit eine Testsuite konzeptioniert, sowie prototypisch implementiert werden. Ziel hierbei ist es, den späteren Prozess der Testentwicklung zu beschleunigen, sowie die Testfiles übersichtlicher und robuster zu gestalten. Zu Beginn der Arbeit wird sich ein Überblick über den Bereich Automated Testing @ GBC07 sowie Software Tests im Embedded Bereich geschaffen. Im Anschluss daran, werden die Testfälle eines vergleichbaren Produkts analysiert und geeignete Testfälle identifiziert. Danach wird eine Softwarearchitektur für die Suite konzeptioniert und diese prototypisch implementiert. Zum Schluss werden die Ergebnisse der Arbeit evaluiert.

Abstract

This thesis deals with the conceptual design and proof of concept implementation of a modular test suite for software testing. The reason for the development of the suite is the porting of an existing product to a new microcontroller generation and the associated increased test effort. The company SICK STEGAMNN GmbH has recently implemented a framework for automatic software testing which offers the possibility to comfortably cover the corresponding software tests. In order to facilitate the generation of test cases for the porting project, a test suite is to be conceptualized and prototypically implemented within the scope of this work. The goal is to accelerate the later process of test development and to make the test files clearer and more robust. At the beginning of the work, an overview of the area of Automated Testing @ GBC07 and software tests in the embedded area is created. Subsequently, the test cases of a comparable product are analyzed and suitable test cases are identified. Afterwards a software architecture for the suite is conceptualized and prototypically implemented. Finally, the results of the work are evaluated.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

SRS Software Requirements Specification

1 Einleitung

1.1 SICK

2 Related Work

In diesem Kapitel werden verschiedene Literaturquellen diskutiert, welche im Zusammenhang mit dem Thema der Arbeit stehen. Hierbei wird der Inhalt kurz zusammengefasst und erläutert, in welchem Zusammenhang die Arbeiten stehen.

Optimized test suites for automated testing using different optimization techniques

In diesem Artikel, welcher von Manju Khari, Prabhat Kumar, Daniel Burgos und Rubén González Crespo geschrieben wurde, befassen sich die Autoren mit der Optimierung von automatisierten Testsuites unter Betrachtung verschiedener Optimierungstechniken. Hierbei werden Tools zur automatischen Generierung von Testsuites verglichen. Die Ergebnisse werden dann im Kontext des Automated Testing betrachtet. Die Arbeit zeigt auf, welche Möglichkeiten bei der Optimierung und automatischen Generierung von Softwaretests bestehen. Im Zuge des konkreten Projekts wird ebenfalls eine Testsuite erstellt. Die Betrachtung von eventuellen Optimierungsmöglichkeiten ist für den späteren Verlauf des Projekts durchaus von Interesse.[?]

Eine Technologe für das durchgängige und automatisierte testen eingebetteter Software

Die Arbeit, welche durch Dipl.-Inform. Till Fischer im Rahmen seiner Dissertation an der Fakultät für Elektrotechnik und Informationstechnik des Karlsruher Instituts für Technologie (KIT)durchgeführt wurde, werden neben den Grundlagen des Testens von eingebetteten Systemen auch die verschiedenen Testebenen diskutiert. Ziel der Arbeit ist die Verbesserung der Durchgängigkeit des Testprozesses für eingebettete Systeme. Als Lösungsstrategie wird durch Fischer eine Testlösung versiert, welche den Quelltext der ausgeführten Software, die Netzwerkkommunikation, sowie das physikalische Verhalten an elektrischen Schnittstellen und der simulierten Umgebung abdeckt. Hier ist

der Bezug zum in dieser Arbeit behandelten Automated Testing @ GBC07 Projekt zu erkennen. [?]

3 Grundlagen Softwareentwicklung und Verifikation

4 Anforderungen

4.1 Anforderungsdefinition

Zu Beginn des Projekts werden die konkreten Anforderungen an die Suite mit dem Auftraggeber abgestimmt und in Form einer Tabelle Festgehalten. Dieses Vorgehen erleichtert die Endabnahme des Projekts, da die Anforderungen klar definiert und beiden Parteien bekannt sind. Auf die Erstellung eines gesamten Software Requirements Specification (SRS) wird im Rahmen dieses Projekt aus Zeitgründen bewusst verzichtet. Bei den Anforderungen wird zwischen funktionalen und nicht-funktionalen unterschieden. Funktionale Anforderungen sind all jene, welche eine konkrete Funktion der Software definieren. Nicht Funktionale Anforderungen sind Anforderungen wie z.B. Zuverlässigkeit und Zeitverhalten. Die Anforderungen sind in den Tabellen "Anforderungen funktional"und "Anforderungen nicht-funktional"dargestellt.

