Hochschule für Wirtschaft und Recht Berlin
Studiengang: MA Nachhaltigkeits- und Qualitätsmanagement

Masterarbeit:

Machbarkeitsstudie über die Automatisierung von Qualitätssicherungsmaßnahmen in der Entwicklung medizinischer Produkte.

Eine vergleichende Analyse der Testautomatisierung der Entwicklung sicherheitskritischer Software zwischen der Automobilindustrie und der Medizintechnik

eingereicht von

Haider Karomi

Matrikel-Nr.: 311964

Abgabe: 20.09.2012

Abstract

Eingebettete Systeme werden verstärkt zum integralen Bestandteil sicherheitskritischer Anwendungen. Sie übernehmen die Steuerung und Regelung komplexer interner Abläufe und/oder interagieren mit der Umgebung. Zudem übernehmen diese Geräte die Aufgabe der Vernetzung verteilter und gleichzeitig ablaufender Prozesse zwischen Sender und Empfänger, was neue Fehlerarten mit sich bringen kann. Der Fehleranteil der Software im Medizinbereich oder im Automobilbereich liegt heute bei ca. 20 Prozent, so dass Qualitätssicherung eine hohe Bedeutung erhält. Das Testen ist eine der wichtigsten Säulen der Qualitätssicherung. Mit einem Test wird geprüft, ob das System entsprechend den Anforderungen implementiert wurde. Verschiedene Teststrategien und Konzepte sind notwendig, um eine akzeptable Testabdeckung bzgl. der Kundenanforderungen, der Normen und gesetzlichen Anforderungen zu erreichen.

Im Anwendungsgebiet Automotive ist die Testautomatisierung bereits sehr weit fortgeschritten. Ob eine Testautomatisierung und entsprechend neue Konzepte auf das Anwendungsgebiet Medizin übertragen werden können, hängt von verschiedenen Faktoren ab, die noch weitgehend unklar sind und die im Rahmen einer technischen und wirtschaftlichen Machbarkeitsstudie für das automatisierte Testen eingebetteter Systeme sicherheitskritischer Anwendungen untersucht werden. Im Rahmen dieser Untersuchung wurden eine technische und eine wirtschaftliche Machbarkeitsanalyse sowie eine Marktanalyse durchgeführt.

Danksagung

Diese Masterarbeit wurde im Rahmen des Studiengangs „Nachhaltigkeit und Qualitätsmanagement“ an der Hochschule der Wirtschaft und Recht im IMB Institut als Abschlussarbeit angefertigt. Die Idee der Arbeit kommt aus der Industrie und ist an den Autor als Auftrag vergeben worden.

An erster Stelle möchte ich mich bei der Leitung des Studiengangs und speziell bei Prof. Anja Grohte bedanken, dass sie mir das Studium in diesem berufsbegleitenden Masterprogramm ermöglicht hat.

Bei Herrn Prof. J. P. Sondermann als Betreuer dieser Abschlussarbeit möchte ich mich herzlich bedanken. Durch seine Anleitung und Veranschaulichungen aus mehr als 30 Jahren Erfahrung in der Automobilindustrie habe ich sehr viel gelernt. Das gilt für das Studium und die Betreuung der Arbeit.

Mein Dank gebührt Dr. Kallow, Geschäftsführer von TeCNeT GmbH, der den Auftrag vergeben hat, diese Thematik zu untersuchen und als Masterarbeit einzureichen.

Meiner Familie, Freunden und Bekannten danke ich herzlich, da sie mich während des zweijährigen berufsbegleitenden Studiums unterstützt haben, sowohl im sozialen Umfeld als auch in finanzieller Hinsicht.

Diese Arbeit widme ich meinem verstorbenen Vater Awni Karomi, der immer ein Vorbild für mich war.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Technische Grundlagen

Softwaretests gehören zum festen Bestandteil der Qualitätssicherungsmaßnahmen bei der Entwicklung sicherheitskritischer Software. Sicherheitskritische Software ist die Bezeichnung für jenen Teil der Software im Fahrzeug oder bei medizinischen Produkten, die für den Menschen lebensgefährlich werden, wenn ein Fehler in der Software enthalten ist.

Softwarefehler wurden in den letzten Jahren vor allem über die Medien bekannt, die über Probleme an Autos berichteten. Die Welt Online schreibt zum Beispiel am 05.09.2011, dass der Hersteller Honda weltweit fast eine Million Autos zurückruft.[1] Darunter ist auch das Modell CR-Z, das in Deutschland verkauft wird. Eines der vielen Probleme an diesem Auto war die fehlerhafte Software im Steuergerät.

Aus der Medizintechnik gibt es ähnliche Berichte. Die Programmierung des Therac-25 - eines computergesteuerten Bestrahlungsgerätes zur Behandlung von Tumoren - enthielt einen Softwarefehler. Das Gerät wurde von Juni 1985 bis Januar 1987 eingesetzt. Der Softwarefehler, aber auch das fahrlässige Verhalten bei der Fehlersuche und Fehlerbeseitigung führten zu sechs Todesfällen.[2]

Softwaretestmaßnahmen in der Qualitätssicherung dienen dazu, solche Fehler im Endprodukt zu verhindern. Der Test wird oft als destruktiv bezeichnet. Ziel ist dabei stets, die entwickelte Software grundlegend zu überprüfen. In diesem Kapitel werden die Grundlagen des Testens von Software erläutert. Der Fokus liegt jedoch auf der Software in Steuergeräten.

[1] Vgl. Axel Springer (2012)
[2] Vgl. Beckert (2012)

1.1 Bedeutung der Softwarequalitätssicherung

Qualitätssicherungsmaßnahmen unterteilen sich in drei wichtige Bereiche: organisatorische Maßnahmen, analytische Maßnahmen und konstruktive Maßnahmen.[3] Zu den organisatorischen Maßnahmen gehören die Aufbau- und die Ablauforganisation, die man in der ISO Norm 9001 findet. Diese Maßnahmen schaffen die Voraussetzungen dafür, dass analytische und konstruktive Maßnahmen durchführbar und wirksam sind. Die analytischen Maßnahmen untersuchen das Prüfteil (Testobjekt oder im englischen Device under Test DUT), um Fehler zu finden. Die analytischen Maßnahmen wirken indirekt auf die Softwarequalität, wenn die gefundenen Fehler anschließend beseitigt werden. Das Testen gehört zu den Verfahren der analytischen Maßnahmen. Die konstruktiven Maßnahmen sind ebenso wichtig, aber wenig bekannt. Man möchte Fehler nicht erst entstehen lassen, um sie dann analytisch suchen und entfernen zu müssen.

Jedes Produkt auf dem Markt muss die vom Kunden geforderte Qualität erfüllen, sonst hat es kaum Chancen, vertrieben werden zu können. Eine Herausforderung bei der Entwicklung industrieller Software liegt darin, die Anforderungen an die Qualität zu bestimmen. Die Softwarequalität gehört zu den Gebieten, die sehr herausfordernd sind. Ein Mangel an Qualität der Software bedeutet immer, dass Kunden abwandern, Vertrauen verspielt und Geld verloren wird. Finanzielle und sicherheitskritische Anwendungen können auch gesundheitliche Schäden zur Folge haben.[4]

Das Airbag-System im Auto soll die Insassen vor den Folgen eines Zusammenstoßes schützen. Eine Qualitätsanforderung an dieses Produkt ist die rechtzeitige Erkennung des Unfalls und die sofortige Zündung des Airbags. In der industriellen Qualitätssicherung macht man einen stichprobenartigen Test, um diese wichtige Qualität der Sicherungsanforderungen zu prüfen. Dieser Gesamtsystemtest soll die Qualität des Produktes bewerten, d. h. ob das Produkt in die Serienproduktion gehen kann.

[3] Vgl. Schneider (2007), S. 173.
[4] Vgl. Schneider (2007), S. 7.

Das Airbag-System besteht aus Hardware- und Softwarekomponenten. Die Software spielt eine immer größere Rolle bei der Entwicklung sicherheitskritischer Anwendungen.

Im Bereich der Softwareentwicklung bedeutet dies: „Software-Qualität ist die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.“[5] Allgemein kann man die Anforderungen oder Eigenschaften einer Software in Bezug auf ihre Qualität wie folgt unterteilen:[6]

- Funktionalität: Was muss die Software anbieten?
- Effizienz (Ressourcenverbrauch, Serverauslastung): Wie viel Rechenleistung, Arbeitsspeicher oder Festplattenspeicher benötigt die entwickelte Software?
- Zuverlässigkeit: Wann ist die Software verfügbar und das System im Betrieb stabil?
- Änderbarkeit (Anpassbarkeit): Können bei veränderten Prozessen (Hardware Plattformen) im Unternehmen Anpassungen an der Software vorgenommen werden?
- Benutzbarkeit (Bedienbarkeit): Kann ein Bediener die Software ohne übermäßig viele Schulungen möglichst intuitiv bedienen?
- Fehlertoleranz (Vorhandensein von Fehlermeldungen): Stürzt die Software ab oder werden falsche Ergebnisse ausgegeben?
- Wiederverwendbarkeit: Kann die Software in der Zukunft für weitere Projekte verwendet werden?
- Testbarkeit: Können die Software und die einzelnen Softwaremodule getestet werden, um die Qualität der Software zu überprüfen?

[5] Vgl. Vigenschow (2004), S. 15.
[6] Vgl. Drescher (2010), S. 6.

- Plattformunabhängigkeit: Kann die Software fehlerfrei auf verschiedenen Betriebssystemen (Basisplattformen) laufen und ihre Anforderungen erfüllen?

Diese Eigenschaften können durch Tests überprüft werden. Dafür werden Prozesse eingesetzt und Testrollen in Entwicklungsteams verteilt. In mehreren aufeinanderfolgenden Testphasen während der Softwareentwicklung werden diese Tests parallel zur Entwicklung durchgeführt.[7]

1.2 Softwaretest und ISO 9000 Normreihe

Die Normreihe 9000 beinhaltet die Anforderungen an das Managementsystem, die ein Unternehmen aufbringen muss, um einem bestimmten Standard bei der Umsetzung des Qualitätsmanagements zu entsprechen.

Ein Unternehmen erfüllt die Anforderungen der Norm informativ und zum Nachweis bestimmter Standards gegenüber Dritten. Ein Zertifizierungsprozess ermöglicht einen Nachweis mit anschließender Ausstellung eines zeitlich befristeten Zertifikats durch unabhängige Zertifizierungsstellen.

Die acht Grundsätze des Qualitätsmanagements sind:[8]

- Kundenorientierung,
- Verantwortlichkeit der Führung,
- Einbeziehung der beteiligten Personen,
- prozessorientierter Ansatz,
- systemorientierter Managementansatz,
- kontinuierliche Verbesserung,
- sachbezogener Entscheidungsfindungsansatz,
- Lieferantenbeziehungen zum gegenseitigen Nutzen.

Im prozessorientierten Ansatz wird davon ausgegangen, dass ein Input in einen Output umgewandelt wird. Die Norm betrachtet diese Prozesse und vergleicht die Soll-Vorgaben mit den Ist-Werten.

---

[7] Vgl. Teichreber (2008), S. 42 ff.
[8] Vgl. Thaller (2001), S. 41.

Bei Abweichungen werden Verbesserungen und Veränderungen definiert und geplant. Somit schließt sich der Kreis Plan – Do – Check – Act, auch PDCA-Zyklus genannt.[9]

Die acht Hauptkapitel der Norm sind in der Norm ISO 9001 zu finden. Softwaretests sind Aufgabe eines Unternehmens, das Software bzw. Embedded Systems entwickelt. Das Kapitel „Messung, Analyse und Verbesserung“ der Norm ISO 9001 beschreibt Vorgehen und Methoden, die zur Risikominderung eines Softwareproduktes im Prozess der Entwicklung genutzt werden können. Prozessüberwachung, Produktüberwachung, kontinuierliche Verbesserung und Fehlerbeseitigung sind im Kapitel der Norm beschrieben, um hohe Kundenzufriedenheit zu erlangen. Der Softwaretest gehört zu dieser Reihe der Methoden.

1.3 Grundlagen des Softwaretests

Voraussetzung des Softwaretestbegriffs ist der Begriff des Fehlers in Softwaresystemen. Ein Fehler ist die Nichterfüllung einer festgelegten Anforderung an das Softwaresystem. Es liegt eine Abweichung zwischen dem Ist-Verhalten und dem Soll-Verhalten vor. Das Ist-Verhalten wird während des Betriebs des Softwaresystems festgestellt. Das Soll-Verhalten wird in der Spezifikation der Testaufgaben fest verankert. Fehler in Softwaresystemen entstehen im Gegensatz zu jenen in physikalischen Systemen nicht durch Verschleiß oder Alterung, sondern während der Entwicklung der Software. Die Fehler bleiben erhalten und kommen bei der Ausführung des Systems zum Tragen. Das Testen hat die Aufgabe, Risiken beim Einsatz der Software zu verringern. Das geschieht dadurch, dass beim Testen möglichst alle Fehler aufgedeckt werden.[10]

Das Ziel des Testens ist die Qualitätsverbesserung durch Beheben eines Fehlerzustands. Testen erfolgt durch das Ausführen eines Programms im Steuergerät. Man braucht dafür einen Testfall und einen Testplan. Ein Testfall hat die Aufgabe festzulegen, welche Eingaben in das Programm gemacht werden und was das Ergebnis sein soll. Man braucht viele Testfälle, um das gesamte

[9] Vgl. Thaller (2001), S. 65.
[10] Vgl. Andreas/Tilo (2006), S. 7.

Programm im Steuergerät abzudecken.[11] Testplan oder auch Testkonzept werden meistens nach der Norm IEEE 829 aufgestellt.[12] Der Inhalt eines Testkonzepts nach IEEE 829 besteht aus folgenden Punkten:[13]

Testkonzeptbezeichnung

Das Testkonzept muss gekennzeichnet werden, um sicherzustellen, dass das Dokument gegenüber anderen Dokumenten klar erkennbar ist. Die Benennung der Dokumente ist in jeder Firma archiviert und gemäß Maßgaben und Regeln festgelegt. Das Dokument muss im Dokumentmanagementsystem abgelegt sein und mindestens Dokumentname, Version und Freigabestatus enthalten.

Einführung

In der Einführung sind Projekt und Hintergründe kurz beschrieben. Die Projektbeteiligten (Kunde, Tester, Testmanager und weitere Rollen) sind dargestellt. Referenzen auf weitere Dokumente, Normen, Standards und Unternehmensregeln müssen enthalten sein.

Testobjekte

Hier sollten die zu testenden Softwarekomponenten und Teile des Systems, die nicht getestet werden, kurz dargestellt sein.

Zu testende Leistungsmerkmale:

Dieses Unterkapitel des Testkonzepts berücksichtigt Funktionen und Eigenschaften, die getestet werden soll. Der Verweis auf die Testspezifikation und die Teststufen muss enthalten sein.

Leistungsmerkmale, die nicht getestet werden:

Die Aspekte in der Software, die nicht getestet werden sollen, müssen hier aufgeführt werden, um unberechtigten Erwartungen vorzubeugen.

[11] Vgl. Kurt (2007), S. 83.
[12] Vgl. Andreas/Tilo (2006), S. 223.
[13] Vgl. Andreas/Tilo (2006), S. 223.

Teststrategie

Die Testziele werden auf Basis einer Risikoanalyse definiert. Testmethoden und notwendige Automatisierungen der Testaufgaben werden festgelegt, die ausgewählten Strategien auf vorhandene Ressourcen geprüft und identifizierte Risiken abgewogen.

Abnahmekriterien

Das Entwicklungsteam muss anhand der durchgeführten Tests entscheiden, ob die implementierte Software an den Kunden geliefert werden soll oder nicht. Eine kurze Ansage, dass die Software fehlerfrei ist, reicht hierbei nicht aus. Kriterien wie „Anzahl der gefundenen Fehler“, „Schweregrad der gefundenen Fehler“ und „Entstandene Kosten“ werden hier herangezogen.

Kriterien für Testabbruch und Testfortsetzung

Es kann vorkommen, dass die Software bei den ersten Tests so schlecht abschneidet, dass weitere Tests sinnlos erscheinen. An dieser Stelle muss überlegt werden, welche Kriterien ausgewählt werden, um Testabläufe zum Abbruch zu bringen.

Testdokumentation

Hier wird entschieden, wann und welche Testdokumente und Ergebnisse an wen kommuniziert werden müssen.

Testaufgaben

Die Zuordnung der Verantwortlichkeiten sowie die Planung und Durchführung der Testaktivitäten werden festgelegt.

Testinfrastruktur

Testwerkzeuge, Arbeitsmittel und Testinfrastruktur müssen aufgelistet werden. Wenn für die Testautomatisierung besondere Werkzeuge und Mittel benötigt werden, müssen diese aufgelistet sein.

Verantwortlichkeiten/Zuständigkeiten

Die organisatorische Einbindung des Testpersonals in das Projekt und die Aufteilung in verschiedene Testgruppen und Teststufen muss geplant werden.

Personal, Einarbeitung, Ausbildung

Die Ressourcen werden geplant. Die Personalausstattung und die Qualifikation werden bestimmt und das Administrationspersonal wird eingewiesen.

Zeitplan/Arbeitsplan

Die Testaktivitäten werden mit Meilensteinen geplant und an den Projektmanager kommuniziert.

Genehmigung/Freigabe

Die Organisationseinheiten, die das Testkonzept genehmigen müssen, werden aufgelistet. Die Zustimmung erfolgt durch die Unterschrift der Führungsebene einer Organisationseinheit.

Glossar

Im Testbereich werden viele verschiedene Begriffe genutzt. Diese sollten in einem Glossar erfasst sein. Im Projekt sollten immer die Test-Fachbegriffe verwendet werden, um unterschiedliche Interpretationen für einen Fachbegriff zu vermeiden.

Die Einordnung der Testaktivitäten in Softwareentwicklungsprozesse wird anhand des V-Modells des Bundes dargestellt (siehe Abbildung 1). Die Grundidee des V-Modells ist, dass Testaktivitäten und Entwicklungsaktivitäten zueinander korrespondierende, gleichberechtigte Tätigkeiten sind.[14] Der linke Ast veranschaulicht die Entwicklungsschritte. Über die Kundenwünsche werden die Anforderungen entworfen, spezifiziert und dann implementiert. Der rechte Ast steht für Integrations- und Testarbeiten. Der Verlauf beschreibt die Integration

[14] Vgl. Andreas/Tilo (2006), S. 39.

und den Test der Bausteine sukzessiv zum Wachsen des Gesamtsystems. Der linke Ast erfasst die Verifikationen nach jedem fertigen Schritt. Parallel dazu validiert der Tester, ob ein Teilprodukt eine festgelegte spezifizierte Aufgabe tatsächlich löst.

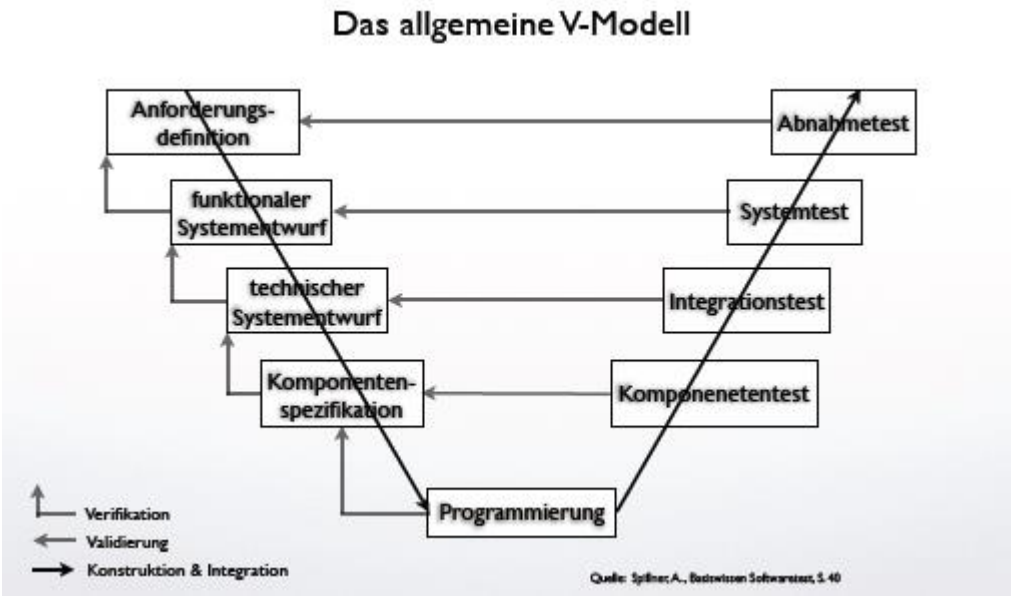

Abbildung 1: V-Modell des Bundes und Testaktivitäten[15]

1.4 Fundamentaler Testprozess

Ein Softwaretest unterliegt einem bestimmten Prozess, der in Abbildung 2 veranschaulicht wird. Der Testmanager hat die Aufgabe, die Tests zu planen und zu steuern. Die Steuerung basiert auf regelmäßigen Treffen des Testteams und der Kontrolle der Testergebnisse. Der Testplan muss vom Testmanager erstellt werden. Die Strategie des Testens und die Steuerung werden im Testplan dokumentiert. Die Testautomatisierung wird als Strategie vom Testmanager angefordert. Diese Aufgaben (Steuerung, Planung) laufen parallel zu allen anderen Testaufgaben im Projekt und müssen immer aktuell gehalten und dokumentiert werden.

