Wilhelm Büchner Hochschule
Private Fernhochschule Darmstadt

Fachbereich Informatik
Ostendstraße 3
D-64319 Pfungstadt

# Konzeption und Entwicklung einer Versionierungssoftware zur besseren Verwaltung von Macro- und SPS-Programmen als Bestandteil der Firmware von Präzisionsmessmaschinen

| | |
|---|---|
| Betreuer: | Prof. Dr. Freytag |
| Autor: | Maik Ledwina |
| Matrikelnummer: | 880767 |
| Anschrift: | Ringstraße 30 |
| | 76437 Rastatt |
| Abgabetermin: | 2. Juli 2016 |

# Zusammenfassung

Dies ist die Zusammenfassung.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Vorwort

Softwareentwicklung, ob für Anwendungen oder Firmeware, unterliegt einer stetigen Evolution. Neues kommt hinzu, bestehendes wird erweitert oder verbessert und Bugs werden beseitigt. Bei Software ist der Rhythmus, in welchem die Neuerungen und Erweiterungen zum Kunden kommen, meist durch einen vom Hersteller geplanten Produktlebenszyklus bestimmt. Nur Bugs werden in kürzeren Abständen durch Updates behoben. Bei der Firmware für Controller ist dies häufig anders, hier bestimmen die Anforderungen von Hardwareentwicklung und Kundenwünschen die Veränderungen und Entwicklung. Wird ein Problem bei einem Kunden festgestellt, werden Fehlerbehebungen nicht mit Updates innerhalb von Wochen sondern von Stunden benötigt bzw. erwartet. Denn das Gerät, dass durch den Controller betrieben wird, muss weiter arbeiten.

## 1.2 Untersuchungsgegenstand

Der Untersuchungsgegenstand dieser Diplomarbeit ist die Firmware eines Präzisionsmessmaschinencontroller und die mit ihr arbeitenden Macros und SPS Programme.

## 1.3 Ausgangspunkt

Die Software für den Controller wurde in einem kleinen Entwicklerteam entwickelt und über Jahre hinweg immer weiter ausgebaut und verbessert. Änderungen werden stets direkt integriert und den Kundenwünschen und Anforderungen angepasst. Bugs werden so schnell wie möglich behoben und die Lösung bei den betroffenen Kunden integriert. Jeder der Entwickler hat seinen festen Aufgabenbereich und damit auch das wissen über alle Neuerungen und Fehlerbeseitigungen. Bei Überschneidungen wird zusammengearbeitet jedoch jeder in seinem Aufgabengebiet. Dadurch wurde die Dokumentation des häufigeren auch etwas stiefmütterlich behandelt und es ist teilweise auch schwierig bis überhaupt nicht nachzuvollziehen, welche Änderungen wann integriert wurden bzw. welcher Kunde dieser bereits erhalten hat.

## 1.4 Aufbau

# Abkürzungsverzeichnis

| | |
|---|---|
| SCCS | Source Code Control System |
| RCS | Revision Control System |
| CVS | Concurrent Version System |
| SVN | Subversion |
| bzw. | Beziehungsweise |
| evtl. | eventuell |
| VSS | Visual Source Safe |
| CAA | Computer Aided Accuracy |
| SPS | Sepicher Programmierbare Steuerung |

# Bezeichnungen und Begrifflichkeiten

| | |
|---|---|
| Repository | Text |
| Mergen | text |
| Push | text |
| Klonen | text |
| Open-Source | Text |
| Stakeholder | Als Stakeholder bezeichnet man die Personen oder Personenkreise, die beim Erheben von Anforderungen ein direktes oder indirektes Interesse an der Lösung haben. |