Nr.	Bezeichnung	Beschreibung	Zuordnung
F01	Error Codes	Eignung zur Stimulation der verschiedenen Fehlerfälle und dadurch Produktion der entsprechenden Fehlermeldungen (01h-23h)	Funktional
F02	Identifikation des DUT	Überprüfung der Firmware Version, des elektronischen Typenschilds auf Korrektheit	Funktional
F03	Diagnose Funktionalität	Überprüfung der vorhandenen Diagnose Funktionalität (z.B. Spannungs Monitoring)	Funktional
F04	Core Funktionalität	Prüfung der externen Temperatur Schnittstelle, sowie der Single und multiturn Positionsbestimmung	Funktional
F05	Schnittstellen Config	Auslesen und überprüfen der Schnittstellen Parametrisierung und Funktionalität	Funktional
F06	Hiperface Befehle	Überprüfung der Funktion der verschiedenen Hiperface Befehle sowie der entsprechnden Antwort	Funktional
F07	Integration in Automated Testing	Die Suite muss in das Projekt Äutomated TestingİntegrierbarÄ sein (Implementierung in ITE, Aufbau nach Vorgaben des Projekts)	Funktional

Tabelle 1: Anforderungen funktional

Nr.	Bezeichnung	Beschreibung	Zuordnung
NF01	Externes Equipment	Möglichkeit zur Integration des am Teststand vorhandenen externe Equipment	Nicht Funktional
NF02	Universell einsetzbar	Der Aufbau der Suite soll so gestaltet sein, das diese auch zukünftig für die Portierung anderer (ähnlicher) Produkte genutzt werden kann	Nicht Funktional
NF03	Coding Guidelines	Der Aufbau der Suite soll so gestaltet sein, das diese auch zukünftig für die Portierung anderer (ähnlicher) Produkte genutzt werden kann	Nicht Funktional

Tabelle 2: Anforderungen nicht-funktional

4.2 Analyse bisheriger Testfälle

Zu Ermittlung der Testfälle welche durch die Suite abgedeckt werden sollen werden die bisherigen Testpläne analysiert. Schwierigkeit hierbei ist, dass der Geber vor ca. 20 Jahren entwickelt wurde. Zum Entwicklungszeitpunkt fand keine, den heutigen Standards entsprechende Dokumentation der Testfälle (Testplan) statt, weiterhin ist zum Projektstart noch kein Testplan für Portierung des Controllers vorhanden. Aus diesem Grund wird für die Analyse der Testfälle auf den Testplan eines ähnlichen Produktes (SEY) welches ebenfalls über eine Hiperface Schnittstelle verfügt zurück gegriffen.
Die im Testplan festgelegten Tests werden in eine Excel Tabelle übertragen, die Testfälle sind im Testplan in verschiedene Kategorien unterteilt (z.B. General Requirements, HW Architecture, functional Requirements). Da der SEY über zwei Mikrocontroller verfügt (Master und Slave) der SK jedoch nicht werden daraufhin die Testfälle des Slave Microcontrollers aus dem Testplan gestrichen. In einem Zweite Schritt wird die Beschreibung jedes Testfalls Analysiert und geprüft ob dieser Testfall automatisierbar ist. Ein Beispiel für einen nicht automatisierbaren Testfall ist z.B. beim Eingriff mittels Debugger während des Tests gegeben. Die verbleibenden Tests werden mit dem Zuständigen SW-Entwickler besprochen und ggf. angepasst um im folgenden als Grundlage für die Kozeptionierung dienen zu können. Die Entwicklung eines kompletten Testplans für die Portierung wird hierbei nicht forciert, da hierzu der Zeitliche Rahmen nicht gegeben ist. Die Analyse der Testfälle zeigt, das sich die Suite besonders für das Testen der Hiperface Schnittstelle eignet. Hier können die verschiedenen Befehle der Schnittstelle gut abgebildet werden. Die Ergebnisse der Analyse sind in "Tabelle mit Analysierten Testfällen"Beispielhaft dargestellt und befinden sich im Anhang. Die Farbgebung der Tabelle entspricht den Testfall Clustern.

Test	Kat.	Inhalt
SW00001	General Requirements	Firmeware version format
SW00030	HW Architecture	Master IcDSL IcDSL resolution is set by EEPORIM parameter Part 1: SEY90
SW00031	HW Architecture	Master IcDSL IcDSL resolution is set by EEPORIM parameter Part 2: SEY70
SW00058	Functional Requirements	Boot-loader
SW00132	Functional Requirements	General user cmds W_EnterBootloader (70h) NO support of safety variants
SW00017	HW Architecture	Master mic. Supply Voltage Monitoring Part 1 (Accuracy)
SW00018	HW Architecture	Master mic. Supply Voltage Monitoring Part 2 (Lower limit)
SW00019	HW Architecture	Master mic. Supply Voltage Monitoring Part 3 (Higher limit)
SW00020	HW Architecture	Master mic. Supply Voltage Monitoring Part 4 (Voltage Correlation)
SW00021	HW Architecture	Master mic. Internal Voltage Rail 1.8V
SW00064	Functional Requirements	Master power supply check
Neu		LED Strom
SW00135	Calibration/Production	Internal temperature sensor
SW00036	HW Architecture	IcDSL Master IcDSL temperature read-out (Temperature coefficients)
SW00037	HW Architecture	IcDSL Master IcDSL temperature read-out (warning limits)

Abbildung 1: Tabelle mit Analysierten Testfällen

5 Konzept

6 Implementierung

7 Evaluation

8 Fazit / Ausblick

9 Literatur

10 Anhang