[15] Vgl. Andreas/Tilo (2006), S. 40.

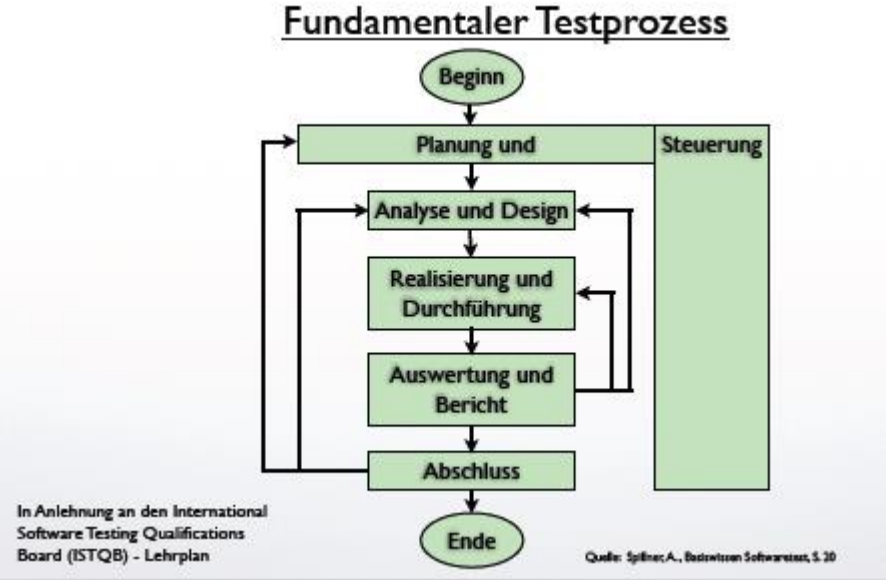

Abbildung 2: Fundamentaler Testprozess[16]

Der Softwaretest-Designer hat die Aufgabe, die Testobjekte zu analysieren, um aus dem Test Artefakte, logische und konkrete Testfälle zu spezifizieren. Die Testrealisierung und Durchführung wird vom Softwaretester erledigt und die Auswertung an das Testteam berichtet.

1.5 Softwareteststufen

Das Softwaretesten in Entwicklungsprojekten läuft laut des V-Modells parallel zu den konstruktiven Entwicklungsaufgaben des Softwareentwicklers. Die Unterteilung des Tests nach dem V-Modell in verschiedene Stufen hat viele Vorteile: Je früher ein Fehler in den Stufen des rechten Astes gefunden wird, desto preiswerter ist seine Behebung. Die Testaufgaben werden besser organisiert. Jede Teststufe im rechten Ast des V-Modells hat ihre korrespondierende Entwicklungsstufe im linken Ast. Dadurch können Fehler in Bezug auf die Entwicklungsstufen gefunden werden. Das zu testende Kriterium kann für jede Stufe besser definiert werden. Die Teststufen im rechten Ast des V-Modells haben bestimmte Zielkriterien und werden wie folgt beschrieben:

[16] Vgl. Andreas/Tilo (2006), S. 20.

Der **Komponententest** oder Unittest wird auch Modultest genannt. Dieser Test findet auf der Ebene der einzelnen Module der Gesamtsoftware statt. Die Anforderungen an das Gesamtsystem werden als kleine Softwaremodule auf die Entwickler im Team verteilt. Testgegenstand ist die Funktionalität innerhalb einzelner abgrenzbarer Teile der Software (Module, Programme oder Unterprogramme, Units). Diese Stufe wird meistens durch den Softwareentwickler selbst durchgeführt. Ziel ist der Nachweis der technischen Lauffähigkeit und korrekter fachlicher (Teil-)Ergebnisse. Auf dieser Teststufe ist die Testautomatisierung ein notwendiges Kriterium. Dies ist darin begründet, dass sich die Software im Laufe des Projekts ständig ändert und alte und neue Versionen mit den gleichen Testfällen überprüft werden müssen (Regressionstests).[17]

Der **Integrationstest** läuft nach dem Komponententest und hat die Aufgabe, die Zusammenarbeit der Komponenten zu testen, die voneinander abhängig sind. Der Testschwerpunkt liegt in dieser Stufe auf den Schnittstellen der beteiligten Komponenten. Diese Stufe soll korrekte Ergebnisse über komplette Abläufe hinweg nachweisen.[18]

Beim **Systemtest** wird das Gesamtsystem betrachtet. Das System wird auf die Systemanforderungen hin (funktionale und nicht-funktionale Anforderungen) getestet. Eine eigene Testumgebung wird für diese Teststufe benötigt. Für die Umgebung müssen reale Testdaten zur Verfügung stehen, um einen Echtzeitbetrieb zu simulieren. Die Testumgebung simuliert die Produktivumgebung des Kunden.[19]

Ein **Abnahmetest** wird auch Akzeptanztest genannt. Dieser Test wird durch den Kunden durchgeführt. Der Auftraggeber testet die gelieferte Software. Diese Tests bilden die Voraussetzungen für die Rechnungsstellung und können bereits auf der Produktivumgebung durchgeführt werden. Die Testdaten können aus Echtdaten des Systemtests abgeleitet werden. Die Testspezifikationen für diese Teststufe können vom Auftraggeber erstellt werden. Oft sind diese Tests

[17] Vgl. Andreas/Tilo (2006), S. 42.
[18] Vgl. Andreas/Tilo (2006), S. 50.
[19] Vgl. Andreas/Tilo (2006), S. 58.

im Vertrag verankert. Nur wenn die Software diese Tests besteht, kann ein Auftrag oder ein weiterer Software-Release freigegeben werden.[20] Die genannten Teststufen sind in der Praxis sehr schwer voneinander zu trennen. Sie können abhängig von der Projektsituation ineinander übergehen oder durch weitere Teststufen verfeinert werden. Sie laufen entweder parallel oder sequentiell nacheinander. Vor allem die Systemtestebene und die Abnahmetests können schwer voneinander getrennt werden. Meistens kann die Abnahme des Systems auf Grundlage der Testergebnisse und Protokolle des Systemtests erfolgen. Eine wichtige Unterscheidung im Softwaretest-Kontext findet sich in der Trennung zwischen White-Box-Verfahren und Black-Box-Verfahren. Bei der White-Box-Variante spielt der Source Code eine wichtige Rolle im Test, durch den man in das Systeminnere schauen und auf die Artefakte der Entwicklung bis in die tiefste Ebene zurückgreifen kann. Die White-Box-Verfahren werden in den tieferen Teststufen eingesetzt. Für System- und Abnahmetests wird das Black-Box-Verfahren angewendet, d. h. der Test orientiert sich nicht am Code der Software, sondern nur am Verhalten der Software bei spezifizierten Anforderungen an das Gesamtsystem (Eingaben des Benutzers, Grenzwerte bei der Datenerfassung etc.). Dazu unterscheidet man zwischen verschiedenen Testarten. Folgende Testarten werden bei den einzelnen Teststufen im rechten Ast des V-Modells favorisiert und gemäß der Anforderungen und des Projektablaufs unterschiedlich eingesetzt:[21]

- Funktionaler Test: Die Funktionalität der Software steht im Mittelpunkt, Ein- und Ausgabeverhalten wird geprüft.
- Nicht-funktionaler Test: Die Merkmale nach ISO 9126 werden geprüft: Zuverlässigkeit, Effizienz und Benutzerbarkeit.
- Änderungsbezogener Test oder Regressionstests: Ein bereits getestetes Programm wird nach dessen Modifikation erneut getestet, um nachzuweisen, dass durch die vorgenommene Änderungen keine neuen Defekte eingebaut wurden.

[20] Vgl. Andreas/Tilo (2006), S. 61.
[21] Vgl. Andreas/Tilo (2006), S. 69.

Im Rahmen dieser Arbeit werden die analytischen Maßnahmen näher untersucht. Dabei wird der Fokus auf die Automatisierung dieser Maßnahmen als Testautomatisierung gelegt. Die dargestellten Konzepte zur Automatisierung setzen die Rahmenbedingungen für die Entwicklung von sicherheitskritischen Anwendungen im speziellen Gebiet der eingebetteten Systeme. Dabei handelt es sich um Rechnereinheiten, die kleine präzise Aufgaben im Produkt erledigen. Solche Systeme werden zunehmend in Fahrzeugen und in medizinischen Produkten eingesetzt, um eine bestimmte Aufgabe zu erfüllen. Das eingebettete System arbeitet nach dem EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe). Dabei werden Umwelterscheinungen in analoger Form digitalisiert und verarbeitet, um bestimmte Aktoren im Umfeld zu steuern.

2 Softwaretest und Testautomatisierung aus der Sicht der funktionalen Sicherheit und Reifegradmodelle

Der Anteil sicherheitsbezogener elektronischer Steuerungen und der Software in Produkten ist in den vergangenen Jahren stark gestiegen. Unfälle und Gefahren können zum einen durch menschliches Versagen und zum anderen durch technische Defekte ausgelöst werden. Um die gesetzlichen Anforderungen an die Sicherheit von Produkten erfüllen zu können, ist die Anwendung von Sicherheitsnormen unumgänglich. Prozess-Modelle wie CMMI oder SPICE können eine Orientierung geben, um die Sicherheitsziele auf effiziente Art und Weise zu erreichen.[22]

Softwaretests gehören zu einem Bereich, in dem heute Gesetze, Standards und Normen entwickelt werden. In diesem Kapitel werden diese Gesetze, Normen und Standards veranschaulicht. Der Fokus wird auf die Anforderungen gelegt, die aus der Sicht der Normen und Standards für den Softwaretest gelten. Angaben in der recherchierten Literatur und Erfahrungen aus der Industrie werden zusammengeführt, um die Anforderungen für die Testautomatisierung aus Sicht der Normen und Standards abzuleiten.

2.1 Gesetze und Richtlinien

Das Testen von Software in Produkten (eingebetteten Systemen) spielt eine entscheidende Rolle in der Beurteilung der Produktsicherheit. In diesem Abschnitt wird ein Überblick über grundsätzliche Anforderungen der EU-Richtlinien und Gesetze zur Produktsicherheit gegeben.

Die Richtlinie über die allgemeine Produktsicherheit 2001/95/EC (GPSD) ist am 15. Januar 2002 in Kraft getreten. Das Ziel der Richtlinie ist der Schutz der Sicherheit und Gesundheit der Verbraucher vor den Gefahren durch Produkte. Sie fordert: Hersteller dürfen nur sichere Produkte in den Verkehr bringen. Die Produkte sollen mit einem Standard entwickelt werden, so dass der Stand der

[22] Vgl. Löw (2010), S. 2.

Technik, des Wissens und der Sicherheit, die der Verbraucher erwartet, beurteilt werden kann. In dieser Richtlinie wird auf die europäischen und internationalen Normen verwiesen und deren Anwendung erwartet. Wenn der Hersteller diese Normen einhält, gehen die Behörden davon aus, dass er die Richtlinie einhält.[23]

Die Richtlinie über die allgemeine Produktsicherheit (GPSD) ist in Deutschland am 06. 01.2004 in ein nationales Recht umgesetzt worden. Das „Geräte- und Produktsicherheitsgesetz“ ist am 01.05.2004 in Kraft getreten.

Es ist wichtig zu wissen, dass im Schadensfall die Nachweispflicht beim Hersteller bzw. Inverkehrbringer liegt. Bei Nichtbeachten drohen Geldstrafen oder Freiheitsstrafen, je nach Schwere und wiederholten Verstößen gegenüber den Verbrauchern. Nicht nur Unternehmen sind einem Risiko ausgesetzt, sondern auch die Mitarbeiter. Dies ist im BGB§ 823 dokumentiert.[24]

2.2 Funktionale Sicherheit und Softwaretest

Um die gesetzlichen Vorschriften einzuhalten, werden Normen herangezogen, um nachzuweisen, dass die Produktsicherheit nach dem Stand der Technik in das Produkt integriert wurde.

Die deutsche Norm DIN EN 61508 „Funktionale Sicherheit sicherheitsbezogener elektrischer/ elektronischer/ programmierbarer elektronischer Systeme“ wurde von der internationalen Norm IEC 61508 übernommen. Die Norm stellt Anforderungen mit dem Ziel, definierte Sicherheitsziele zu erreichen, d. h. das Restrisiko, das vom System ausgeht zu mindern. Die Norm ist in sieben Teile gegliedert:[25]

- Teil 1: Allgemeine Anforderungen (IEC 61508-1:1998 + Corrigendum 1999)
- Teil 2: Anforderungen an sicherheitsbezogene elektrische/elektronische/programmierbare elektronische Systeme (IEC 61508-2:2000)

[23] Vgl. Löw (2010), S. 47.
[24] Vgl. Löw (2010), S. 48.
[25] Vgl. Löw (2010), S. 8.

- Teil 3: Anforderungen an Software (IEC 61508-3:1998 + Corrigendum 1999)
- Teil 4: Begriffe und Abkürzungen (IEC 61508-4:1998 + Corrigendum 1999)
- Teil 5: Beispiele zur Ermittlung der Stufe der Sicherheitsintegrität (safety integrity level) (IEC 61508- 5:1998 + Corrigendum 1999)
- Teil 6: Anwendungsrichtlinie für IEC 61508-2 und IEC 61508-3 (IEC 61508-6:2000)
- Teil 7: Anwendungshinweise über Verfahren und Maßnahmen (IEC 61508-7:2000)

Die Norm dient als Basis für anwendungsspezifische Normen. Folgende anwendungsspezifische Normen sind aus der Grundnorm IEC 61508 abgeleitet. Sie stellen die Implementierung der Grundnorm in bestimmten Anwendungsgebieten dar.[26]

- IEC 61511: Funktionale Sicherheit — Sicherheitstechnische Systeme für die Prozessindustrie
- IEC 61513: Kernkraftwerke — Leittechnik für Systeme mit sicherheitstechnischer Bedeutung – Allgemeine Systemanforderungen
- EN 50128: Bahnanwendungen — Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Sicherheitsrelevante elektronische Systeme für Signaltechnik
- IEC 62061: Sicherheit von Maschinen — Funktionale Sicherheit sicherheitsbezogener elektrischer, elektronischer und programmierbarer elektronischer Steuerungssysteme
- IEC 60601: Medizinische elektrische Geräte - Allgemeine Festlegungen für die Sicherheit
- ISO 26262: Road vehicles – Functional safety

Die Norm definiert eine grundsätzliche Vorgehensweise bei der Risikominderung. Die Vorgehensweise besteht darin, gefährliche Ausfälle des Systems zu vermeiden oder zu beherrschen. Diese Ausfälle entstehen durch systematische Fehler oder durch zufällige Hardwareausfälle. „Systematische Fehler sind Fehler, die aufgrund menschlichen Versagens in den verschiedenen Stadien des Lebenszyklus entstehen. Dazu zählen Spezifikationsfehler, Entwurfsfehler, Implementierungsfehler und Installations- oder Bedienungsfehler. Zufällige Hardwareausfälle sind das Ergebnis der begrenzten Zuverlässigkeit von Hardwarebauteilen.“[27]

[26] Vgl. Löw (2010), S. 9.
[27] Vgl. Löw (2010), S. 9.

Die Sicherheitsziele der Norm werden durch die folgenden Anforderungen an eine systematische Vorgehensweise erreicht:

- Eine Risikoanalyse und Spezifikation der Sicherheitsanforderungen muss durchgeführt werden. Welche Risikominderung ist erforderlich?
- Sicherstellung einer vollständigen und nachweisbaren Umsetzung der Sicherheitsanforderungen durch ein Management der Aktivitäten im Sicherheitslebenszyklus.
- Vermeidung und Beherrschung der Fehler durch den Entwurf der Hardware- und Softwarearchitektur nach vorgegebenen Prinzipien.
- Die Entwicklung des Systems muss geplant und nachvollziehbar über definierte Prozesse (Projektmanagement und Testmanagement) sein.
- Anwendung von bestimmten Techniken und Maßnahmen zur Vermeidung und Erkennung von systematischen Fehlern (z. B. Testverfahren).

Die abgeleiteten branchenspezifischen Normen folgen in der Vorgehensweise der Grundnorm. Es ist empfehlenswert, die branchenspezifische Norm einzusetzen, da dann weniger Aufwand bei der Interpretation der Grundnorm und ihrer Anforderungen entsteht. Die Norm stellt frei, ob die Testautomatisierung im Entwicklungsprojekt eingesetzt wird oder nicht. Vielmehr werden Techniken und Methoden der Testverfahren vorgeschlagen, die je nach Sicherheitsstufe eingesetzt werden müssen.

2.3 Anforderung der Norm ISO/DIS 26262 (Personenkraftwagen)

Diese Norm stellt eine Interpretation des Sicherheitslebenszyklus nach der Grundnorm für den Bereich Automotiv dar. Ziel ist, eine systematische und konsistente Vorgehensweise zu etablieren, die für alle Phasen des in der Automobilindustrie üblichen Lebenszyklus anwendbar ist.[28] Die Norm besteht aus zehn Teilen, die folgenden Inhalt abdecken:[29]

[28] Vgl. Löw (2010), S. 118.
[29] Vgl. Löw (2010), S. 119.

- Vokabular,
- Management der funktionalen Sicherheit,
- Konzeptphase,
- Produktentwicklung: Systemebene,
- Produktentwicklung: Hardwareebene,
- Produktentwicklung: Softwareebene,
- Produktion, Betrieb und Außerbetriebnahme,
- unterstützende Prozesse,
- ASIL- und sicherheitsorientierte Analysen,
- Guideline (nur informativ).

Auf der Softwareebene im sechsten Teil werden Anforderungen an die Entwicklung der Software und deren Test vorgegeben. Abhängig von der Sicherheitsstufe werden bestimmte Testverfahren vorgeschlagen. Der Sicherheitslebenszyklus ist im Vergleich zur Grundnorm auf die Serienproduktion im Bereich Automotiv zugeschnitten. Dabei werden konkrete Arbeitsprodukte im Vergleich zur Grundnorm dargestellt.

2.4 Anforderung der Norm DIN EN 60601 (Medizingeräte)

Die Normreihe DIN EN 60601 betrachtet die medizinischen elektrischen Geräte. Sie besteht aus zwei Hauptabschnitten: Der Hauptabschnitt Teil 1 beinhaltet die „Allgemeine Festlegung für die Sicherheit“ und betrachtet die wesentlichen Leistungsmerkmale von medizinischen elektrischen Geräten. Der zweite Abschnitt ist in 52 Normen unterteilt und enthält besondere Festlegungen für die Sicherheit einzelner medizinischer Geräte, wie Beatmungsgeräte, Defibrillatoren und Röntgengeräte.[30]

[30] Vgl. Löw (2010), S. 174.