# 2 theoretischer Teil

## 2.1 Versionskontrolle

### 2.1.1 Anfänge der Versionsverwaltung

Die Anfänge der Versionsverwaltung reichen bis in die 70er Jahre des vergangenen Jahrhunderts zurück. Anfang der 70er wurde bei dem amerikanischen Telekommunikationsunternehmen AT & T eines der ersten Versionsmanagement System entwickelt, Source Code Control System. Es war für die Verwaltung einzelner Dateien ausgelegt, die nur von einem Entwickler bearbeitet wurden. Ganze Projekte und Entwicklung im Team konnte mit diesem System nicht verwaltet werden. Etwa 10 Jahre später wurde an der Purdue University durch Walter F. Tichy das Versionierungstool RCS entwickelt. Dabei handelte es sich wieder um ein Dateien verwaltendes Versionierungsprogramm. Dies war bereits für die Entwicklung im Team ausgelegt, vom Aufbau her jedoch so, dass wenn ein Entwickler an einer Datei gearbeitet hat diese für den Schreibzugriff durch andere Entwickler blockiert war. Beide Systeme hatten gemein, dass sie für die Versionsverwaltung lokaler Dateien waren. Ein Jahr nach der Entwicklung von RCS wurde mit der Entwicklung von CVS (1986) begonnen. Ursprünglich als Servererweiterung für RCS gedacht, wurde das zuerst auf UNIX Shell-Skripten basierende System im Jahre 1989 in C neu programmiert. CVS verfolgte einen anderen Ansatz als seine Vorgänger, die Dateien und Projekte waren zentral auf einem Server abgelegt und die einzelnen Entwickler konnten sich ihre Arbeitskopien von der herunter laden. So konnten auch mehrere Entwickler an der gleichen Datei arbeiten. Auch war CVS in der Lage Verzeichnisstrukturen, wenn auch noch sehr eingeschränkt, zu verwalten. Viele der später entwickelten System bauen auf diesen Systemen auf.

### 2.1.2 Grundlagen der Versionsverwaltung

Der Ursprüngliche Gedanke der Versionsverwaltung war, dass man damit erfolgte Änderungen in unterschiedlichen Versionen sichtbar und differenzierbar machen

konnte. Es sollte möglich sein eine Historie der erfolgten Änderungen abzubilden und für den/die Entwickler sichtbar zu machen. Noch größere Bedeutung bekam das ganze als nicht mehr nur ein Entwickler ein Thema bearbeitete sondern mehrere Entwickler im Team arbeiteten. Die Entstehung von Open-Source-Projekten mit weltweit verteilten Entwicklern hat die Entwicklung entsprechender Versionierungstools beschleunigt und deren Funktionsumfang und Herangehensweise maßgeblich beeinflusst. Die Funktionsweise der Systeme lässt sich in Lokale, Zentrale und Verteilte Versionsverwaltung unterteilen.

- Lokale Versionsverwaltung
  Bei der Lokalen Versionsverwaltung waren die zu versionierenden Dateien lokal auf dem Rechner des Entwickler abgelegt. In diesen Bereich fallen auch nur die beiden zuerst entwickelten Systeme SCCS und RCS.

- Zentrale Versionsverwaltung
  Bei der Zentralen Versionsverwaltung werden die Daten bzw. Repositoryen auf einem zentralen Server abgelegt und die Entwickler können sich von diesem Server eine lokale Arbeitskopie herunter laden und nach dem Bearbeiten wieder zurück spielen. Die Versionsgeschichte ist nur auf dem Server ersichtlich und wenn man eine Information aus einer früheren Version benötigt ist zwingend eine Verbindung zum Server notwendig. In diesen Bereich fallen zwei der am bekanntesten Open-Source-Systeme CVS und Subversion. Aber auch viele der kommerziellen Lösungen, die auf dem Markt sind, beruhen auf diesem System.

- Verteilte Versionsverwaltung
  Bei der Verteilten Versionsverwaltung besitzt jeder Entwickler lokal ein eigenes Repository des Projekts an dem er arbeitet. Die einzelnen Repository werden zwischen den unterschiedlichen Entwicklern immer wieder synchronisiert und somit alle Änderungen verteilt bzw. zusammengeführt. Ein zentraler Server ist hier nicht notwendig. Sehr häufig wird jedoch ein zentrales Repository eingesetzt um von dort aus das Produktivsystem zu bilden oder neuen Entwicklern einen Zentralen Punkt zu schaffen an dem sie sich

ihre erste Arbeitskopie herunterladen können. Bei diesem System hat jeder Entwickler immer die komplette Versionsgeschichte mit in seiner Kopie des Repository und kann evtl. später entdeckte Bug's lokal in den früheren Versionen suchen und beheben.