Die Norm verlangt ein Risikomanagement. Ziel ist, konzeptionelle und konstruktive Maßnahmen einzuleiten, um die Risiken des Gerätes zu minimieren oder zu beseitigen. Die Norm verlangt, dass die Geräte so ausgelegt werden, dass beim Auftreten eines Fehlers die Sicherheit erhalten bleibt oder das Restrisiko auf ein tolerierbares Maß beschränkt bleibt. Die Risikoanalyse dient dazu, Fehler zu identifizieren, die eine Gefahrensituation verursachen.

Tests sind verbindlich und sollen in diesem Fall dazu dienen, diese Situationen zu prüfen und nachzuweisen, dass das Gerät ein tolerierbares Maß von Restrisiko hat. Die Norm verlangt für programmierbare Geräte ein hohes Maß an Tests und Dokumentationen. Eine konkrete Umsetzung einer Testautomatisierung verlangt die Norm nicht. Es empfehlt sich daher, die Definition von Prozessen und deren praktische Umsetzung nach dem CMMI (Capability Maturity Model Integration) und nach der Sicherheitsnorm DIN EN 61508.[31]

CMMI wurde vom Software Engineering Institute (SEI) der Carnegie Mellon University entwickelt. CMMI löste das CMM der Version 1.1 im Jahre 2002 ab. Beide Modelle stammen von SEI. CMMI gehört der Gruppe der Assessmentmodelle an und folgt der Annahme, dass eine Verbesserung der Prozesse der Softwareerstellung zu höherer Qualität der entwickelten Software als auch zur genaueren Planung von Terminen und Ressourcen führt.[32]

Für CMMI gibt es vier Anwendungsgebiete:

- Systementwicklung (CMMI-SE),
- Softwareentwicklung (CMMI-SW),
- Integrierte Prozess- und Produktentwicklung (CMMI-IPPD),
- Beschaffung über Lieferanten (CMMI-SS).

[31] Vgl. Löw (2010), S. 174 ff.
[32] Vgl. Spillner (2011), S. 8.

CMMI unterscheidet zwischen einer stufenförmigen und einer kontinuierlichen Darstellung. Die stufenförmige Darstellung definiert fünf Reifegradstufen.

Stufe 1- initial: Prozesse sind nicht oder nur unzureichend definiert.
Stufe 2 - gemanagt: Die Prozesse des Managements sind etabliert und werden angewendet.
Stufe 3 - definiert: Die Prozesse sind organisationsweit einheitlich eingeführt.
Stufe 4 - quantitativ gemanagt: Aufgrund von Kennzahlen werden die Prozesse verbessert.
Stufe 5 - optimierend: Die Verbesserung der Prozesse wird kontinuierlich und systematisch durchgeführt.

In der kontinuierlichen Darstellung wird jedem Prozess eine Fähigkeitsstufe zugeordnet, so genannte Prozessgebiete. Jedes Gebiet hat eine Reihe von Zielen. Die Ziele sind zum einen spezifisch und zum anderen generisch. Die generischen Ziele stellen die Aufgaben dar, die notwendig sind, um die spezifischen Ziele dauerhaft und effizient umzusetzen. Die Prozessgebiete und deren Reifegradstufen haben eine bestimmte Zuordnung. Die Prozessgebiete werden den folgenden Kategorien zugeordnet:

- Prozessmanagement,
- Projektmanagement,
- Ingenieursdisziplin,
- Unterstützung.

Von entscheidender Bedeutung sind die Prozessgebiete Verifikation und Validation. Beide Prozessgebiete gehören zum Reifegrad 3 (definiert). In der kontinuierlichen Darstellung sind diese der Kategorie Ingenieursdisziplinen zugeordnet. Die Anforderungen an die Verifikation und Validation sind allgemein formuliert. Für die Verifikation werden Review, statischer Code, Analyse und Test vorgeschlagen. Bei der Validation fordert CMMI die Vorbereitung und Durchführung der Validation. Eine konkrete Methode oder die Nutzung von bestimmten Teststrategien wird nicht aufgeführt, aber welche Aktivitäten notwendig sind, um eine Testumgebung einzurichten.

Der Testprozess ist ein Teil des Entwicklungsprozesses von Software. Wenn ein Unternehmen eine hohe Reifegradstufe besitzt, dann profitiert davon auch das Testteam.

2.5 SPICE und Softwaretest

SPICE (Software Process Improvement and Capability Determination) oder ISO/IEC 15504 ist eine Bewertungsmethode, die das CMM als Grundlage nutzt. SPICE hat im Gegensatz zu CMMI nur eine kontinuierliche Darstellung. Es werden Fähigkeitsgrade einzelner Prozesse ermittelt. „Die Norm bildet den Rahmen für eine einheitliche Bewertung der Leistungsfähigkeit einer Organisationseinheit durch Bewertung der gelebten Prozesse.“[33]

Im SPICE sind sechs Fähigkeitsgrade definiert:

- Ist unvollständig (Stufe 0).
- Wird durchgeführt (Stufe 1).
- Wird gemanagt (Stufe 2).
- Ist etabliert (Stufe 3).
- Ist vorhersagbar (Stufe 4).
- Ist selbstoptimierend (Stufe 5).

Für SPICE gibt es verschiedene Prozessgruppen. Hinter den Prozessgruppen verbergen sich mehrere hundert Basisprozesse oder besser Basispraktiken. Ähnlich wie bei CMMI gibt es hier auch generische Praktiken, die allgemeingültig sind.[34] In Bezug auf das Testen sind die Prozesse „Software construction (ENG. 6)“, „Software integration (ENG. 7)“, „Software testing (ENG. 8)“, „System integration (ENG. 9)“ und „System testing (ENG. 10)“ von besonderem Interesse.

[33] Vgl. Spillner (2011), S. 187.
[34] Vgl. Spillner (2011), S. 189.

Die Norm legt fest, dass nach dem V-Modell gearbeitet werden soll, d. h. dass Testaufgaben parallel zum linken Ast durchgeführt und durch unabhängige Entwickler erledigt werden müssen. Auf die Automatisierung wird nicht eingegangen.

Es ist empfehlenswert, die branchenspezifische Norm oder den Prozess einzusetzen, da dann weniger Aufwand in der Interpretation der Anforderungen entsteht. Die vorgestellten Normen und Reifegradmodelle stellen frei, ob die Testautomatisierung im Entwicklungsprojekt eingesetzt wird oder nicht. Vielmehr werden Techniken und Methoden der Testverfahren vorgeschlagen, die je nach Sicherheitsstufe eingesetzt werden müssen. Im nächsten Kapitel wird auf die Konzepte und Werkzeuge der Testautomatisierung eingegangen, die in SPICE und in der Automobilindustrie im Einsatz sind.

3 Testautomatisierungskonzepte

Die Grundlagen des Softwaretestens anhand des Testkonzeptes haben klar gezeigt, dass die Testautomatisierung eine Strategie darstellt. Diese Strategie muss vor Beginn des Testens in einem Projekt klar definiert sein. Es ist Aufgabe des Testmanagers, die Strategie aufzustellen und klar zu charakterisieren, warum diese Strategie, Werkzeuge und Konzepte eingesetzt werden sollten und nicht andere.

Die Normen und Standards geben keine Vorgaben für den Einsatz der Testautomatisierung. Vielmehr werden Anforderungen an die verwendeten Werkzeuge und Testmethoden vorgegeben. Die Testmethoden stellen ein Werkzeug dar, Testfälle zu spezifizieren. Die Automatisierung der Testfälle hängt meistens vom eingesetzten Werkzeug ab. Im Rahmen dieses Kapitels werden Werkzeuge, Konzepte und Methoden aus der Automobilindustrie näher betrachtet, die dazu dienen, die spezifizierten Testfälle zu automatisieren.

3.1 Konzepte, Methoden und Techniken

Es gibt keinen Industriebetrieb, der sich nicht ständig mit der Optimierung des Produktionsprozesses beschäftigt. Vorbild und bestes Beispiel dafür ist die Automobilindustrie. Sie bringt neue Ideen und Ansätze zu den Themen Prozesssteuerung, Produktionsgestaltung und Messung sowie Qualitätsmanagement, die in anderen Industriezweigen eingesetzt werden können. Die Innovationen in diesen Bereichen stehen für das Bestreben, immer besser zu werden, um im globalen Konkurrenzdruck zu bestehen.[35]

Bei der Testautomatisierung von Software für sicherheitskritische Anwendungen agiert die Automobilindustrie als bestes Beispiel und Vorbild für andere Industriezweige. Die technischen Testautomatisierungskonzepte unterliegen keinen allgemeinen Normen, sondern sind abhängig von kommerziellen Herstellern oder Open-Source-Communitys.[36] Dennoch existieren allgemeine Prin-

[35] Vgl. Seidl (2012), S. 1.
[36] Vgl. Seidl (2012), S. 3.

zipien in der Konzeption, Organisation und Durchführung automatisierter Softwaretests. Die Prinzipien basieren auf dem Wunsch der Unternehmen, von spezifischen Werkzeugen unabhängig zu sein. Der Einsatz dieser Prinzipien unterscheidet sich nach dem praktischen Umfeld:

- Einsatzgebiet nach Testart:
  - funktionale Tests,
  - Zuverlässigkeitstests,
  - Test der Benutzerbarkeit,
  - Test der Effizienz,
  - Test der Änderbarkeit,
  - Test der Übertragbarkeit.
- Einsatzgebiet nach Projektart:
  - klassisches Softwareentwicklungsprojekt,
  - Wartungsprojekt und Produktweiterentwicklung,
  - agile Projekte,
  - Migrationsprojekte.
- Einsatzgebiet nach Teststufen:
  - Komponententests,
  - Integrationstests,
  - Systemtests.
- Einsatzgebiet nach Systemart:
  - Desktop-Applikation,
  - Embedded Systems,
  - Webapplikationen,
  - Dynamische GUIs: (Graphische Oberflächen).

Die Testautomatisierung ist je nach Einsatzgebiet mit entsprechenden Kosten und Lernaufwänden verbunden. Daher haben Testautomaten in der Vergangenheit keine Verbreitung gefunden. Die Weiterentwicklung der Werkzeuge und ihre Fähigkeit zur Automatisierung haben dazu beigetragen, die Kosten-Nutzen-Relation zu verbessern.[37]

[37] Vgl. Seidl (2012), S. 5.

3.2 Testautomatisierung nach Teststufen für Embedded Systems

Im Rahmen dieser Arbeit wird das Einsatzgebiet Embedded Systems und die Testautomatisierung nach Teststufen ausgewählt und untersucht. Die folgenden Überlegungen basieren auf den Erfahrungen des Autors, d. h. einer mehrjährigen Tätigkeit in der Softwareentwicklung, und dem Test sicherheitskritischer Anwendungen in der Industrie. Der Fokus des Autors liegt sowohl auf den praktischen Anwendungen als auch auf sicherheitskritischer Software im Bereich der eingebetteten Systeme. Das Kapitel behandelt die Softwaretest-Automatisierung in der Automobilindustrie. Inwieweit dies auf die Medizintechnik übertragbar ist, behandelt Kapitel 5. Dieses Kapitel wird Werkzeuge und Maßnahmen der Testautomatisierung im rechten Ast des V-Modells[38] darstellen. Dabei werden die einzelnen Stufen nach dem SPICE-Modell[39] untergliedert. Die gängigen Werkzeuge der Tests in der Automobilindustrie und die mögliche Testautomatisierung werden veranschaulicht.

Die Thematik wird auf Tests und Softwareentwicklung beschränkt. Der Autor geht davon aus, dass die Softwareentwicklung manuell codiert wird, ohne Nutzung von modellbasierten Werkzeugen. Bei diesen wird erst modelliert und danach der Quellcode auf dem Steuergerät generiert. Es wird vorausgesetzt, dass die einzelnen Stufen des linken Astes des V-Modells schon durchgeführt wurden und man sich auf den parallel laufenden Aktivitäten des rechten Astes befindet. Zudem wird davon ausgegangen, dass sich nur ein Mikrocontroller und dessen Peripherie auf dem Steuergerät befinden. Der Weg von den Anforderungen zum Test wird manuell durchgeführt. Das Kapitel konzentriert sich auf die Automatisierung der schon manuell erstellten Testspezifikation aus den Anforderungsdokumenten. Im rechten Ast des V-Modells werden folgende Stufen näher betrachtet:

---

[38] Das V-Modell gehört zu den Vorgehensmodellen in der Softwareentwicklung, bei denen der Softwareentwicklungsprozess in Phasen organisiert wird. Neben diesen Entwicklungsphasen (Konstruktive Phasen) definiert das V-Modell auch das Vorgehen zur Qualitätssicherung (Testen) phasenweise (Destruktive Phasen)

[39] SPICE (Software Process Improvement and Capability Determination) oder ISO/IEC 15504 gehört zu den internationalen Standards zur Durchführung von Bewertungen (Assessments) von Unternehmensprozessen. Dabei liegt der Schwerpunkt auf der Softwareentwicklung.

- System Test (ENG. 9 und ENG. 10 nach SPICE),
- Software Test (ENG. 8 nach SPICE),
- Software Integrationstest (ENG. 7 nach SPICE),
- Modell-Ebene und Modultest (ENG. 6 nach SPICE).

Um ein besseres Verständnis für die oben genannten Stufen zu erhalten, ist es notwendig, einen Blick auf die Architektur der Stufen und deren Software zu werfen. Wenn man annimmt, dass die Software nur auf einem Steuergerät läuft und das Steuergerät nur aus einem Mikrocontroller und dessen Peripherie besteht, dann sieht die Architektur der Tests auf jeder Ebene folgendermaßen aus:

Die Informationsverarbeitung auf einem Steuergerät verläuft nach dem EVA-Prinzip. Die Software auf dem Steuergerät ist dafür verantwortlich, die Eingaben aus dem Umfeld zu verarbeiten und in Ausgaben zur Verfügung zu stellen. Die Software wird immer in Module zergliedert, um den Aufwand zu reduzieren und gleichzeitig durch mehrere Entwickler und Tester implementieren und testen zu lassen. Abbildung 3 zeigt die allgemeine Architektur der Software. Die einzelnen Teststufen organisieren sich auf der oben dargestellten allgemeinen Architektur eines eingebetteten Systems. Auf der Systemtest-Ebene kann die Architektur der Teststufe folgendermaßen aussehen: Die Umgebung wird simuliert. Die Interaktion des Steuergerätes mit anderen Steuergeräten im Gesamtsystem steht im Fokus. Die Anforderungen an das Testen werden aus den Anforderungen des Gesamtsystems abgeleitet. Der Fokus liegt auf der Systemfunktionalität. Wie die einzelnen Module der Software auf dem Steuergerät funktionieren oder miteinander kommunizieren, steht nicht im Fokus dieser Teststufe.

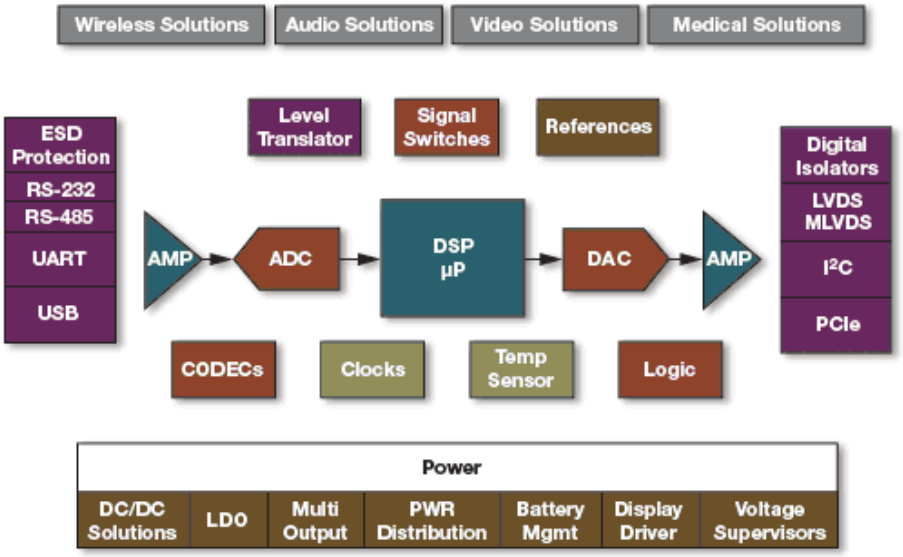

Abbildung 3: Architektur der Softwaremodule[40]

Das Steuergerät wird in einen Prüfstand integriert, so dass die Kommunikation über ein Gesamtsystemkonstrukt mit anderen Steuergeräten zu Stande kommt. Der Begriff HiL (Hardware in The Loop)[41] hat sich in diesem Kontext durchgesetzt. Eine andere Möglichkeit, im Automobilkontext den Systemtest durchzuführen, ist, das Steuergerät in das Auto einzubauen und Testfahrten durchzuführen. Im Laufe der Testfahrten werden Daten gesammelt, die mit Referenzdaten nach der Testfahrt verglichen werden. Somit kann die Funktionalität auf der Systemebene überprüft werden.

Die Softwaretest-Ebene auf der anderen Seite betrachtet die gesamte Software auf einem Steuergerät. Die Kommunikation mit anderen Steuergeräten und die Systemfunktionalität stehen am Rande. Man spricht im Automobilkontext von Restbussimulation. Das bedeutet, dass die restlichen Komponenten des Gesamtsystems simuliert werden, um ausgewählte Funktionalitäten in dieser Ebene testen zu können. Ob mit einer Restbussimulation oder ohne getestet wird, legt man im Testplan fest, da diese Funktionalitäten meist hohe Prioritäten

[40] Vgl. www.ti.com, Datenblätter eines beliebigen Controllers, angesehen am 1.09.2012
[41] Hardware in the Loop ist eine Methode zum Testen und Absichern von eingebetteten Systemen, zur Unterstützung während der Entwicklung sowie zur vorzeitigen Inbetriebnahme von Maschinen und Anlagen.

haben. Die Infrastruktur dieser Ebene besteht aus dem Steuergerät und einem Debugger.[42]

Die Softwareintegrationsebene betrachtet die Kommunikation der einzelnen Module der Software auf dem Steuergerät. Die Infrastruktur dieser Ebene besteht aus einem Prüfstand (meist Hardware in The Loop). Der Fokus liegt auf die Überprüfung der richtigen Integration und des Zusammenspiels der einzelnen Softwaremodule auf einem Steuergerät. Das Steuergerät wird in sich als Komponente allein betrachtet.

Die letzte Ebene, der Softwaremodul-Test, betrachtet die einzelnen Module der Software als einzelne Komponenten. Die Softwaremodule werden entweder mit dem Entwicklungswerkzeug oder mit einem eigenen Werkzeug als einzelne Komponenten getestet. Der Fokus liegt auf dieser Ebene auf dem Modul. Das Modul versteht sich als gekapselte fertige Komponente, die auf die richtige Funktionalität überprüft werden soll. Soll aber das Modul mit anderen Komponenten interagieren und kann die Funktionalität erst getestet werden, wenn andere Komponenten fertig sind, wird folgendermaßen vorgegangen: Entweder wartet man auf die Entwicklung der anderen Komponente, wenn dasselbe Entwicklungsteam die Komponente implementieren soll, oder man simuliert die andere Komponente durch ein so genanntes Stub[43] oder Mocks.[44] Sie liefern nur die Funktionalität, die notwendig ist, um mit der zu testenden Komponente kommunizieren zu können. Der Testfokus liegt immer noch auf dem zu testenden Software-Modul.

---

[42] Ein Debugger (von engl. bug im Sinne von Programmfehler) gehört zu den Werkzeugen zum Diagnostizieren und Auffinden von Fehlern in Steuergeräten, vor allem in der Software, aber auch in der für die Ausführung benötigten Hardware.

[43] Stub (von englisch stub, Stubben, Stummel, Stumpf oder Stutzen) ist eine Methode in der Entwicklung von Software und bezeichnet einen Programmcode, der anstelle eines anderen Programmcodes steht. Der Programmcode, den der Stub ersetzt, ist noch nicht entwickelt oder liegt auf einem anderen Rechner oder in einem anderen Speicherbereich.

[44] Mock-Objekt ist eine gängige Bezeichnung in der Softwareentwicklung und wird in der testgetriebenen Softwareentwicklung „Dummy“-Objekt oder Attrappe genannt. Sie dienen als Platzhalter für echte Objekte, die innerhalb von Modultests verwendet werden.

Nachdem die einzelnen Stufen näher betrachtet worden sind, werden im Folgenden die Werkzeuge in der Automobilindustrie vorgestellt, die eingesetzt werden, um einzelne Testaufgaben auf jeder Stufe realisieren zu können. Die Auswahl der Werkzeuge wurde aufgrund folgender Kriterien durchgeführt:

- eigene Erfahrung des Autors mit dem Werkzeug,
- Marktanteile in der Automobilindustrie,
- Bekanntheit und Nutzungsvorteile,
- Dokumentation und Support der Werkzeuge.

Auf der Systemtestebene wird der HiL-Prüfstand (Hardware in the Loop) herangezogen. Der Systemtest im Fahrzeug wird nicht näher betrachtet. Volkswagen nutzt die EXAM-Technologie[45], die eine umfassende Methodik für Systemtest darstellt. Sie basiert auf der Unified Modeling Language (UML)[46], mit der Testfälle dargestellt, durchgeführt und ausgewertet werden können. Die Testabläufe können ohne Programmierkenntnisse grafisch in Sequenzdiagrammen modelliert werden. EXAM stellt eine einheitliche Sprache zur Darstellung von Testsachverhalten zur Verfügung und eignet sich für den Einsatz in der Hardware-in-the-Loop-Simulation (HiL), der Prüfstands-Automation und der Industrie-Automation. EXAM war eine gemeinsame Entwicklung der AUDI AG, der Volkswagen AG und der MicroNova und hat sich seitdem als konzerneinheitliche Testautomatisierung für HiL-Simulatoren im Volkswagen-Konzern etabliert. Die Anwender von EXAM wurden von Anfang an sehr stark einbezogen, so dass nun ein ausgereiftes System mit Tooling und Bibliotheken zur Verfügung steht. EXAM bietet eine Aufteilung von Aufgaben in unterschiedliche Rollen im Testprozess, z. B. Testdesigner, Prüfstandsbetreuer und Testergebnis-Reviewer. Das EXAM-Rollenmodell stellt eine einfache Integration in bestehenden Testmanagement-Prozessen dar. Somit unterstützt EXAM den kompletten Testprozess von der Testspezifikation bis zur Testergebnisverwaltung.

---

[45] Vgl. MicroNova unter http://www.exam-ta.de/ angesehen am 13.07.12.
[46] Die Unified Modeling Language (Vereinheitlichte Modellierungssprache), gehört zu den graphischen Modellierungssprachen zur Spezifikation, Konstruktion und Dokumentation von Software-Teilen und anderen Systemen.

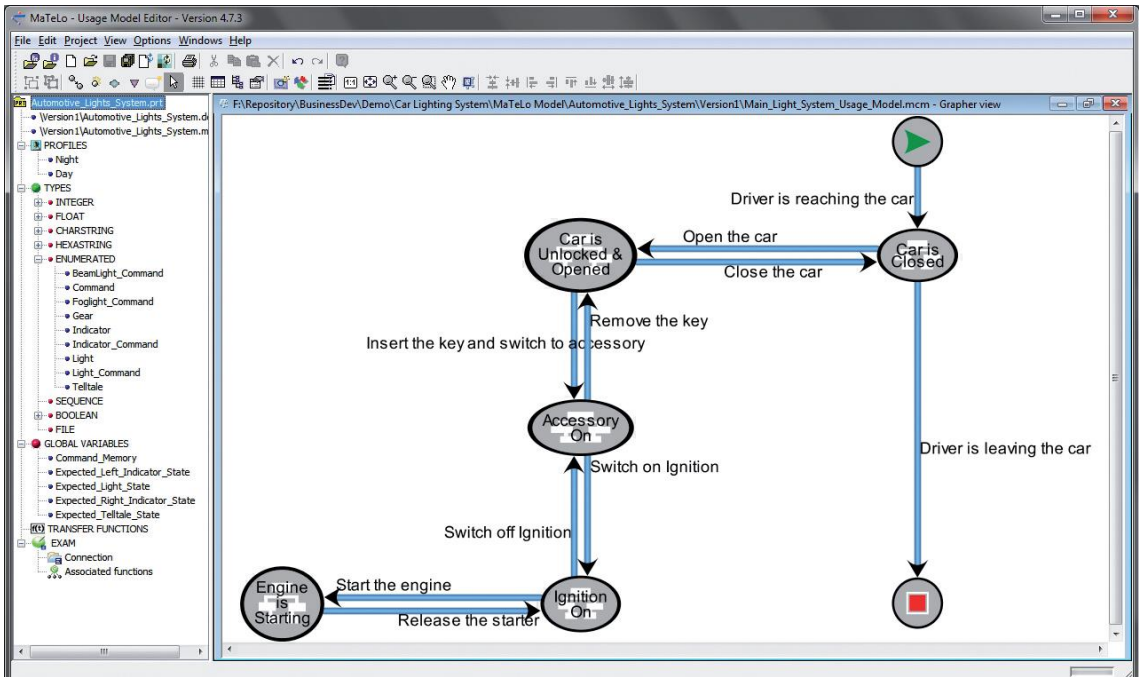

Abbildung 4: EXAM-Technologie[47]

Auf der Softwaretest-Ebene spielen andere Werkzeuge eine Rolle. Im vorliegenden Fall geht es um das Steuergerät als Black-Box. Das Steuergerät und die enthaltene Software stellen das DUT (Device Under Test) dar. In der Automobilindustrie haben sich Anbieter von Debuggern auf Mikrocontroller-Familien durchgesetzt. Ein Beispiel für solch einen Debugger ist winIDEA[48].

[47] Vgl. MicroNova unter http://www.exam-ta.de/, zuletzt angesehen am 13.07.12.
[48] Vgl. iSystem unter http://www.isystem.com/products/winidea, zuletzt angesehen am 13.07.12.

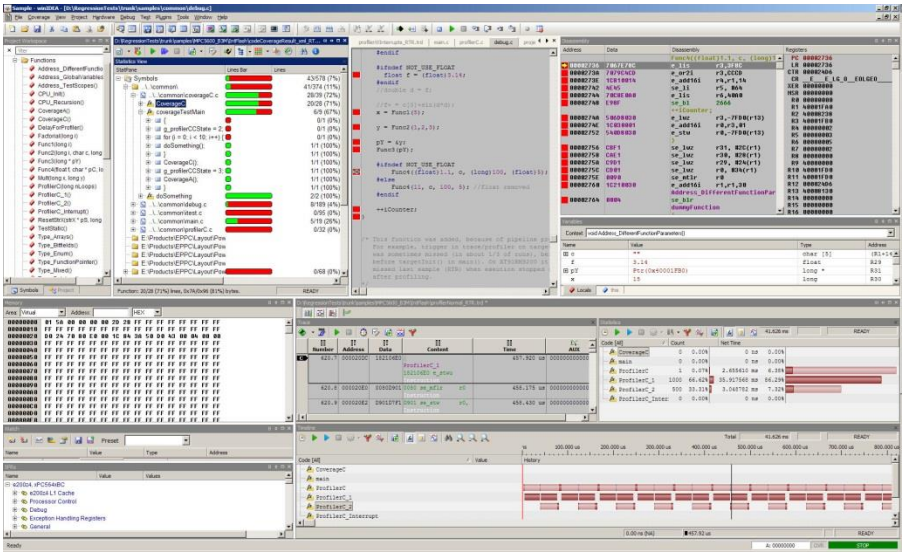

Abbildung 5: Werkzeug für die Softwaretestebene mit einem Debugger[49]

Der Debugger ist ein Werkzeug der Softwareentwicklungsphase. Auf dieser Testebene wird er eingesetzt, um die Software auf Korrektheit zu überprüfen. Der Debugger bietet mehrere Möglichkeiten, auf die softwareinterne Struktur zuzugreifen. Auf die Funktionalität des Debuggers und daraus erwachsende Möglichkeiten wird hier nicht eingegangen. Die Testautomatisierung ist im Debugger ein fester Bestandteil, da ohne sie der Aufwand für den Test sehr groß wäre.

Die nächste Ebene in der Betrachtung ist die Softwareintegrationsebene. Meistens kommt auf dieser Ebene ein Mini-HiL-Prüfstand zum Einsatz. Anbieter und Vorreiter in diesem Zusammenhang ist die Firma Vector GmbH[50] mit ihren Werkzeugen. Ein oft benutztes Werkzeug der Firma ist CANoe, das heute unter der Version 8.0 vertrieben wird, mit HiL-Simulatoren als modularem Test-Hardware-VT-System.

Diese Werkzeuge der Vector GmbH haben ihren Vorteil in der einfachen Implementierung der Automatisierung der Testaufgaben. Die Schnittstellen, die Bedienung und die mächtigen Hardwarekomponenten und Module stellen eine Bereicherung der Testaufgaben in der Automobilindustrie dar. Die Erfahrung des Autors bestätigt diese These. Dennoch sind die Komponenten der Vector

[49] Vgl. iSystem unter http://www.isystem.com/products/winidea, zuletzt angesehen am 13.07.12.
[50] Vgl. Vector unter www.vector.com, zuletzt angesehen am 13.07.2012.

GmbH sehr teuer, aber der Support der Firma ist sehr flexibel und im deutschen Raum einfach und schnell zu erreichen.

Die letzte Stufe der Abhandlung ist die Modultestebene. Auf dieser Ebene existieren kommerzielle Lösungen und Open Source Software. Beide stellen mächtige Varianten zum Test in der Modultestebene. Die gebräuchliche kommerzielle Lösung im Modultest ist die Software Tessy.[51]

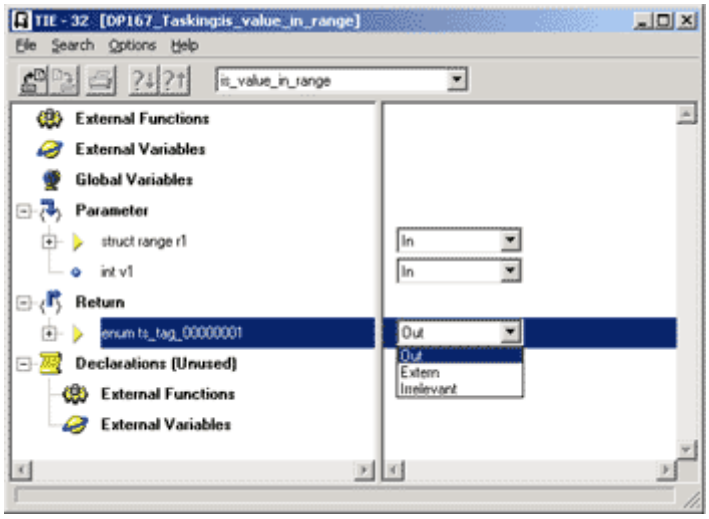

Abbildung 6: Werkzeug für die Modultestebene

Tessy gehört zu den Werkzeugen, die den automatisierten Modultest durchführen und ist geeignet, die Software auf eingebetteten Systemen, die in den Programmiersprachen C oder C++ geschrieben sind, zu testen. Tessy wurde 1990 im Software-Forschungslabor der Daimler AG in Berlin entwickelt.

Das Pendant zu Tessy in der Open Source Welt heißt xUnit Frameworks. Es ist die Bezeichnung verschiedener Frameworks für automatisierte Softwaretests. Diese Frameworks erlauben das Überprüfen verschiedener Elemente (units) von Software, wie etwa Funktionen und Klassen.

Auf der Modultestebene erlauben die vorgestellten Werkzeuge die Automatisierung. Jedes Werkzeug beinhaltet bestimmte Anweisungen, um die Automatisierung der Testaufgaben durchzuführen.

---

[51] Hitex unter http://www.hitex.com/index.php?id=module-unit-test. zuletzt angesehen am 13.07.2012.

Die Entscheidung zwischen der Open Source-Lösungen und der kommerziellen Lösungen kann anhand folgender Kriterien zu Stande kommen:

- Zertifizierung der Testwerkzeuge in der Branche,
- Aufwand, Kosten und vorhandenes Know-how beim Entwicklungsteam,
- Support und Dokumentationen der Software,
- Schulungsangebote für das Werkzeug.

Die Testautomatisierung ist im Software-Testplan verankert. Es ist Aufgabe des Testmanagers, die Strategie im Laufe des Testprozesses festzulegen. Der Ablauf der Testautomatisierung wird folgendermaßen durchgeführt: Der Testmanager legt die Testaufgaben fest. Die Testspezifikationen leiten sich aus den Anforderungen an die eingebettete Software ab. Alle vorgestellten Werkzeuge bieten die Möglichkeit, Testaufgaben automatisiert durchzuführen. Automatisiert bedeutet, dass die Testspezifikationen (Testfälle) in einer beliebigen Anzahl automatisch mit einfacher Bedienung durchgeführt werden können.

Man unterscheidet zwischen vollautomatisierten und teilautomatisierten Tests. Der Unterschied macht sich an der Bedeutung eines Testfalls fest. Testfälle sind Ergebnisse der Spezifikationsphase. Sie werden entweder manuell oder automatisch aus den Anforderungen hergeleitet. Der Fokus in dieser Arbeit richtet sich auf die manuelle Herleitung. Automatische Testfallgenerierung funktioniert nur, wenn die Anforderungen formal in einem Modell vorliegen. Ein Testfall spezifiziert die Soll- und die Ist-Werte. Das Ergebnis ist der Vergleich dieser Werte. Er legt die voreingestellten Eingabewerte für das Programm des Steuergerätes fest. Normalerweise steuert der Testrechner die Umgebung, so dass über die Schnittstellen des Steuergerätes kommuniziert werden kann. Mit dem Testwerkzeug werden über die Schnittstellen des Steuergerätes mehrere Testfälle gleichzeitig durchgeführt. Die vorgestellten Werkzeuge bieten diese Möglichkeit. Wird der Ist-Soll-Vergleich für alle Testfälle nacheinander automa-

tisiert durchgeführt, spricht man von Teil-Automatisierung. Die Ergebnisse liegen als Report in HTML-[52] oder XML[53]-Format vor.

Um die Definition der Voll-Automatisierung verstehen zu können, benötigt man organisatorische Hintergründe zur Testdurchführung. Im idealen Zustand werden die Testfälle am Beginn der Entwicklung parallel zur Bestimmung der Anforderungen und Pflichtenhefte spezifiziert. Aufgrund der Notwendigkeit der Rückverfolgbarkeit werden sogar die Testfälle mit den Anforderungen verlinkt. Genau genommen sind die Testfälle am gleichen Ort aufbewahrt wie die Anforderungen. Dazu dienen Werkzeuge des Anforderungsmanagements. Die Verlinkung der Anforderungen mit den Testfällen hat das Ziel, nachzuweisen, dass die Anforderung getestet worden sind. In der Automobilindustrie bewerten Assessoren die Entwicklungsprozesse und legen auf die Rückverfolgbarkeit sehr viel Wert.

Die Voll-Automatisierung ist erreicht, wenn die Testfälle aus dem bewahrten Ort (Anforderungsmanagement-Werkzeug) maschinell in das Testwerkzeug exportiert werden, so dass die Testfälle nicht implementiert werden müssen. Man implementiert mit dem Testwerkzeug einen Rahmen, um die Testfälle nach dem Export verarbeiten zu können. Auf die gleiche Weise exportiert man die Testergebnisse aus dem Testwerkzeug, um diese wiederum in das Anforderungsmanagementwerkzeug zu importieren. Die Testergebnisse müssen aus Sicht der Rückverfolgbarkeit mit den Testfällen verlinkt werden.

---

[52] Die Hypertext Markup Language (Hypertext-Auszeichnungssprache) gehört zu den textbasierten Auszeichnungssprachen zur Strukturierung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten. HTML-Dokumente bilden die Grundlage des World Wide Web und werden von einem Webbrowser dargestellt.

[53] Die Extensible Markup Language (Erweiterbare Auszeichnungssprache) gehört zu den Auszeichnungssprachen zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien.

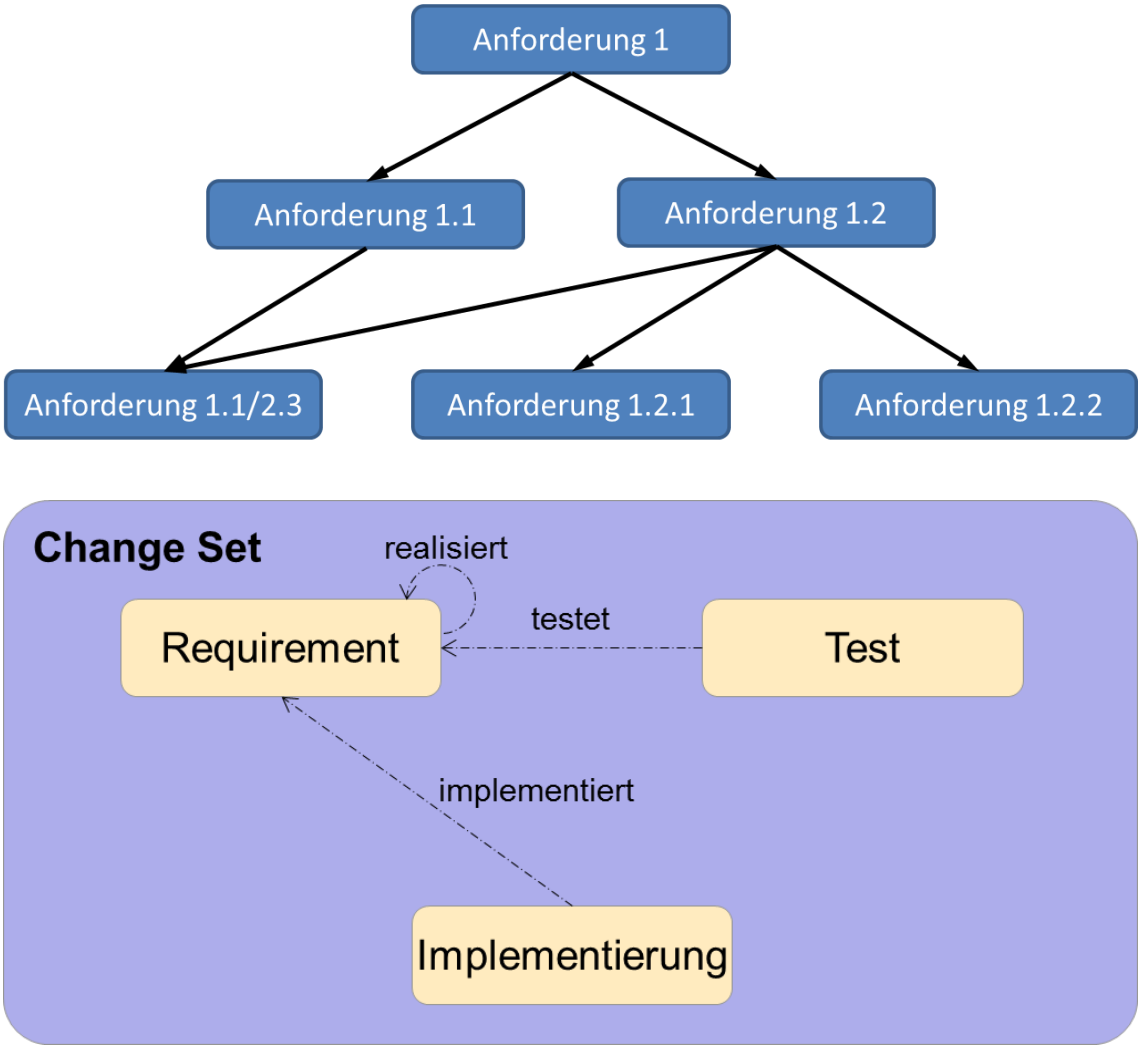

Abbildung 7: Anforderungen und Testfälle[54]

Testautomatisierung in all ihren Facetten gehört zu den zentralen aktuellen und zukünftigen Themen in der Entwicklung von Software. In den letzten Jahren hat die Testautomatisierung massiv an Bedeutung gewonnen. Der Markt bietet immer mehr Werkzeuge, die für unterschiedliche Methoden und Einsatzgebiete geeignet sind. Diese Werkzeuge unterstützen mehr Technologien und Verfahren.[55]

[54] Vgl. HOOD, Expert in Requirements ,http://blog.hood-group.com/blog/2012/05/29/was-braucht-man-zum-gedankenlesen/, zuletzt angesehen am 1.09.2012
[55] Vgl. Seidl (2012), S. 165.