Bei den unterschiedlichen Versionierungssystemen, gibt es auch unterschiedliche Konzepte, was die Arbeitsweise angeht. Man unterscheidet hier zwischen:

- Lock Modify Write
  Bei dieser Arbeitsweise, erstellt sich der Entwickler eine Arbeitskopie der Datei, welche er ändern möchte. Das System sperrt in dieser Zeit die Datei für Schreibzugriffe anderer Entwickler. So können von einer Datei nie zwei unterschiedliche Versionen entstehen und auch keine Konflikte beim Zusammenführen der Dateien. Der Nachteil daran ist, dass nur ein Entwickler an der Datei arbeiten kann und ein anderer, der evtl. dringende Änderungen einfügen muss, dies erst tun kann, wenn der erste Entwickler die Datei wieder freigegeben hat. Ein weiterer Nachteil, wenn eine Datei gesperrt war und es bei dem Entwickler der die Datei gesperrt hatte, einen Systemverlust gab, war die Sperre noch immer vorhanden und musste umständlich entfernt werden. Bekannteste Vertreter für diese Arbeitsweise sind RCS und VSS von Microsoft. Bei den Verteilten Versionsverwaltung ist diese Arbeitsweise ausgeschlossen.

- Copy Modify Merge
  Bei dieser Arbeitsweise, erstellt sich jeder Entwickler seine lokale Arbeitskopie und entwickelt in dieser, hat er einen Teil fertig gestellt und will diesen den anderen zur Verfügung stellen oder ihn in das Haupt Repository übertragen, führt er einen Merge aus. Das heißt, die von ihm geänderte Datei wird auf den Server überspielt. Sollte auf dem Server eine andere Dateiversion verfügbar sein, als die Basis auf der der Entwickler seine Kopie herunter geladen hatte, so entsteht ein Konflikt. Je nach Versionierungssystem, kann dieses Konflikte zum teil automatisch lösen. Kann ein System die wird nun nach einer gemeinsamen Basis beider Dateien gesucht und überprüft ob es die bei-

den Dateien zusammenfügen kann. Wurde der Code beider Dateien an unterschiedlichen stellen geändert so können diese System den Konflikt durch zusammenführen beider Dateien selbständig beheben. Wurden die gleichen Stellen im Code geändert, so muss dies manuell durch einen der beteiligten Entwickler behoben werden.

## 2.2 Stand der Technik

Die beiden aktuell am verbreitetsten Systeme sind Subversion und Git. Als Hauptgründe hierfür kann man sehen, dass beide einen sehr großen Umfang an Möglichkeiten und Flexibilität mitbringen und dadurch eines der beiden für die meisten aller zu versionierenden Projekte passend ist. Aber beiden haben ihre weite Verbreitung auch dadurch erfahren, dass sie Open-Source Projekte sind und beide bei großen Open-Source Projekten zum Einsatz kommen.

### 2.2.1 Subversion

### 2.2.2 Git

## 2.3 Grundlagen zum untersuchten Einsatzgebiet

### 2.3.1 Päzisionsmessmaschinen

Strenggenommen handelt es sich dabei eigentlich um Präzisionsmessgeräte in der Umgangssprache hat sich jedoch die Bezeichnung als Maschine durchgesetzt und wird fast einheitlich von allen Herstellern verwendet. Zum Einsatz kommen Präzisionsmessmaschinen überall dort wo an Produkte ein hohes Maß an mechanischer Qualität gefordert ist. So werden mit Präzisionsmessmaschinen alle möglichen gefertigten Bauteile, von kleine Schrauben und Zahnrädern bis hin zu Kompletten LKW, Schiffsmotoren oder Teile von Windkraftanlagen vermessen. Üblicherweise werden Präzisionsmessmaschinen in die Gruppen Portal-, Brücken- und Ständermessmaschinen unterteilt. Die einzelnen Achsen werden wahlweise Rollen- oder, wenn größere Präzision gefragt ist, Luftgelagert. Der Typische Messbereich solcher Maschinen erstreckt sich im Bereich der

- Zahnradmesstechnik
  bei einem Zylindrischen Messbereich mit Durchmessern von ø280mm bis ø4000mm

  und

- Koordinatenmesstechnik
  bei einem Quaderförmigen Messbereich von 500mm x 500mm x 400mm bis 4000mm x 16000mm x 3000mm in Portal- und Brückenbauweise und 1000mm x 600mm x 1000mm bis 56000mm x 3600mm x 3600mm in Ständerbauweise.