Zusammenfassend ist die Testautomatisierung eine Strategie, um folgende Vorteile zu erhalten:

- schnelle Abarbeitung der Testfälle,
- gut geeignet für Regressionstests,
- Wiederverwendbarkeit der Testfälle,
- Zeit, Kosten und manuelle Arbeit sparen,
- bessere Platzierung in Reifegradmodellen.

Für den Einsatz der vorgestellten Werkzeuge für eine bestimmte Domäne sollten folgende Bedingungen in Betracht gezogen werden, die u. U. Problemfelder für die Testautomatisierung darstellen:[56]

- Technische Schnittstellen: Welche Schnittstellen hat das Testobjekt (Embedded System)? Können diese Schnittstellen vom Testwerkzeug angesprochen werden, um Daten bidirektional zu verarbeiten?
- Organisatorische Schnittstellen: Inwieweit kann das Werkzeug sich reibungslos in den aktuellen Testprozess des Unternehmens integrieren? Können teil- oder vollautomatisierte Tests mit dem Werkzeug und dem aktuellen Prozess durchgeführt werden?
- Ökonomische Betrachtung der Investitionen an das Werkzeug und die notwendige Einarbeitung des Personals.
- Die Innovation des Werkzeugs muss betrachtet werden: Inwieweit kann das Testteam das Werkzeug weiter anpassen und weiterentwickeln? Stehen die Ressourcen und die Schnittstellen dafür bereit?
- Die soziale Ebene im Testteam muss betrachtet werden: Inwieweit werden die Tester durch die Arbeit mit dem Werkzeug motiviert?

[56] Vgl. Seidl (2012),S. 166.

4 Wirtschaftliche Prüfung der Testautomatisierung

Im Rahmen der wirtschaftlichen Prüfung wird das magische Dreieck herangezogen[57]. In der Entwicklung spielen die drei Faktoren Zeit, Kosten und Qualität eine Rolle. In diesem Kapitel wird der Kostenfaktor näher betrachtet.

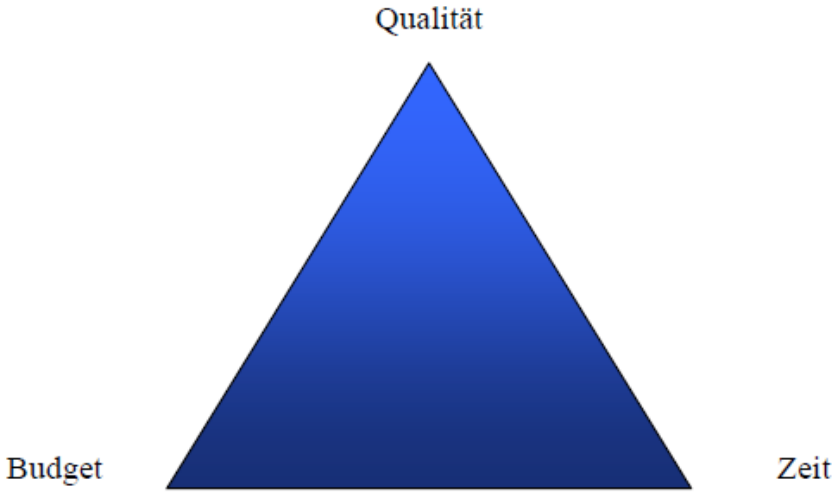

Abbildung 8: Magisches Dreieck: Qualität, Zeit und Budget

Qualität und Zeit hängen zum einem vom ausgewählten Werkzeug und zum anderen von den Projektbedingungen ab. Sie können sehr gut mit einer Evaluierung des Werkzeugs zum automatisierten Testen abgeschätzt werden. Im 5. Kapitel dieser Arbeit werden Argumente zur Auswahl und zur Evaluierung eines Werkzeugs wiedergegeben.

Bei der wirtschaftlichen Prüfung eines Testautomatisierungs-Rahmenwerks muss der Nutzen gegenüber dem Risiko abgewogen werden. Zur Untersuchung der wirtschaftlichen Prüfung wird die Diplomarbeit von Arne Drescher[58] und der Artikel von Software Quality Lab[59] herangezogen.

Kosten-Nutzen-Überlegungen sind immens wichtig, wenn das Management eine Entscheidung für eine Testautomatisierung treffen möchte. Die Analyse

[57] Vgl. Drescher (2010), S.39.

[58] Vgl. Drescher (2010), S.40.
[59] Vgl. Quality Lab (2009).

muss im Rahmen der Evaluierungsphase eines Werkzeugs und am besten in einem Pilotprojekt durchgeführt werden. Meistens scheitert eine Testautomatisierung daran, dass das Management keinen effektiven Nutzen darin sieht. „Es reicht nicht mehr aus, die Lizenzkosten und jährliche Wartungskosten bei der Auswahl einer Testautomatisierungssoftware zu betrachten. Weiters müssen auch neu entstehende Personalaufwände und die Schulung dieses Personals betrachtet werden. Die Kosten müssen den Aufwänden für die manuelle Testdurchführung gegenübergestellt werden. Testautomatisierung ist nur dann wirtschaftlich, wenn der Nutzen der Testautomatisierung im Vergleich zu den entstehenden Kosten überwiegt.“[60]

Einführungskosten, Anschaffungskosten der Testautomatisierungssoftware, die Personalkosten für die Testspezifikation und für die Programmierung von initial benötigten Steuerelementen und / oder Testfällen spielen eine wesentliche Rolle. Schulungskosten und Infrastrukturkosten sind die Hauptkostenfaktoren, die in Betracht gezogen werden müssen. Die Erhaltungskosten setzen sich aus den Update-, Wartungs- und Supportkosten zusammen. Dazu spielen die Personalkosten für die Wartung bestehender Testfälle und für die Programmierung von Unterstützungswerkzeugen oder Testfällen eine Rolle.

Um diese Kostenfaktoren zu veranschaulichen, wird ein Beispiel herangezogen. Für zwei Testautomatisierungstools wird der Personalaufwand berechnet, wobei für die Aufwandsabschätzung folgendes Szenario verwendet wird:

- 1000 Testfälle, wobei jeder 5.Testfall gewartet wird,
- die Aufwände für die Testfallerstellung und Testfallwartung werden getrennt betrachtet,
- die Auswertung wird normalisiert, um die Vergleichbarkeit der beiden Tools zu gewährleisten.

Laut Quality Lab wird für dieses Beispiel folgendes Personal eingesetzt:

- Fachtester (für die Erstellung und Wartung automatisierter Testfälle),

[60] Vgl. Quality Lab, (2009).

- Toolspezialist (hat gute Toolkenntnisse, übernimmt die Aufgabe der Korrektur fehlgeschlagener Testfälle),
- Programmierer (programmiert die Unterstützungswerkzeuge zur Testautomatisierung),
- Testspezifizierer (führt die Spezifikation der Testfälle für automatisierte Tests durch),
- Testkoordinator (übernimmt Testmanagementaktivitäten).

„Bei Tool 1 ist die Einbindung des Fachpersonals nur bedingt möglich - es werden mehr Programmierer als technische Toolspezialisten benötigt.“[61] „Fachpersonal kann bei Tool 2 stärker in die Testfall-Erstellung eingebunden werden - es werden mehr fachliche Toolspezialisten, aber weniger Programmierer benötigt.“[62]

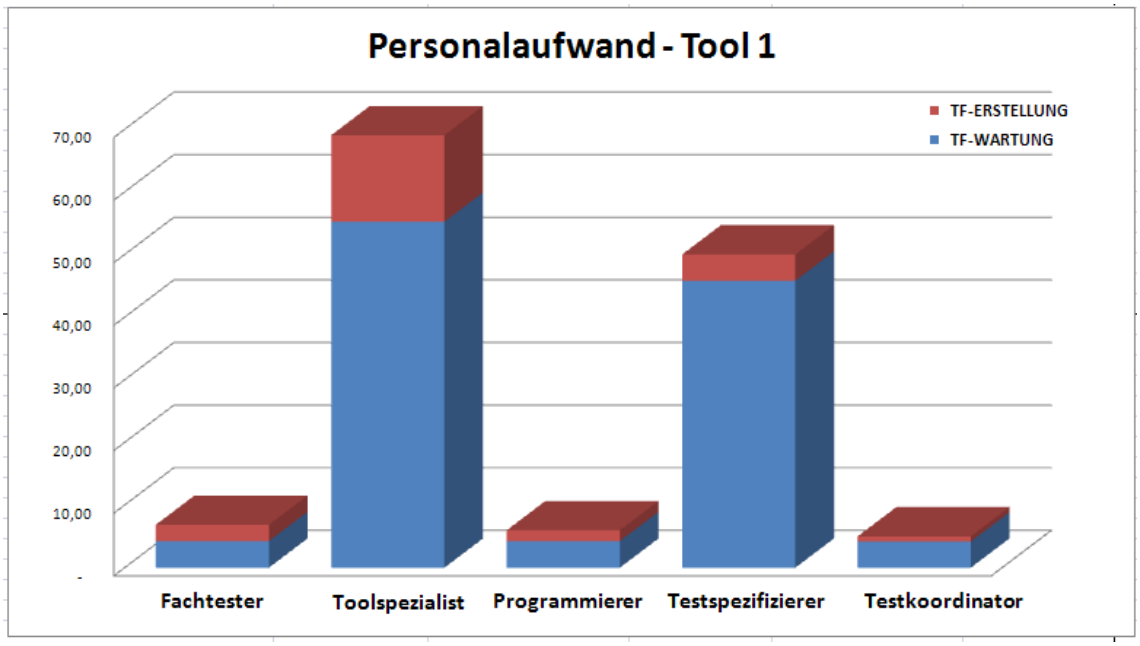

Abbildung 9: Personalaufwand Tool 1[63]

[61] Vgl. Quality Lab, (2009).
[62] Vgl. Quality Lab, (2009).
[63] Vgl. Quality Lab, (2009).

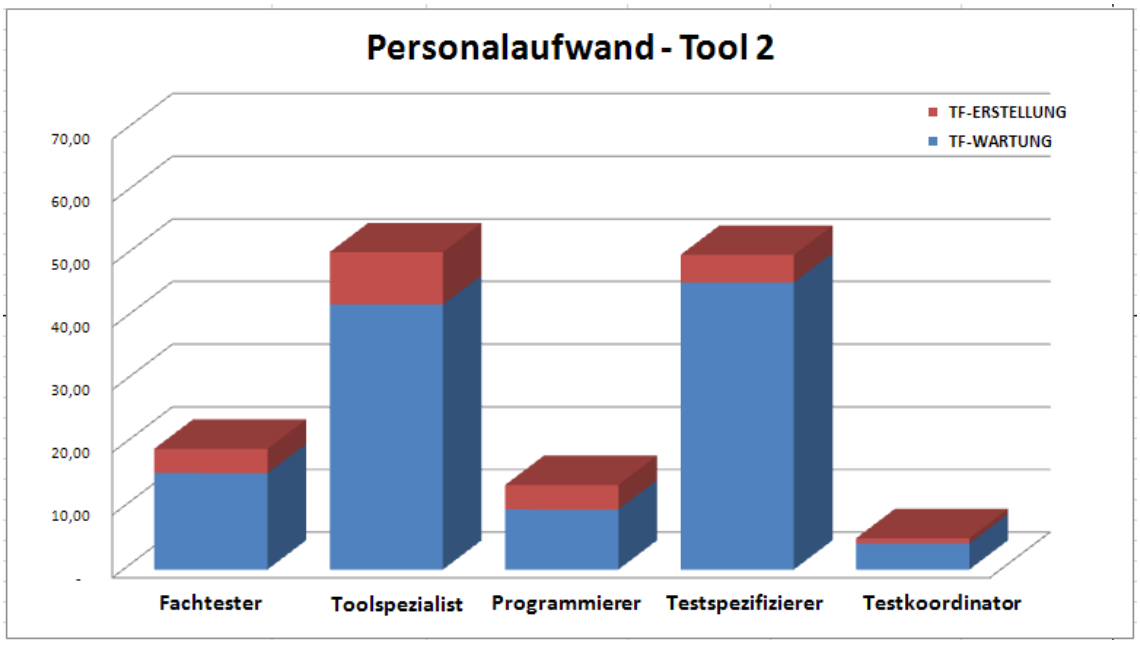

Abbildung 10: Personalaufwand Tool 2[64]

Das Beispiel soll zeigen, dass auf die verschiedenen Rollen im Testteam verschiedene Aufwände verteilt werden können und dass die Evaluierungsphase Schlussfolgerungen liefern kann, in welchem Maße der Aufwand den einzelnen Testrollen zugerechnet werden kann. Aus diesem Beispiel lässt sich ableiten, dass Testen mit Personalaufwand und wirtschaftlichen Kosten verbunden ist. Diese Kosten können mit der Testautomatisierung verringert werden.

Dazu kann man Tests koordiniert ablaufen lassen. Die Testspezifikation und Wartung der Testfälle spielt eine entscheidende Kostenrolle. Die Testautomatisierung benutzt die Testfälle, um das DUT, also das Testobjekt, zu überprüfen. In dieser Arbeit ist das Steuergerät das DUT. Die Testfälle werden im Laufe der Entwicklung immer wieder angepasst und verändert, da sich die Anforderungen an das Testobjekt immer verändern. Die Testautomatisierung spielt eine immer wichtigere Rolle, wenn bei jedem Software-Release die Tests des Vorgänger-Release automatisiert wiederholt werden (Regressionstest). Nicht alle Testfälle, die man im Laufe des Projekts spezifiziert, können für die Testautomatisierung herangezogen werden. Immer wieder müssen Testfälle der Software manuell

[64] Vgl. Quality Lab, (2009).

oder in Testgruppen allein ausgeführt werden. Wichtig ist, die Schwelle zu definieren, an der ein Testfall zur Automatisierung herangezogen wird oder nicht.

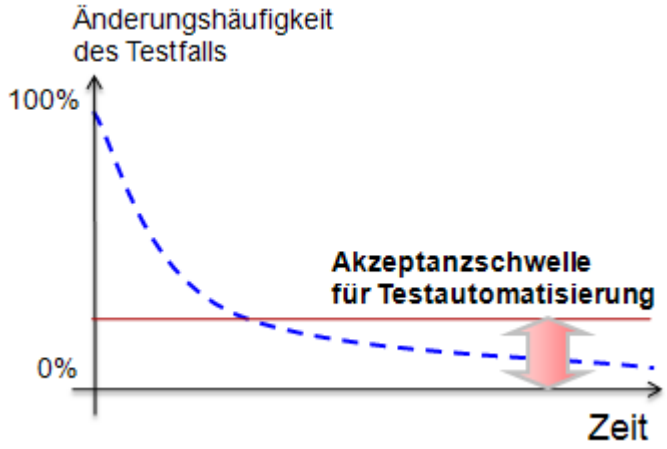

Abbildung 11: Akzeptanzschwelle der Testautomatisierung[65]

Diese Schwelle ist durch drei Faktoren definiert:

- Stabilität des Testobjekts,
- Häufigkeit der Testdurchführung,
- Kritikalität des Testobjekts und des Testfalls.

Das Entscheidende für die wirtschaftliche Beurteilung der Testautomatisierung ist der Return of Investment (ROI). Testautomatisierung und manuelle Testdurchführung werden dazu gegenübergestellt. Kann mit der Testautomatisierung ein ROI erreicht werden? Die einfachste Beurteilung, ob sich eine Testautomatisierung lohnt, ist der Vergleich von Aufwänden der manuellen mit der automatisierten Testdurchführung. Entscheidend bei der Beurteilung der Aufwände ist, wie oft die Testdurchführung wiederholt wird. Ein Regressionstestfall wird bei jedem neuen Status der Softwareentwicklung (Release) durchgeführt. Bei häufigen Regressionstests wird der ROI der Testautomatisierung schneller erzielt. Für diesen Vergleich werden die Aufwände für die Testspezifikation manueller und automatisierter Testfälle eingerechnet, wobei der Aufwand für die Testspezifikation beim automatisierten Testfall höher angenommen wird als beim manuellen Testfall.

[65] Vgl. Quality Lab, (2009).

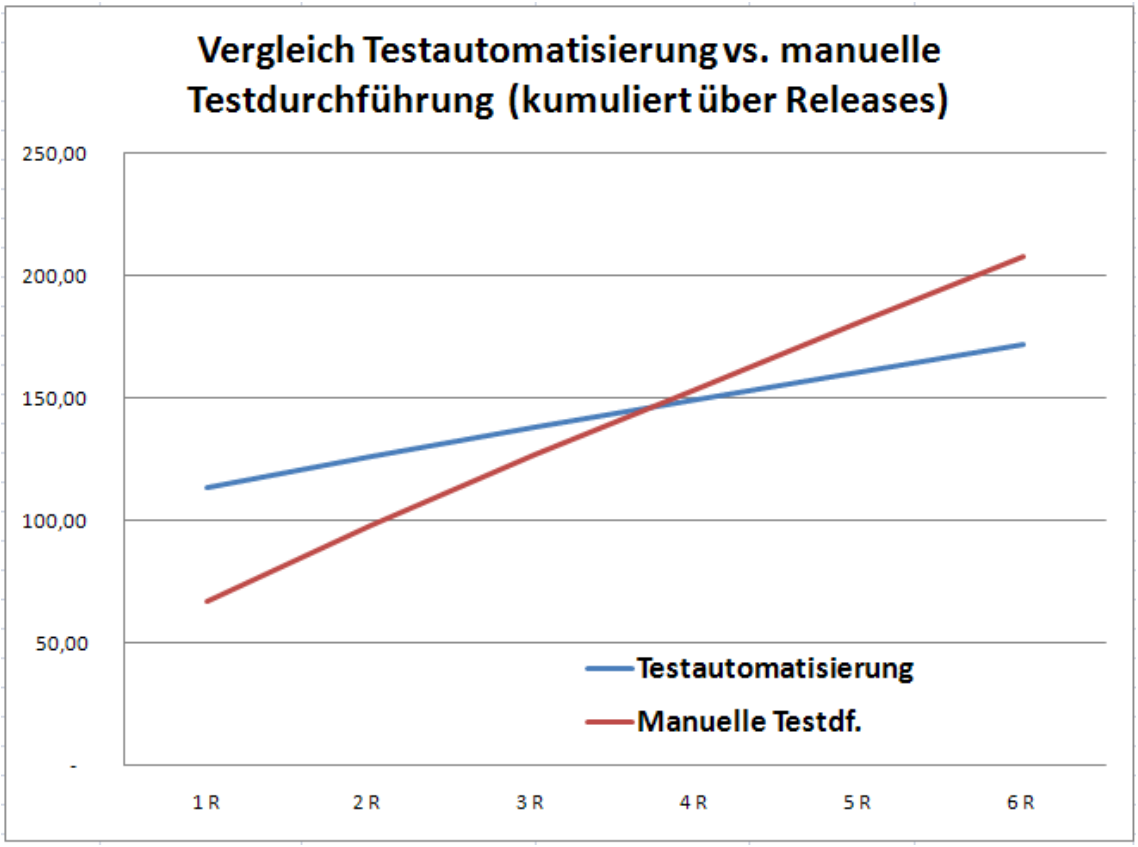

Abbildung 12: Vergleich Testautomatisierung vs. manuelle Testdurchführung[66]

Abbildung 12 zeigt, dass der Aufwand am Anfang der Spezifikation der automatisierten Testfälle viel höher ist als die der manuellen. Im Laufe des Projekts verringert sich dieser Aufwand linear gegenüber den manuellen Testfällen. Folgende Vorteile durch Testautomatisierung sind laut Quality Lab meist gegeben:[67]

Reduktion der Kosten und des Aufwands

- Die Kosten für die Testerstellung und die Spezifikation werden sich nach wenigen Wiederholungen der Tests (Regressionstests) amortisieren.
- Die Abschätzung für den Aufwand lässt sich einfach ermitteln.
- Für den wiederholten Test benötigt man wenig bis kein Personal, die Testautomatisierung reduziert Personalaufwand.

[66] Vgl. Quality Lab, (2009).
[67] Vgl. Quality Lab, (2009).

Entschärfung der kritischen Pfade

- Mit der Automatisierung der Testfälle kann man eine hohe Anzahl an Testfällen (bestimmte Testgruppen) nur in kurzer Zeit durchführen. In kritischen Projektphasen ist dieser Zeitvorteil von besonderer Bedeutung.
- Bei der Automatisierung der Testfälle kann man erreichen, dass die Tests nachts oder an Wochenenden durchgeführt werden, so dass eine Reduktion von Ressourcenengpässen erreicht werden kann.
- Expertenwissen ist nur dann notwendig, wenn es darum geht, die Tests vorzubereiten, Fehler im Testfall zu beheben und bei der Testauswertung eine Analyse abzugeben.

Sichere Dokumentation und Erreichung von Reproduzierbarkeit

- Die Testfälle werden mit vielen Überlegungen spezifiziert. Somit ist die Reproduzierbarkeit der Testdurchführung unter identischen Ausgangsbedingungen möglich.
- Die Dokumentation wird mittels Teil- oder Vollautomatisierung automatisch protokolliert. Das ist eine Eigenschaft, die in jedem Automatisierungswerkzeug vorhanden ist.

Test- und Produktqualität wird verbessert und erhöht

- Die Aufgabe, einen Testfall zur Automatisierung zu spezifizieren, benötigt Kreativität und Transparenz. Das erhöht die Qualität in der Testdurchführung und ermöglicht die Fehlervermeidung.
- Ermöglicht neue innovative Entwicklungsmethoden.

Dieser Kapitel hat gezeigt, dass sich aus der Automatisierung der Qualitätssicherung wirtschaftliche Vorteile ergeben. Zusammenfassend lässt sich feststellen, dass sich die Automatisierung im Laufe des Projekts amortisiert, da die Testdurchführung wiederholt werden kann, wenn die Anforderungen an die Software im Laufe des Projekts und im Rahmen des Zuwachses der Softwaremodule verändert werden.

In den sicherheitskritischen Anwendungen kann die Automatisierung viel Zeit und Kosten sparen, wenn man die Qualitätssicherung auf kritischen Komponenten im Laufe der Entwicklung wiederholt.

5 Vergleichende Analyse zur Medizintechnik

Das Ziel dieser Arbeit ist unter anderem herauszufinden, inwieweit die Konzepte und Methoden der Testautomatisierung aus der Automobilindustrie in die Domäne der Medizintechnik übertragen werden kann. Um diese Frage beantworten zu können, wird zunächst die Domäne der Medizintechnik näher betrachtet.

Im Rahmen dieses Kapitels wird die Domäne der Medizintechnik kurz beschrieben. Die Rahmenbedingungen für eine Übertragung der Testautomatisierungskonzepte auf Medizinprodukte werden veranschaulicht und diskutiert. Das Ergebnis wird pragmatisch dargelegt.

5.1 Die Medizintechnikbranche

„Medizintechnik, auch biomedizinische Technik genannt, ist die Anwendung von ingenieurwissenschaftlichen Prinzipien und Regeln auf das Gebiet der Medizin. Sie kombiniert Kenntnisse aus dem Bereich der Technik, besonders dem Lösen von Problemen und der Entwicklung, mit der medizinischen Sachkenntnis der Ärzte, um die Krankenpflege, Rehabilitation und Lebensqualität gesunder Einzelpersonen zu verbessern. Als verhältnismäßig neue Disziplin besteht viel der Arbeit in der Medizintechnik aus Forschung und Entwicklung (F&E), z. B. in den folgenden Bereichen: Medizinische Informatik, Signalverarbeitung physiologischer Signale, Biomechanik, Biomaterialien und Biotechnologie, Systemanalyse, Erstellung von 3D Modellen etc. Beispiele konkreter Anwendungen sind die Herstellung biokompatibler Prothesen, medizinischer Therapie- und Diagnosegeräte, wie z. B. EKG-Schreiber und Ultraschallgeräte, bildgebender Diagnostik, wie z. B. Magnetresonanztomographie (MRT) und Elektroenzephalografie (EEG) und der Herstellung neuer Medikamente.“[68]

Die Branche der Medizintechnik gilt in Deutschland als besonders innovativ, wachstumsstark und zukunftsträchtig.

---

[68] Vgl. Lexikon von BIONITY, http://www.bionity.com/de/lexikon/Medizintechnik.html, angesehen am 11.09.2012

„Medizinprodukte umfassen eine große Bandbreite von medizintechnischen Produkten und Verfahren, die Leben retten, heilen helfen und die Lebensqualität der Menschen verbessern. Beispiele sind Geräte für Diagnostik, Chirurgie, Intensivmedizin, Implantate, Sterilisation sowie Verbandmittel, Hilfsmittel oder OP-Material. Zu Medizinprodukten gehören nach dem Medizinproduktegesetz (MPG) darüber hinaus auch Labordiagnostika.

Die Welt der Medizintechnologien ist faszinierend. Kardiologische Implantate bringen schwache Herzen wieder in Rhythmus. Die Endoprothetik bringt kranke Gelenke zum schmerzfreien Bewegen. Künstliche Linsen und die refraktive Chirurgie bringen kranke Augen zum Sehen. Moderne Implantate und Geräte bringen taube Ohren zum Hören. Neue MedTech-Verfahren und -Produkte verbessern die Lebensqualität, ja sie retten und erhalten oftmals Leben. Medizinprodukte leisten nicht nur einen wichtigen Beitrag für eine effiziente Gesundheitsversorgung, sie sind auch ein bedeutender Wirtschafts- und Arbeitsmarktfaktor. Die Unternehmen der Medizintechnologie tragen damit zu einer positiven Entwicklung der Gesundheitswirtschaft in Deutschland bei.“[69]

5.2 Definition eines Medizinproduktes

Die Richtlinie 93/42/EWG der europäischen Union erschien am 14. Juni 1993. Sie wird in Deutschland und Österreich kurz als Medizinproduktrichtlinie bezeichnet. Insgesamt gibt es drei Medizinprodukte-EU-Richtlinien, eine davon ist die 93/42/EWG. Der Begriff Medizinprodukt wird in der Richtlinie wie folgt definiert:

„Medizinprodukt: alle einzeln oder miteinander verbunden verwendete Instrumente, Apparate, Vorrichtungen, Software, Stoffe oder andere Gegenstände, einschließlich der vom Hersteller speziell zur Anwendung für diagnostische und/oder therapeutische Zwecke bestimmte und für ein einwandfreies Funktionieren des Medizinprodukts eingesetzte Software, die vom Hersteller zur Anwendung für Menschen für folgende Zwecke bestimmt sind:

---

[69] Vgl. Glossar von BVmed, http://www.bvmed.de/glossar/glossar/medizintechnologien-in-deutschland.html, angesehen am 11.09.2012

- Erkennung, Verhütung, Überwachung, Behandlung oder Linderung von Krankheiten;
- Erkennung, Überwachung, Behandlung, Linderung oder Kompensierung von Verletzungen oder Behinderungen;
- Untersuchung, Ersatz oder Veränderung des anatomischen Aufbaus oder eines physiologischen Vorgangs;
- Empfängnisregelung,

und deren bestimmungsgemäße Hauptwirkung im oder am menschlichen Körper weder durch pharmakologische oder immunologische Mittel noch metabolisch erreicht wird, deren Wirkungsweise aber durch solche Mittel unterstützt werden kann."[70]

Folgende Medizinprodukte werden nicht in der Richtlinie 93/42/EWG betrachtet:[71]

- Aktive implantierbare medizinische Geräte (z. B. Herzschrittmacher) unterliegen als Medizinprodukte der Richtlinie 90/385/EWG.
- In-vitro-Diagnostika unterliegen als Medizinprodukte der Richtlinie 98/79/EG.
- feste nicht wiederverwendbare Einheiten mit einem Arzneimittel (Arzneimittelkit),
- Nationale Ausnahmen einzelner Länder.

Die Europäische Kommission erarbeitet derzeit den Vorschlag einer Neufassung der drei Medizinprodukte-Stammrichtlinien. Eine neue EU-Medizinprodukte-Verordnung (Medical Device Regulation - MDR) soll künftig die beiden Richtlinien 90/385/EWG und 93/42/EWG ersetzen. Erste nichtöffentliche Vorentwürfe liegen den Behörden bereits vor. Der Verordnungsvorschlag im EU-Amtsblatt wird (spätestens) für den 26. September 2012 erwartet.

[70] Vgl. Richtlinie 93/42/EWG des Rates über Medizinprodukte
[71] Vgl. Richtlinie 93/42/EWG des Rates über Medizinprodukte

5.3 Normen und Entwicklungsprozesse in der Medizintechnik

In der Medizintechnik existieren folgende harmonisierte relevante Normen:[72]

- Qualitätsmanagement EN ISO 13485,
- Risikomanagement EN ISO 14971,
- Software Lebenszyklus Prozesse EN 62304,
- Gebrauchstauglichkeit EN 62366 und EN 60601-1-6,
- Normenreihe EN 60601 für medizinische elektrische Geräte.

Die aufgezählten Normen werden herangezogen, um medizinische Software zu entwickeln. Die Normen stellen bestimmte Anforderungen an Softwaretests, sind aber für diese Domäne spezifisch. Die Entwicklung der Produkte nach dem V-Modell kann herangezogen werden, um die Tätigkeiten und Rollen des Softwaretestens darzustellen. In der Praxis können diese abweichen, die Grundannahme bleibt jedoch gleich. Die Stufen des Softwaretestens sind in der Automobilindustrie und in der Medizintechnik ähnlich, wenn die Software nach dem V-Modell entwickelt wird. Für die Übernahme der SPICE-Teststufen in die Medizintechnik müssen gewisse Voraussetzungen erfüllt sein. ENG. 9, 10 nach SPICE ist ein Test auf Systemebene, bei dem die Systemeigenschaften des Medizinproduktes in Betracht gezogen werden müssen. Die Herkunft der vorgestellten Werkzeuge aus der Automobilindustrie bedingt einige Änderungen an der Testumgebung. Es müssen die Systemeigenschaften analysiert werden, um die Testumgebung auf die Eignung für die Aufgabe überprüfen zu können. Die restlichen Stufen (ENG. 6 bis ENG. 8) können übernommen werden, wenn folgende Aspekte übereinstimmen:

- Die Hardware-Plattform wird vom Werkzeug unterstützt.
- Die Programmiersprache und die Schnittstellen werden vom Werkzeug unterstützt.

[72] Vgl. Johner (2011), S. 26 ff.

5.4 Medizinische Informatik

Die Informationssysteme im Gesundheitswesen übernehmen verschiedene Aufgaben. Einige dieser Aufgaben sind:[73]

- Verarbeitungsunterstützung (statistische Auswertung),
- Dokumentationsunterstützung (Archivierung oder Patientenakte),
- Organisationsunterstützung (Termine, Behandlungspfade),
- Kommunikationsunterstützung (Kommunikationsserver),
- Entscheidungsunterstützung (Therapievorschläge).

Zudem zeichnet sich das Gesundheitswesen durch eine hohe Anzahl von verschiedenen Akteuren aus. Zu diesen zählen:[74]

- niedergelassene Ärzte und Psychotherapeuten,
- Apotheken,
- Krankenhäuser,
- Anbieter medizinischer Dienstleistungen,
- Krankenversicherungen,
- staatliche Organisationen,
- Medizingerätehersteller.

Wichtig ist für die Entwicklung von medizinischen Geräten in Bereichen, in denen Embedded Systems das Gehirn der Datenverarbeitung darstellen, dass sich die Entwickler und Tester das Domänenwissen aneignen, das für das Gesundheitswesen spezifisch ist. Dazu zählt das Wissen um gesetzliche und ökonomische Zusammenhänge, das Verständnis der übertragenen Daten und die Möglichkeit, diese zu klassifizieren.

---

[73] Vgl. Johner (2011), S. 189.
[74] Vgl. Johner (2011), S. 187.

5.5 Rahmenbedingungen und Ergebnis der Analyse

Das Marktvolumen für eingebettete Systeme liegt in Deutschland bei über 19 Mrd. Euro. Der Markt entwickelt sich mit stabilen Zuwachsraten. Beispiele für eingebettete Systeme sind in der Medizintechnik Herzschrittmacher. In der Automobilindustrie herrschen andere Rahmenbedingungen für die Entwicklung und Tests von Embedded Software. Folgende Kriterien spielen dort eine wichtige Rolle:

- Gesetzgeber (Normen und Gesetze),
- Kunde,
- Statistik,
- hohe Entwicklungskosten und hohe Stückzahlen,
- Umweltbedingungen,
- Innovation und technologischer Fortschritt (in Bezug auf neue Ideen und Entwicklungswerkzeuge),
- Architektur der Verteilung von Embedded Systems im Fahrzeug,
- Entwicklungsprozesse (V-Modell),
- Bewertungsmodelle (A-SPICE),
- Performance und Speicher der zu entwickelnden Embedded Systems,
- Betriebssysteme der Embedded Systems (Spezifisch für die Automobilindustrie).

Die eingebetteten Systeme im Fahrzeug verarbeiten Sensordaten aus der Umwelt und steuern damit Aktuatoren. Applikationen in der Medizintechnik bestehen meistens aus Desktop-Applikationen, die eine Art Patienten-Monitor darstellen. Die eingebetteten Systeme in der Medizintechnik verarbeiten Sensordaten aus dem Sollwertgeber und den Sensoren. Sie steuern damit Aktuatoren wie in Abbildung 13 veranschaulicht wird.

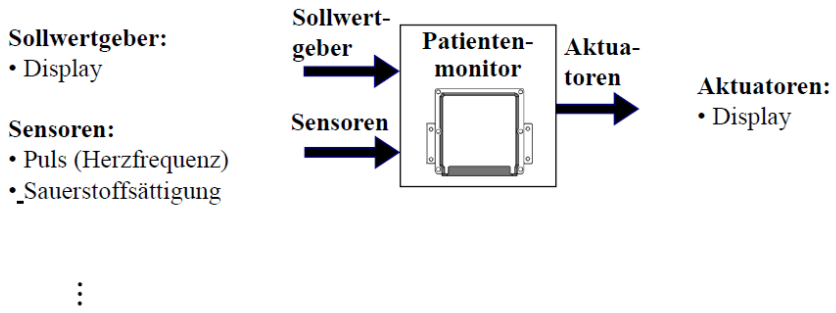

Abbildung 13: Sensordatenverarbeitung im Steuergerät einer medizintechnischen Anwendung

In der Medizintechnik herrschen wiederum andere Rahmenbedingungen für die Entwicklung und den Test von Embedded Software. Folgende Kriterien spielen dort eine wichtige Rolle:

- Gesetzgeber (Normen und Gesetze),
- Patient,
- Innovation und technologischer Fortschritt (in Bezug auf neue Ideen und Entwicklungswerkzeuge),
- Architektur der Verteilung von Embedded Systems im Gerät (Dies ist im Vergleich zur Automobilindustrie auf eine kleine Anzahl von Steuergeräten begrenzt - meist nur Master und Slave.),
- hohe Entwicklungskosten und begrenzte Stückzahlen nach Bedarf,
- Entwicklungsprozesse (V-Modell),
- Bewertungsmodelle (CMMI),
- Performance und Speicher der zu entwickelnden Embedded Systems,
- Betriebssysteme des Embedded Systems (gängige Betriebssysteme wie Windows oder Linux).

Folgende Gemeinsamkeiten der Rahmenbedingung sprechen dafür, die Entwicklung von Embedded Software mit Blick auf die Testautomatisierung von der Automobilindustrie auf die Medizintechnik zu übertragen:

- ähnliche Ziele beim Gesetzgeber,

- das magische Dreieck (Zeit, Kosten und Qualität),
- Entwicklungsprozesse (V-Modell, Teststufen),
- Bewertungsmodelle,
- beide Sektoren gehören zur Gruppe der sicherheitskritischen Anwendungen,
- Notwendigkeit der Softwarequalitätssicherung,
- hohe Anforderungen an Softwarequalität,
- Entwicklungswerkzeuge von Embedded Software,
- Gemeinsamkeiten in Normen wie die funktionale Sicherheit.

Automatische Testprogramme liefern zuverlässige, aber auch oft exakt reproduzierbare Ergebnisse. Mit der Automatisierung entsteht mehr Freiraum für kreatives Testen und komplexere Testtätigkeiten. Die Erwartungen im Management richten sich auf Zeit- und Kostenersparnis bei kürzeren Testzyklen. Bei der Teil-Automatisierung sollte vor allem die Testdurchführung und -auswertung automatisiert werden. Bei der Voll-Automatisierung spielen noch Export und Import von Testfällen und Testergebnissen eine Rolle.[75]
Die Normen, Vorgaben und Standards im Bereich Test sind im Gebiet der sicherheitskritischen Anwendungen ähnlich. Die wirtschaftlichen Vorteile der Testautomatisierung sind eindeutig. Es spricht nichts dagegen, die Automatisierung voranzutreiben, wenn es die Kreativität fördert und die Qualität erhöht.
Um eine Analyse der Gebiete Medizintechnik und Automobilindustrie durchführen zu können, benötigt man Analysekriterien, um entscheiden zu können, ob die Testautomatisierung und deren Werkzeuge geeignet sind. Folgende Aspekte spielen bei den Testautomatisierungswerkzeugen eine entscheidende Rolle:[76]

[75] Vgl. Link/ Hafner (2012).
[76] Vgl. Richard/Manfred/Thomas (2012), S.175.

Werkzeugtyp

Das Entwicklungsteam muss sich fragen, ob das Werkzeug ohne Zusatzaufwand eingesetzt werden kann. Lässt sich das Werkzeug in den Testprozess des Unternehmens einfügen?

Evaluierungshilfen

Demoversionen, Hilfen und eine Dokumentation für das Tool können helfen, eine Entscheidung zu treffen.

Infrastruktur

Welche Systemlandschaft benötigt das Tool (Windows, Linux)? Benötigt das Tool neue Arbeitsplätze für die Evaluierungsphase und für den Echteinsatz?

Usability

Welche Antwortzeiten haben die Tools? Wie ist die Benutzeroberfläche gestaltet? Kann man das Reporting des Werkzeugs direkt nach dem Generieren verwerten? Die Bedienungssprache des Werkzeugs gehört zu den Kategorien, die unbedingt betrachtet werden müssen: Ist das Werkzeug für Multiuser geeignet?

Skalierbarkeit/Stabilität

Kann das Werkzeug um weitere Komponenten erweitert werden? Ist das Werkzeug verlässlich und belastbar?

Dokumentation

Welche Sprache hat die Dokumentation? Gibt es für das Werkzeug Toturials und Webinare?

Schulung

Das Entwicklungsteam muss die Qualität und die Flexibilität der Schulungsangebote für das Werkzeug hinterfragen.

Support/Wartung

Beim Support muss das Team klären, in welcher Sprache und wie der Support und die Wartung des Werkzeugs vorgenommen werden.

Strategische Kriterien

Gehört das Tool zur Standardsoftware oder muss diese für die Testautomatisierung angepasst werden? Sind Tool oder Komponenten als Open-Source oder kommerziell erhältlich?

Kompatibilität mit dem Testobjekt

Sind die verfügbaren Komponenten des Testwerkzeugs für die Kommunikation mit dem Testobjekt geeignet?

Data Interface

Wie können die Daten auf dem Automatisierungswerkzeug verwaltet und gehalten werden? Sind Standard-Verbindungen (Bus-System) mit dem Testobjekt vorhanden?

Programmierung

Mit welcher Programmiersprache wird das Werkzeug bedient? Sind Erfahrungen im Team für diese Sprache vorhanden? Gibt es wiederverwendbare Module? Kann das Werkzeug bestimmte Sprachkonstrukte unterstützen, um das entwickelte Programm lesbar zu machen?

Verwaltung/Integration

Wie können Testfälle für das Werkzeug importiert werden? Wie werden die Testfälle verwaltet? Wie funktioniert der Report von Ergebnissen?