Von Präzisionsmessmaschinen spricht man, da die Genauigkeit mit der die Geräte messen können bei einem Maschinengrundfehler von 1µ und teilweise weniger

beginnen.

### 2.3.2 Controller

Als Controller wird im Allgemeinen eine Steuerung bezeichnet, welche in der Lage ist, durch integrierte oder dazugehörende Firmware oder Software, eine Maschine zu steuern und zu regeln. Im Falle des Untersuchungsgegenstand sind ein hohes Maß an Genauigkeit, Zuverlässigkeit und Qualität gefordert. Messmaschinen müssen, Positionen auf Zehntel Mikrometer halten, verfahren und positionieren können. Jede Präzisionsmessmaschine hat zusätzlich zur hohen Fertigungsgenauigkeit noch ein CAA Feld, welches die mechanischen Restfehler der Maschine enthält und entsprechend vom Controller ausgeglichen werden muss. An allen Achsen der Messmaschine befinden sich Positionsmesssysteme und an den Motoren der Achsen zusätzliche Tachometer. An jeder Messmaschine werden Messköpfe zur Aufnahme von Tastpunkten angebracht. Aufgabe des Controllers ist es diese Komponenten zusammen zu betreiben, synchronisieren und zu überwachen.

### 2.3.3 SPS-Programme

Speicherprogrammierbare Steuerungen finden in der Steuerung von in der Industrie betriebenen Fertigungsmaschinen eine weite Verbreitung, sie ermöglichen es viele Sensoren zu überwachen und die Maschine entsprechend zu steuern. Im Minimalfall enthält eine SPS-Steuerung Eingänge, Ausgänge, einen Mikrocontroller und Speicher. Für die Steuerungen werden Programme geschrieben und integriert, welche die über die Eingänge ankommenden Informationen verarbeiten und entsprechende Aktionen an die Ausgänge weitergeben. Im Untersuchungsgegenstand, ist eine Software simulierte SPS integriert und dazugehörend eine Vielzahl von SPS-Programmen die den Controller in der Verarbeitung verschiedenster Sensoren unterstützen und die Ergebnisse und Informationen an den Controller weiter geben. Hierdurch ist es möglich, den Controller die Maschinen zu erweitern und

um die beim Kunden geforderten Erweiterungen anzupassen. Dabei kann es sich um Erweiterungen wie die Überwachung einer automatischen Bestückung, Überwachung zusätzlicher Sicherheitseinrichtungen in oder um die Maschinen oder die Ansteuerung zusätzlicher Komponenten für die fertigungsintegrierte Werkstückprüfung handeln.

### 2.3.4 Makros

Zur Software des Controller im Untersuchungsgegenstand gehören eine Vielzahl von Makros, welche als Erweiterungen bzw. zusätzliche Optionen zur Software des Controller gehören und welche mit in die Versionsverwaltung integriert werden müssen. Die Makros sind vom Controller ausführbare Dateien, welche Funktionen die im Controller enthalten sind zu bestimmten Abläufen zusammenfassen oder auch weitere Funktionen enthalten um den Funktionsumfang des Controllers zu erweitern.

## 2.4 Besonderheiten der Industriellen Fertigung

### 2.4.1 Rückführbarkeit der Versionen

### 2.4.2 Anforderungen an die Wartbarkeit der Systeme

### 2.4.3 Anforderung an die Wartbarkeit der Systeme

### 2.4.4 Reaktionszeiten im Fehlerfall

## 2.5 Anforderungsmanagement

Das Anforderungsmanagement, ist eine Management Disziplin die das Software Engineering unterstützt. Es setzt sich damit auseinander eine optimale Lösung durch strukturierte und organisierte Planung und Definition der Anforderung zu erreichen.

Durch das Anforderungsmanagement soll vermieden werden, dass durch schlechte, unzureichende oder missverständliche Definitionen, eine Entwicklung am Ende nicht dem Entspricht, was der Auftraggeber haben wollte.