Stabilität

Gibt es für das Testwerkzeug Ausnahmebehandlungen? Was passiert, wenn ein Testfall fehlschlägt?

Libraries/Out-of-the-Box-Lösungen

Besteht Zugriff auf externe Programme? Kann man benutzerdefinierte Module erstellen? Gibt es für das Tool eine Community?

Dokumentation

Kann man im Werkzeug Kommentare in Testprogramme eingeben? Läuft die Dokumentation automatisch oder manuell?

Logging

Werden Testergebnisse und Testdurchführung mit Pass oder Fails geloggt? Gibt es dafür extra Tags?

Recording

Können Testaufgaben aufgezeichnet werden? Werden nur logische Aktionen oder auch analoge Aufgaben aufgezeichnet? Dies ist notwendig, um Testabläufe aufzunehmen und wieder abspielen zu können.

Checking

Wie funktioniert der Ist-Soll-Vergleich der Testfälle? Benötigt man dafür eine Anpassung des Werkzeugs oder bietet das Tool eine automatische Erkennung?

Events

Beim Test von Steuergeräten werden Ereignisse immer eine große Rolle spielen. Kann das Tool damit umgehen? Sind Ereignisse erkennbar (synchron oder asynchron)?

Files und Versionskontrolle

Bietet das Tool eine Anbindung an das Versionsmanagement? Wie lautet das Fileformat? Wie viel Speicherplatz gibt es für die Testaufgaben?

Parallelität

Kann das Tool parallele Aufgaben durchführen?

Kosten und Lizenz

Die Frage nach den Kosten spiegelt sich in den folgenden Phasen wider:

- Evaluierungsphase,
- Nutzungsphase,
- Supportkosten,
- Wartungs- und Updatekosten,
- Lizenztyp,
- extra Infrastruktur.

Auf dem Markt existieren viele Anbieter. Die Automatisierungswerkzeuge aus der Automobilindustrie stellen einen Vorschlag dar. Hier muss zwischen verschiedenen Anbietern und Werkzeugen entschieden werden. Die vorgestellten Kriterien der Rahmenbedingungen für Werkzeuge stellen einen Leitpfad dar. Es empfiehlt sich, eine Evaluierung der Werkzeuge vor dem Kauf durchzuführen. Die Evaluierungsphase und eine genaue Betrachtung der vorgestellten Aspekte kann eine Entscheidung bringen. Ein Unternehmen in der Medizintechnikbranche kann eine Entscheidungsmatrix bilden, um eine Gewichtung vornehmen zu können. Die drei Faktoren Zeit, Kosten und Qualität spielen bei der Entscheidung die entscheidende Rolle. Die Vorteile der Testautomatisierung liegen auf der Hand. In der Medizintechnik muss geprüft werden, inwieweit die Zertifizierung der Werkzeuge wichtig ist. Der nationale als auch der internationale Markt müssen betrachtet werden.

Nachdem die Rahmenbedingungen veranschaulicht wurden, muss nun eine pragmatische Antwort auf die am Anfang des Kapitels gestellte Frage abgegeben werden. Inwieweit können die Automatisierungskonzepte auf die Medizintechnik und im Speziellen auf die Entwicklung von Embedded Systems in der Medizintechnik übertragen werden?

Die Antwort soll an folgenden Aspekten ausgerichtet sein:

- Inwieweit können die SPICE-Stufen ENG. 6 bis ENG. 10 in der Entwicklung von Embedded Systems in der Medizintechnik angewendet werden?
- Können die vorgestellten Werkzeuge der SPICE-Stufen eingesetzt werden?
- Wie steht das Projekt aus ökonomischer Sicht da? Bestehen Schulungsbedarf und Bedarf an Investitionen?
- Wollen die Teammitglieder des Testteams die Werkzeuge einsetzen?
- Kann das Unternehmen die Werkzeuge für zukünftige Projekte einsetzen?
- Inwieweit können die Werkzeuge an die domänenspezifischen Anforderungen angepasst werden?
- Sind die Werkzeuge zertifiziert, so dass als Ergebnis der Testaufgaben ein echter Nachweis für die Zertifizierungsbehörden vorliegt?

Auf der Stufe der ENG. 6 nach SPICE kann das Konzept übertragen werden, wenn die Programmiersprache unterstützt wird und ausreichende Ressourcen im Projekt vorhanden sind. Im Embedded System-Bereich wird oft die Programmiersprache C eingesetzt.

Die vorgestellten Konzepte – ob Open Source oder kommerzielle – mit dem Tool können dann in der Medizintechnik eingesetzt werden. Einer der wichtigsten Vorteile dieser Stufe ist die frühe Erkennung von Fehlern, so dass schon auf der Modulebene die Fehler gefunden und beseitigt werden. Das spart Zeit und Kosten.

Auf der Stufe der ENG. 7 nach SPICE kann das Konzept übertragen werden, wenn die interne Schnittstelle des Embedded Systems durch das Testwerkzeug unterstützt wird und ausreichende Ressourcen im Projekt vorhanden sind.

Im Embedded System-Bereich werden meist Standard-Schnittstellen eingesetzt. Die vorgestellten Konzepte mit dem Tool können dann auch in der Medizintechnik eingesetzt werden. Einer der wichtigsten Vorteile dieser Stufe ist die

frühe Erkennung von Fehlern, so dass schon früh im Projekt die Fehler der Integration der Software-Module gefunden und beseitigt werden können. Das spart Zeit und Kosten.

Auf der Stufe der ENG. 8 nach SPICE kann das Konzept übertragen werden, wenn der Prozessor und seine Debugger-Schnittstelle des Embedded Systems durch das Testwerkzeug unterstützt werden und ausreichende Ressourcen im Projekt vorhanden sind.

Auf der Stufe der ENG. 9 und ENG. 10 nach SPICE kann das Konzept übertragen werden, wenn die externe Schnittstelle des Embedded Systems durch das Testwerkzeug unterstützt wird und ausreichende Ressourcen im Projekt vorhanden sind.

Dieses Kapitel hat die Rahmenbedingungen der Entwicklung von Embedded Systems und speziell die enthaltene Software dargestellt. Inwieweit die Branche der Automobilindustrie mit der Medizintechnik Gemeinsamkeiten hat, wurde beschrieben. Der Kapitel hat eine pragmatische Antwort auf die Fragestellung gegeben, inwieweit die Testautomatisierungskonzepte und Werkzeuge aus der Automobilindustrie in die domänenspezifischen Anforderungen der Medizintechnik übertragen werden kann.

Neben den technischen Anforderungen existieren organisatorische Anforderungen, die bei Übernahme der Konzepte geklärt werden sollten. Sie hängt vor allem vom Unternehmen, seiner Größe, finanziellen Lage und von den projektspezifischen Anforderungen ab. Die Aktivitäten der Testautomatisierung sollten nicht nur für das aktuelle Projekt gelten, sondern eine strategische Entscheidung auf der Ebene des gesamten Unternehmens und seiner Forschungs- und Entwicklungsabteilung darstellen, die dann für zukünftige Projekte wieder verwendet werden kann.

Die Testautomatisierung ist mit Personal- und Sachkosten verbunden, vor allem wenn das Personal durch Schulung auf die Automatisierungswerkzeuge eingearbeitet werden muss. Diese Kosten amortisieren sich im Laufe des Projekts oder durch die Wiederverwendung der Testautomatisierungskonzepte und deren Werkzeuge in zukünftigen Projekten im Unternehmen.

6 Bewertung der Testautomatisierung

Testautomatisierung stellt eine strategische Entscheidung auf der Ebene des gesamten Unternehmens und seiner Forschungs- und Entwicklungsabteilung dar. Das Unternehmen muss in die Lage versetzt werden, seine Strategie bewerten zu können. In diesem Kapitel wird eine Methode vorgestellt, um die Testautomatisierung als Strategie im Unternehmen zu bewerten. Die Sustainability Balanced Scorecard wir als Bewertungsmethode vorgeschlagen.

6.1 Die Sustainability Balanced Scorecard

Das Konzept der Balanced Scorecard entstand Anfang der 1990er Jahren in den USA als ein neuer Ansatz der Leistungsmessung im Unternehmen. Damals wurde kritisiert, dass die bestehenden Kennzahlensysteme zu einseitig, kurzfristig und vergangenheitsorientiert ausgerichtet sind. Zudem wurde beanstandet, dass detaillierte Aussagen über Ursachen bestimmter Entwicklungen nur beschränkt möglich seien und eine unzureichende Verzahnung der strategischen und operativen Ebene bestehe.[77][78]

Unter der Leitung von R. S. Kaplan und D. P. Norton wurde mit dem Ziel, die vorhandenen Kennzahlensysteme den gestiegenen Anforderungen der Unternehmen anzupassen und ein neues System zu entwickeln, ein Forschungsprojekt mit zwölf US-amerikanischen Unternehmen durchgeführt. Das System sollte sich nicht mehr auf monetäre Größen und finanzielle Faktoren der Unternehmensführung fokussieren. Angestrebt wurde ein ganzheitliches und ausgewogenes Modell, das strategische und operative Anforderungen durch eine ausgewogene Berücksichtigung von finanziellen und nicht-finanziellen Kennzahlen integriert.[79]

---

[77] Vgl. Hoffmann 1999, S. 49.

[78] Vgl. Wirtschaftlexikon, http://wirtschaftslexikon.gabler.de/Definition/balanced-scorecard.html, angesehen am 11.09.2012

[79] Vgl. Kaplan/Norton, 2001, S. 11.

Aus dem Forschungsprojekt entstand das Konzept der Balanced Scorecard, bei dem nun die traditionellen finanziellen Kennzahlen durch eine Kunden-, eine interne Prozess- und eine Lern- und Entwicklungsperspektive ergänzt wurden.[80] Dieses Konzept soll laut Kaplan und Norton den strategischen Führungsprozess im Unternehmen unterstützen bzw. als Handlungsrahmen für diesen Prozess dienen.

Die Balanced Scorecard beinhaltet vier Perspektiven, die sich folgendermaßen charakterisieren lassen:[81]

Die **finanzielle Perspektive** zeigt zum einen durch ihre Kennzahlen die Leistung auf, die von der Unternehmensstrategie erwartet wird, zum anderen dient sie gleichzeitig als Ziel für die anderen Perspektiven, die zumindest teilweise mit ihr verknüpft sind. Als Kennzahlen dienen bekannte Größen wie zum Beispiel die Eigenkapitalrendite etc.

Die **Kundenperspektive** drückt ebenfalls mit Kennzahlen die Zielvorgaben des Managements hinsichtlich der Kunden und des Marktes aus, in den das Produkt eingeführt wird oder bereits mit anderen Produkten konkurriert.

Durch die **Prozessperspektive** sollen die internen Prozesse eines Unternehmens so strukturiert und geplant werden, dass die Ziele der finanziellen Perspektive und der Kundenperspektive erreicht werden.

Durch die **Lern- und Entwicklungsperspektive** werden die Maßnahmen und die Infrastruktur beschrieben, die dazu führen, die Ziele der drei vorangegangenen Perspektiven zu erreichen. Dabei spielt für viele Unternehmen insbesondere die Qualifizierung von Mitarbeitern, das Informationssystem sowie die Motivation der Mitarbeiter eine wichtige Rolle.[82]

---

[80] Vgl. http://wirtschaftslexikon.gabler.de/Definition/balanced-scorecard.html
[81] Vgl. Kaplan/Norton 1997, S. 9, 33.
[82] Vgl. http://wirtschaftslexikon.gabler.de/Definition/balanced-scorecard.html

In jeder Perspektive bestimmen Ergebnisgrößen und Leistungstreiber die Gestaltung. Hierbei wird nach gegenseitiger Ursache-Wirkungsbeziehung analysiert, um damit ein Gleichgewicht zwischen kurzfristigen und langfristigen Zielen zu erreichen und die Balance von gewünschten Resultaten und den Leistungstreibern mit ihren einzelnen Maßnahmen zu finden. Das Konzept ist jedoch nicht starr vorgegeben. Es soll ausdrücklich an die unternehmens- und strategiespezifischen Umstände angepasst werden.[83] Allerdings beinhalten die Bereiche in allen Perspektiven fast ausschließlich marktwirtschaftliche und ökonomische Prozesse. Außerhalb des Marktmechanismus ablaufende Vorgänge werden kaum betrachtet.

Da jedoch zunehmend auch ökologische und soziale Aspekte eine wichtige Rolle spielen, sollten diese ebenfalls berücksichtigt werden. Ziel sollte deshalb sein, eine „Sustainability oder Sustainable Balanced Scorecard“ (SBSC) zu entwickeln, die die drei Dimensionen der Nachhaltigkeit im System berücksichtigt. Das neue Konzept soll im Bereich der Nachhaltigkeit eine verstärkte Orientierungsfunktion für Unternehmen darstellen.

Inzwischen gibt es Konzeptionen, bei denen die Balanced Scorecard als Grundlage dient und die drei Dimensionen der Nachhaltigkeit integriert werden. Grundsätzlich gibt es drei mögliche Vorgehensweisen bei der Formulierung einer SBSC:

- Umwelt- und Sozialaspekte können in die vier Perspektiven einer bereits bestehenden Balanced Scorecard eingearbeitet werden,
- eine bestehende Balanced Scorecard kann um eine fünfte, zusätzliche Perspektive zur Berücksichtigung der Umwelt- und Sozialaspekte erweitert werden und
- neben der bereits bestehenden Balanced Scorecard wird eine separate Sustainability Balanced Scorecard entwickelt, die die Umwelt- und Sozialaspekte zusammenfassend darstellt.[84]

---

[83] Vgl. Kaplan/ Norton 1997, S. 33.
[84] Vgl. Figge et al. 2001, S. 20 ff.

6.2 Sustainability Balanced Scorecard der Testautomatisierungsstrategie

Für die SBSC werden Indikatoren in Bezug auf die zu untersuchende Strategie benötigt. Im Fall der Testautomatisierung werden folgende Indikatoren wie in Abbildung 14 herangezogen.

		Balanced Scorecard Prespektive			
		Finanzperspektive	Kundenperspektive	Prozessperspektive	Lern- u. Entw. Perspektive
Dimensionen der Nachhaltigkeit	Ökonomisch	Rendite erhöhen Absatzsteigerung Cashflow	Kundenzufriedenheit Kundenbindung Neukunden	Durchlauf- und Bearbeitungszeit Produktivitätauslastung Fehler- Ausfallquote	Mitarbeiterzufriedenheit Mitarbeitertreue Ausfallzeiten Forschung & Entwicklung
	Sozial	Faire Konditionen für die Tester (Gehälter und Bonus System). Work-Life-Balance	Informationspolitik Gewährleistung Image	Arbeitsunfälle Frauenquote Schwerbehindertenquote Vorschlagswesen	Fort- u. Weiterbildung Mitarbeiterschulung Beratungskosten Support
	Ökologisch	Transport Reisetätigkeit Betriebsdauer	Recyclierbarkeit und Reuse Produktverantwortung Entsorgung	Emissionen Energieverbrauch Wasserverbrauch Abfallmenge	Normen und Standards Gesetzte Best practice

Abbildung 14: Merkmale der SBSC für die Testautomatisierungsstrategie

Ziel der Indikatorenbildung ist, die subjektiven und objektiven Bewertungen im Laufe des Projekts in Bezug auf die Automatisierungsstrategie und deren Artefakte (Werkzeuge, Methoden) zu berücksichtigen. Die Indikatoren in Abbildung 14 stellen einen Vorschlag dar, der sich von Projekt zu Projekt ändern kann. Die Indikatoren sollen Unternehmensgröße, Projektumfang und Anzahl der beteiligten Mitarbeiter berücksichtigen.

Im SBSC werden Indikatoren für die folgenden Perspektiven gebildet: Finanzperspektive, Kundenperspektive, Prozessperspektive und Lern- und Entwicklungsperspektive. Dabei wird die vertikale Bewertung nach den Nachhaltigkeitsdimensionen untergliedert: ökonomisch, sozial und ökologisch. Jeder dieser Indikatoren hat einen Bezug zur Automatisierungsstrategie. Die Auswahl der Indikatoren hängt von der subjektiven Wahrnehmung der einzelnen Personen im Testteam und der objektiven Bewertung des Teams als Einheit ab. Folgende Aspekte spielen bei der Auswahl der Indikatoren eine Rolle:

- fachliche Kenntnisse der Rollen im Testteam,

- Kosten der Testumgebung und der Automatisierung,
- Benutzerfreundlichkeit der Testumgebung (Kann man die Testautomatisierung einfach bedienen?),
- Beratung, Support und Schulungskosten für die Mitarbeiter im Testteam,
- Erfahrung des Testmanagers,
- Auswahl von geeigneten Standard-Werkzeugen (Stand Alone Software),
- eingesetzte Programmiersprache (Compiler, Open Source oder kommerziell),
- hemmende und fördernde Faktoren auf sozialer Ebene im Testteam bei der Arbeit an der Testumgebung,
- Mitarbeiterfluktuation und Gesundheitszustand der Tester (Kann die Arbeit an der Testumgebung gut dokumentiert werden, um eventuell andere Tester schnell einarbeiten zu können?).

Alle ausgearbeiteten Indikatoren müssen in einem Workshop mit dem Testmanager als Moderator ausgewählt werden. Eine Gewichtung sollte anhand einer Checkliste durchgeführt werden, um bei einem Brainstorming im Workshop nur Indikatoren in das SBSC aufzunehmen, die im Projekt von großem Interesse sind. Zur Veranschaulichung werden einige Indikatoren exemplarisch betrachtet:

Mitarbeiterschulung: Es handelt sich um die Schulung der Mitarbeiter (Tester) zur Anwendung der Automatisierungswerkzeuge. Hier kann man nach der Umsetzung der Strategie bewerten, ob die Schulung auf das Projekt positive oder negative Wirkung hatte.

Rendite: Es handelt sich um die ökonomischen Vorteile, die man erzielen kann, wenn das Testteam die Automatisierung vollzogen hat.

Transport: Wie viele Reisen musste das Testteam in Anspruch nehmen? Solche Reisen können für Support, Schulung und Beratung in Anspruch genommen werden.

Für jeden Indikator wird vor der Einführung der Strategie und nach der Umsetzung eine Entscheidung für dessen Gewichtung und Bedeutung getroffen. Wenn der Indikator positiv für das Projekt ist, wird der Indikator grün markiert. Bei negativen Auswirkungen wird der Indikator rot markiert. Wenn keine Entscheidung getroffen werden kann, ob der Indikator positive oder negative Auswirkungen hat, wird der Indikator gelb markiert. Am Ende entsteht ein Ampelsteuerungsinstrument, mit dem die Automatisierungsstrategie bewertet werden kann.

Die SBSC stellt ein Instrument dar, mit dessen Hilfe die Automatisierungsstrategien und deren Werkzeuge und Projekte besser bewertet werden können. Somit können Entwicklungsteams aus den Erfahrungen und Vergleichen der verschiedenen eingesetzten Artefakte im Test Tipps und Best Practice für zukünftige Arbeiten entnehmen.

7 Zusammenfassung und Ausblick

Zusammenfassend ist festzustellen, dass die Übertragung der Konzepte der Testautomatisierung aus der Automobilindustrie in die Medizintechnik-Branche möglich und empfehlenswert ist. Die Arbeit hat die ökonomischen Vorteile deutlich hervorgehoben. Eine Testautomatisierung kann helfen die Meilensteine im Projekt besser zu erreichen, um qualitative Software nach den Maßgaben der Projektrahmenbedingungen Just in Time zu liefern.

Die Arbeit betrachtete die Aufgabenstellung pragmatisch. Die Grundlagen des Softwaretests wurden veranschaulicht und die verschiedenen Testarten besprochen. Die Unterteilung der Testaufgabe in verschiedene Teststufen und Rollen macht klar, dass die Testautomatisierung auf diese Rollen und Stufen angewendet werden muss. Die Analyse, wie das Softwaretesten in die Qualitätssicherung im Unternehmen integriert werden sollte, machte deutlich, dass diese Aufgabe nicht zu unterschätzen ist. Im Grundlagenkapitel wird dargestellt, dass die Testautomatisierung eine Strategie im Gebiet der Softwaretests ist, dass das Testteam und vor allem der Testmanager das Testkonzept klar klassifizieren muss und warum, wie und wann automatisiert werden sollte.