Das Anforderungsmanagement lässt sich in unterschiedliche Arbeitspakete unterteilen, hierzu lassen sich in der Literatur die unterschiedliche Ausprägungen und Anzahlen finden. Bei Marcus Grande[1] wird in vier Haupttätigkeiten unterteilt. Für die Diplomarbeit habe ich mir beispielhaft die Punkte ausgesucht, wie sie im Buch Änforderungsmanagement in sieben Tagen"[2] zu finden sind. Es gibt

### 2.5.1 Arbeitspakete im Anforderungsmanagement

- Anforderungsmanagement planen
  Das Anforderungsmanagement dient der Planung und so ist es auch nicht weiter verwunderlich, dass auch in der Vorbereitung eine Planung für das Anforderungsmanagement sinnvoll ist. Es stehen eine Vielzahl von Tools, welche das Anforderungsmanagement unterstützen, die Art und weise der Dokumentation sollte festgelegt werden und die Anforderungsattribute sollten festgelegt werden. Bei den Attributen muss auf die fünf Pflichtattribute für Anforderungen geachtet werden.[3]

[1] Marcus Grande. *100 Minuten für Anforderungsmanagement : Kompaktes Wissen nicht nur für Projektleiter und Entwickler*. 2., aktualisierte Aufl. 2014. Springer Vieweg, 2014, S. 10.
[2] Thomas Niebisch. *Anforderungsmanagement in sieben Tagen*. Springer-Verlag, 2013, S. 30-31.
[3] Grande, *100 Minuten für Anforderungsmanagement : Kompaktes Wissen nicht nur für Projektleiter und Entwickler*, S. 40.

- Ausgangspunkt finden
  *Um das Ziel zu finden, sollte man wissen wo man los geht*
  Bevor man mit der Zielfindung beginnen kann, sollte man wissen, wie der aktuelle Stand ist. Man kann das mit einem Orientierungsmarsch bei der Bundeswehr vergleichen, bei dem Soldaten mit einem Kompass und einer Geländekarte ein Ziel erreichen müssen, wenn sie nicht wissen wo sie los gehen, ist es unwahrscheinlich, dass sie das ausgegebene Ziel erreichen. Wie schaut der zu bearbeitende Prozess aus und wer ist darin Akteur und/oder Betroffener. Durch das ermitteln des IST-Zustand kann man seine Stakeholder ermitteln, lernt den Prozess im aktuellen Zustand, seine Grenzen und die von außen einwirkenden Elemente kennen.

- Anforderungen erheben
  Für die Erhebung von Anforderungen existieren verschiedene Methoden und Quellen.

  **Quellen** für Anforderungen sind

  - Stakeholder, die Aufgrund ihrer Mitarbeit am Prozess oder durch Auswirkungen und Ergebnissen dieses, betroffen sind.
  - Prozessmodelle, in denen der Ablauf des Prozesses dokumentiert ist
  - Beobachtungen des Prozessalltags
  - Prozessbeschreibungen oder Problemberichte

  **Methoden** zur Erhebung von Anforderungen sind

  - Interviews mit Stakeholdern oder Verantwortlichen
  - Workshops mit Stakeholdern

- Analysen von Berichten und Beschreibungen

Je nach zur Verfügung stehender Quelle ist eine dazu passende Methode zur Erhebung dieser anzuwenden.

- Anforderungen dokumentieren
  Mit dem Erheben von Anforderungen wird auch das dokumentieren dieser zu einem Thema. Wenn die Disziplin des Anforderungsmanagement in einem größeren Unternehmen schon weit fortgeschritten ist wird es für die Dokumentation evtl. schon geeignete Tools im Unternehmen geben. Bei kleinen Firmen wird dies in der Regel wohl nicht der Fall sein. Für die Anforderungsdokumentation stehen verschiedene Methoden, Darstellungsformen und Tools zur Verfügung. Diese dienen zum einen der besseren Visualisierung der Anforderungen, evtl. der Gestaltung beispielhafter Oberflächen als Entwurfsmuster können aber im Falle einer rein Verbalen Funktionsbeschreibung auch in Form einer Tabelle sein. Wichtig bei der Dokumentation ist immer, dass man einen Mix aus den zur Verfügung stehenden Methoden wählt um alle Aspekte, die durch die Anforderungen erhoben werden auch passend und widerspruchsfrei abbilden zu können.[4] Es ist darauf zu achten, dass man sich mit dem gewählten gut auskennt um das gesamte Potenzial ausschöpfen zu können und bei der Gesaltung darauf achten das auch die beteiligten Stakeholder das Ergebnis verstehen.