Im Kapitel „Softwaretest aus Sicht der funktionalen Sicherheit und Reifegradmodelle“ wurden die Gesetze, Normen und Standards veranschaulicht. Der Fokus lag auf den Anforderungen, die aus Sicht der Normen und Standards für den Softwaretest gelten. Das Kapitel hat gezeigt, dass es empfehlenswert ist, die branchenspezifische Norm oder den entsprechenden Prozess einzusetzen, da somit weniger Aufwand in der Interpretation der Anforderungen entsteht. Die vorgestellten Normen und Reifegradmodelle stellen frei, ob die Testautomatisierung im Entwicklungsprojekt eingesetzt wird oder nicht. Es werden Techniken und Methoden der Testverfahren vorgeschlagen, die je nach Sicherheitsstufe eingesetzt werden müssen.

Das Kapitel „Testautomatisierungskonzepte“ hat ein Beispiel für die Testautomatisierung nach dem Prinzip der Teststufen in der Automobilindustrie gezeigt. Das Kapitel betrachtete die Testautomatisierung als Strategie, die Vorteile mit sich bringt. Um aber die Konzepte auf bestimmte Domänen und ihre spezifischen Anforderungen übertragen zu können, müssen Problemfelder erst betrachtet werden.

Das Kapitel „Wirtschaftliche Prüfung der Testautomatisierung“ hat gezeigt, dass es wirtschaftliche Vorteile durch die Automatisierung in Bezug auf die Qualitätssicherung durch das Testen gibt.

Im letzten Kapitel dieser Arbeit wurde eine Methode vorgestellt (SBSC), um die Testautomatisierung als Strategie im Unternehmen bewerten zu können.

Testautomatisierung bringt sowohl einen ökonomischen als auch einen sozialen Vorteil. Der ökonomische Vorteil kommt zum Tragen, wenn folgende Prozesse in das Projekt implementiert werden: Regressionstest, Iterative Softwareentwicklung, Weiterentwicklung der Software, Wartung und Pflege der Software und Migration der Software auf eine andere Architektur oder andere Hardware-Plattform. Auf der sozialen Ebene kann die Automatisierung das Klima im Test- und Entwicklungsteam fördern, da die Testautomatisierung eine Innovation im Projekt darstellt.

Wenn die vorgestellten Konzepte eingehalten werden, dann bringt die Automatisierung folgende Vorteile:

- Der Pfad und Weg zum Testergebnis ist klar und einfach gestaltet. Das ermöglicht mehreren Mitarbeitern und der Führungsebene flexibler zu sein.
- Die Bedienung der Werkzeuge ist einfacher und transparent.
- Das gewonnene Know-how kann einfach auf mehrere Projekte übertragen werden.
- Eine Erhöhung der Qualität unter Einhaltung der Abgabetermine ist möglich.

- Die Rückverfolgbarkeit im Projekt ist gewährleistet.
- Man kann höhere Bewertungen durch den Kunden anhand der Bewertungsmodelle (SPICE, CMMI) erreichen.

Die Arbeit hat gezeigt, dass die Automobilindustrie ein Vorbild für andere Industriezweige ist. Nicht nur die Optimierung des Testprozesses, auch die organisatorischen und ökonomischen Vorteile können sich im globalen Wettbewerb amortisieren. Bei einer Übertragung der Konzepte der Testautomatisierung in die Medizintechnik sollten folgende Aspekte beachtet werden:

- verfügbares Budget und Zeit im Projekt,
- Auswahl der Werkzeuge,
- Sicherheitsstufen und Ziele,
- Know-how der Mitarbeiter,
- Komplexität der Software,
- der Kunde und die gesetzlichen Rahmenbedingungen.

Die Entscheidung für die Testautomatisierung sollte immer dazu beitragen, schneller, qualitativer und besser zu sein als andere Unternehmen. Dies gilt sowohl für die Automobilindustrie als auch für die Medizintechnik.

Mit dieser Arbeit ist die Testautomatisierung in der Medizintechnik nicht abgeschlossen. Die vorgestellten Konzepte in der Automobilindustrie müssen noch evaluiert werden. Es empfiehlt sich, die Konzepte in einem Pilotprojekt zu evaluieren. Inwieweit die Konzepte einen ökonomischen, innovativen Vorteil bergen, lässt sich mit dem Pilotprojekt bestätigen. Dabei sollten die Qualitätsverbesserung und -optimierung im Vordergrund stehen.

Die Arbeit hat Teststufen nach SPICE vorgeschlagen. Die weiteren Automatisierungskonzepte nach Einsatzgebieten müssen noch untersucht werden. Es ist notwendig, eine Studie in der Medizintechnik über Softwaretestprozesse und Testautomatisierung durchzuführen, um Lücken und Schwachpunkte herauszufinden. Die Studie sollte diese Arbeit ergänzen und die Ansatzpunkte für zukünftige Arbeiten darstellen. Die ökonomischen, innovativen und sozialen Vorteile wurden in dieser Arbeit verdeutlicht.

Die Studie sollte Prozesse und verwendete Werkzeuge fokussieren, um mögliche Synergien aus dieser Arbeit aufzuzeigen und die praktische Durchführung der Testaufgaben in der Medizintechnik zu beschreiben. Im Rahmen der Studie sollte eine Befragung von Herstellern medizinischer Produkte durchgeführt werden, um eine Statistik über eingesetzte Prozessmodelle und Werkzeuge auszuarbeiten.

8 Anhang

8.1 Glossar

Im Umfeld der Qualitätssicherung von Softwareprodukten - insbesondere im professionellen Testmanagement und Projektmanagement - werden verschiedene Begriffe eingesetzt, die hier erklärt werden. Das Glossar wird nach der Erscheinung der Begriffe und deren Bedeutung für das Verständnis sortiert:

DUT: Device Under Test ist das Testobjekt.

Testfall: „Umfasst folgende Angaben: die für die Ausführung notwendigen Vorbedingungen, die Menge der Eingabewerte (ein Eingabewert je Parameter des Testobjekts) und die Menge der erwarteten Sollwerte, die Prüfanweisung sowie erwarteten Nachbedingungen.“[85]

Testphase bzw. Teststufe: „Abstufung der Anforderung an die Testtiefe von Testfällen zu einem bestimmten Entwicklungsstand.“[86]

Testmanagement: „Das Testmanagement befasst sich mit der Planung und Steuerung der Aktivitäten im Testprozess während der gesamten Testphase.“[87]

Testplan (auch Testkonzept) beschreibt ein Artefakt im Testmanagement von Software, was nach einer bestimmten Norm mit Inhalt gefüllt werden soll. Es gehört zu den Aufgaben des Testmanagers, am Beginn der Testaufgaben ein Konzept aufzustellen.

Qualitätssicherung in der Software: Diagnostische Maßnahmen (z. B. Testen) zur Bestimmung der Qualität der Software in einem Produkt.

Der **Logische Testfall** ist ein Testfall ohne Angabe von konkreten Werten für die Eingabe und Ausgabe.

---

[85] Vgl. Andreas/Tilo (2006), S. 254.
[86] Vgl. Drescher (2010), S.2.
[87] Vgl. Drescher (2010), S.3.

Der **Konkrete Testfall** ist ein Testfall mit Angabe von konkreten Werten für die Eingabe und Ausgabe.

Das **Black-Box-Testverfahren** testet auf Grundlage der Anforderungen und Spezifikationen des Testobjektes, wobei die innere Struktur des Testobjektes (Implementierung, Source Code) nicht herangezogen wird. Systemtests (ENG. 9,10 nach SPICE) werden meistens mit Black-Box-Verfahren durchgeführt.

Das **White-Box-Testverfahren** testet auf Grundlage der inneren Struktur des Testobjekts (Implementierung und Source Code). Komponententests (ENG.6 nach SPICE), Integrationstests (ENG. 7 nach SPICE) und Softwaretests (ENG.8 nach SPICE) werden meistens mit White-Box-Verfahren durchgeführt.

Der **Regressionstests** ist ein Test, der nach Änderungen am System zu wiederholen ist. Damit wird sichergestellt, dass das System nach Äderungen keine Seiteneffekte verursacht.

Änderung beschreibt die Software nach Korrektur von falschem Verhalten.

Die **funktionale Anforderung** beschreibt das funktionale Verhalten des Systems.

Der **funktionale Test** leitet sich aus funktionalen Anforderungen ab.

Funktionalität spezifiziert das Verhalten einer Komponente.

Die **Anforderung** beschreibt das Soll-Verhalten.

Debuggen ist nicht gleich Testen. Mit einem Test wird die Fehlerwirkung erkannt und systematisch aufgedeckt. Durch den Prozess des Debuggens versucht man, die Stelle des Defektes uns seine Ursache zu lokalisieren, um sie zu beheben.

Debugger ist ein Werkzeug, mit dem das Debuggen durchgeführt werden kann.

„**SPICE** bildet den Rahmen für eine einheitliche Bewertung der Leistungsfähigkeit einer Organisationseinheit durch Bewertung der gelebten Prozesse.“[88]

CMMI ist ein Assessment Modell, das die Grundannahme verfolgt, dass eine Verbesserung der Prozesse der Softwareerstellung sowohl die Qualität der Software als auch die Planung und Umsetzung der Managementaufgaben verbessert.

Realtime und Embedded Systems (Echtzeit- und eingebettete Systeme oder kurz RTES) sind als normale Computer verbreitet. RTES ist ein übergeordnetes System, eingebettet in Hardware- und Software-Teile, das mit seiner Umgebung interagiert. Es wird auch als Anlage bezeichnet. Signale werden über Sensoren empfangen. Über Aktoren werden Signale nach außen gegeben.[89]

„Bei einem **Release** muss es sich nicht um ein fertiges Produkt handeln. Es ist eine Art Zwischenstand bei der Entwicklung eines Produktes. Es wird zwischen internen Releasen für die Entwicklung oder eine Produktpräsentation und externen Releasen für die Endkundenauslieferung unterschieden.“[90]

Meilenstein ist ein „Zwischenstand im Projektablauf, der zum Ende eines bestimmten Projektabschnitts erreicht werden muss. Die Erreichung des Meilensteins ist Voraussetzung für den Beginn der folgenden Projektphase oder eines weiteren Meilensteins. Der Meilenstein definiert ein Ergebnis und ist in der Regel mit einem festen Termin gekoppelt.“[91]

Auftraggeber ist im Projektumfeld derjenige, der den Auftrag zum Projekt erteilt und in der Regel auch das Budget zur Verfügung stellt.

---

[88] Vgl. Spillner (2011), S. 188.
[89] Vgl. Vischnow (2010), S. 141.
[90] Vgl. Drescher (2010), S. 3.
[91] Vgl. Drescher (2010), S. 3.

8.2 Eigenschaften von Tessy

Tessy ist ein Tool für automatisierte Unit-, Modul- und Integrationstests von eingebetteter Software. Tessy ist das unverzichtbare Testwerkzeug. Im Folgenden sind die Eigenschaften von Tessy aufgeführt:

- „Automatisierter Unit- und Modultest von eingebetteter Software,
- Komponententest = Integrationstest der Units,
- einfacher Einstieg – es muss keine Scriptsprache gelernt werden.
- Testtreiber und Stub-Funktionen werden automatisch generiert,
- ideal für Regressionstests,
- wesentlich um Zertifizierungen nach IEC 61508, DO-178B und anderen Standards zu erhalten,
- misst Code-Überdeckung,
- erzeugt Testdokumentation in verschiedenen Formaten (XML, HTML, Word, Excel, HLP),
- unterstützt ASAP2,
- steuert HiTOP und andere gängige Debugger zur automatischen Testausführung,
- beinhaltet den Klassifikationsbaumeditor (Classification Tree Editor CTE) zur Testfallspezifikation,
- bezieht Hardware in die Tests ein.“[92]

Tessy ist für die Software-Entwicklung von sicherheitskritischen Anwendungen nach IEC 61508 und ISO 26262 qualifiziert.

---

[92] Vgl. http://www.hitex.com/index.php?id=module-unit-test&L=2, zuletzt angesehen am 8.09.2012.

8.3 Eigenschaften von EXAM

EXAM stellt eine Methodik zur Darstellung, Durchführung und Auswertung von plattformunabhängigen, wiederverwendbaren Testfällen dar. Die Verifikation und Validierung von Embedded Systems und Fahrzeugfunktionen benötigt eine automatisierte Testumgebung und Werkzeuge, die das unterstützen. Es hat sich in der Fahrzeugerprobung die Verwendung automatisierter Tests insbesondere für HiL-Prüfstände etabliert. Die Wiederverwendbarkeit und die Austauschbarkeit der vorhandenen Testfälle spielen eine entscheidende Rolle bei der Reduzierung des Erprobungsaufwandes für Regressionstests. Diese Reduzierung des Aufwandes gilt auch für die Entwicklung von neuen Fahrzeugvarianten und -Modellen. Bereits erprobte Testfälle können mit minimalem Anpassungsaufwand wiederverwendet werden.[93]

Die EXAM-Methodik spezifiziert die Testfälle in UML in formaler Form, dazu bietet die Methodik eine Bibliothek mit Testfällen und Testfallfragmenten an.

Mit der Bibliothek stehen für alle Beteiligten intuitiv lesbare und leicht wiederverwendbare Testfälle zur Verfügung. „Aus dem UML-Diagramm wird mit der EXAM-Toolsuite das fertige, ausführbare Test-Programm generiert. Die Testingenieure konzentrieren ihr Wissen und ihre Erfahrung auf die Testmodellierung – die Generierung der ausführbaren Testfall-Skripte und die Darstellung des Testergebnisses übernimmt EXAM. Zur EXAM-Methodik gehört eine klare Rollenverteilung, mit der die Produktivität und Qualität des Erprobungsprozesses gesteigert werden. Mit der EXAM-Methodik existiert ein OEM- und Testsystemunabhängiges Werkzeug, mit dem Testingenieure bei der Erstellung automatisierter, wiederverwendbarer Tests bestmöglich unterstützt werden.“[94]

[93] Vgl. Kiffe (2012), S. 1.

[94] Vgl. Kiffe (2012), S. 2.

8.4 Eigenschaften der modularen Test-Hardware, VT-System von Vector GmbH

Das VT-System von Vector GmbH stellt ein I/O-Schnittstellenmodul für den Steuergerätetest dar.

Um das System (Embedded System) komplett testen zu können, muss die Möglichkeit gegeben sein, dass das System mit den Kommunikations-Netzwerken und den I/O-Schnittstellen an das Testsystem angeschlossen werden kann. Das Vector VT-System übernimmt diese Aufgabe. „Es vereinfacht den Aufbau von Prüfständen und HIL-Testsystemen erheblich, indem es alle für die Beschaltung eines I/O-Kanals notwendigen Komponenten in ein Modul integriert. Beispiele für I/O-Kanäle sind der Ausgang eines Steuergeräts zur Ansteuerung einer Scheinwerferlampe oder ein Eingang für den Anschluss eines Helligkeitssensors. Typische Testaufgaben für das VT-System sind:

- Messung von Ausgangssignalen (Spannung, PWM-Parameter),
- Darstellung von Fehlerzuständen wie Leitungsunterbrechungen und Kurzschlüssen,
- Aufschalten von Originalsensoren und –aktoren,
- Simulation unterschiedlicher Lasten und Sensoren.“ [95]

[95] Vgl. http://www.vector.com/vi_vt_system_overview_de.html, zuletzt angesehen am 8.09.2012

Literaturverzeichnis

BITKOM (2010): Test-Methodik für Embedded Software in der Automobil- und Medizintechnik.

Drescher, Arne (2010): Leitfaden für die Einführung einer Testautomation zur effizienten Qualitätssicherung von Softwareprodukten, Diplomarbeit, GRIN Verlag.

Figge, F., Hahn, T., Schaltegger, S., Wagner, M. (2001): Sustainability Balanced Scorecard – Wertorientiertes Nachhaltigkeitsmanagement mit der Balanced Scorecard.

Hoffmann, O. (1999): Performance Management – Systeme und Implementierungsansätze. Bern, Haupt Verlag.

Johner, Christian; Hölzer-Klüpfel, Mathias; Wittorf, Sven (2011): Basiswissen Medizinische Software, dpunkt-Verlag, Heidelberg.

Kaplan, R. S., Norton, D. P. (1997): Balanced Scorecard - Strategie erfolgreich umsetzen. Stuttgart.

Kaplan, R. S., Norton, D. P. (2001): Die strategiefokussierte Organisation: Führen mit der Balanced Scorecard. Schäffer-Poeschel.

Löw, Peter; Pabst, Roland; Petry, Erwin (2010): Funktionale Sicherheit in der Praxis, dpunkt-Verlag, Heidelberg.

Schneider, Kurt (2007): Abenteuer Software Qualität, dpunkt-Verlag, Heidelberg.

Seidel, Richard; Baumgartner, Manfred; Bucsics, Thomas (2012): Basiswissen Testautomatisierung, Konzepte, Methoden und Techniken, dpunkt-Verlag, Heidelberg.

Software Quality Lab, Ausgabe 2009/1, Testautomatisierung — Kosten & Nutzen.

Spillner, Andreas; Linz, Tilo (2006): Basiswissen Softwaretest, dpunkt.verlag.

Spillner, Andreas; Roßner, Thomas; Winter, Mario; Linz, Tilo (2011), Praxiswissen Softwaretest Testmanagement, dpunkt-Verlag, Heidelberg.

Teichreber, P. E. (2008): Praktische Software-Qualitätssicherung, Leitfaden für Testorganisation und – Dokumentation, Hanser Verlag

Thaller, Georg Erwin (2001): ISO 9001:2000 Softwareentwicklung in der Praxis, Heise Verlag.

Vigenschow, U. (2004): Objektorientiertes Testen und Testautomatisierung in der Praxis, Konzepte – Techniken und Verfahren, d-punkt Verlag.

Vigenschow, Uwe (2010): Testen von Software und Embedded Systems, dpunkt Verlag.

Zurawka, T. (2010): Systecs, Test-Methodik für Embedded Software in der Automobil- und Medizintechnik.

Beckert (Seminar) unter http://formal.iti.kit.edu/~beckert/teaching/Seminar-SoftwarefehlerSS03/Ausarbeitungen/ pfeifer.pdf angesehen am 13.07.2012

Internetquellen

Axel Springer AG 2012, URL: http://www.welt.de/motor/article13586655/Honda-ruft-fast-eine-Million-Autos-zurueck.html, zuletzt abgerufen am 13.07.2012.

Pfeifer, Martin, Berühmt berüchtigte Softwarefehler, URL: http://formal.iti.kit.edu/~beckert/teaching/Seminar-Softwarefehler-SS03/Ausarbeitungen/pfeifer.pdf, zuletzt abgerufen am 13.07.2012

Therac-25, Firma MicroNova, URL: http://www.exam-ta.de, zuletzt abgerufen am 13.07.2012.

Firma iSystem, URL: http://www.isystem.com/products/winidea, zuletzt abgerufen am 13.07.2012.

Firma Vector GmbH, URL: http://vector.com/, zuletzt abgerufen am 13.07.2012.

Firma Hitex, URL http://www.hitex.com/index.php?id=module-unit-test, zuletzt abgerufen am 13.07.2012.

Link, Sabine; Hafner, Martina: Testen auf Knopfdruck, URL: http://www.elektronikpraxis.vogel.de/themen/embeddedsoftwareengineering/testinstallation/articles/242263/, zuletzt abgerufen am 15.07.2012.

Kiffe, Gerhard (AUDI AG); Paggel, Andrea (T-Systems Enterprise Services GmbH), EXAM – Methodik zur Darstellung, Durchführung und Auswertung von plattformunabhängigen, wiederverwendbaren Testfällen, URL: http://www.fkfs.de/fileadmin/media/04_unternehmen/veranstaltungen/autotest/pdf_dokumente/paper_2008/paper_16_Audi_Kiffe.pdf, zuletzt abgerufen am 08.09.2012.

Eidesstattliche Versicherung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

20.09.2012, Haider Karomi