- Anforderungen qualitätssichern Nachdem alle Anforderungen zusammengetragen wurden, müssen diese vom Anforderungsmanager auf Integrität, Vollständigkeit und Widersprüchlichkeit überprüft werden. In der Regel ist eine Iteration der vorangegangen Schritte notwendig um Lücken zu schließen und Widersprüchlichkeiten auszumerzen. Hierzu sind Interviews und erneute Workshops ein probates mittel. Am Ende muss eine Lücken freie Prozessbeschreibung durch die Anforderungen vorliegen und alle Stakeholder diese Anforderungen unterschreiben.

[4] Niebisch, *Anforderungsmanagement in sieben Tagen*, S. 93-137.

- Anforderungen verwalten Im Buch 100 Minuten für Anforderungsmanagement[5] wird dieser Punkt als Pflege und Verwaltung von Anforderungen bezeichnet, was auch zutreffender ist. Denn es handelt sich hierbei nicht nur um ein reines Verwalten der Anforderungen.

  Anforderungen können sich innerhalb eines Projektverlaufs ändern, es können Anforderungen hinzu kommen oder auch weg fallen. Hierbei ist es immer notwendig, dass jede Veränderung dokumentiert wird und die Anforderungen Versioniert sind, dadurch hat man bei Terminverschiebungen etwas in der Hand, da durch verändern oder erweitern der Anforderungen auch immer eine Verschiebung der Termine möglich sein kann. Neben dem Namen des ändernden, einem Datum und einer Versionsnummer sollte auch immer eine Erklärung, was geändert wurde, mit erfasst werden. Stakeholder sollten Zugriff auf die verwalteten Dokumente haben um jederzeit noch einmal nachlesen zu können, wie die Anforderungen getroffen wurde. Wenn es in der Entwicklungsphase zu viele Änderungen in den Anforderungen gibt, läuft man Gefahr, dass das Projekt zu sehr in die Länge gezogen wird und im Extremfall zu scheitern droht. Es ist darauf zu achten, dass die Änderungen mit der Zeit abnehmen und ab einem gewissen Zeitpunkt stabil sind und keine zusätzlichen Forderungen oder Änderungen mehr hinzu kommen.

## 2.6 Anforderungen

Anforderungen unterliegen verschiedenen Attribute, Arten und bedürfen bestimmter Formulierungen, damit sowohl Anwender als auch Entwickler unmissverständlich das gleiche aus ihr ableiten können.

[5] Grande, *100 Minuten für Anforderungsmanagement : Kompaktes Wissen nicht nur für Projektleiter und Entwickler*, S. 103.

### 2.6.1 Definition der Anforderung

In der Literatur finden sich die unterschiedlichsten Definitionen für Anforderungen und so kommt auch eine Analyse des KIT zur Schlussfolgerung, dass keine von ihnen als allgemein gültig betrachtet werden kann.[6] Es lässt sich dabei festhalten, dass eine Anforderung immer etwas definiert, was man von einem Produkt erwartet und diese vorzugsweise in irgendeiner schriftlichen Beschreibung festgehalten und dokumentiert werden sollte.

### 2.6.2 Arten der Anforderungen

Anforderungen lassen sich in viele unterschiedliche Arten unterteilen. Das hängt auch mit dem jeweiligen Projekt zusammen. So gibt es Arten von Anforderungen die immer auftreten und andere, die vorkommen können oder auch nicht. Bei der Unterteilung der Anforderungen ist man sich in der Literatur auch nicht immer einig. Während die einen als grobe Trennung in Anwenderanforderungen und Systemanforderungen unterteilen[7] wählen die anderen Funktionale-Anforderungen und Nicht-funktionale Anforderungen[8].[9] es werden die Anforderungen bei beiden Varianten noch in weitere Unterkategorien unterteilt.

- Funktionale- & Nicht-Funktionale Anforderungen
  Bei der Unterteilung in Funktionale und Nicht-funktionale Anforderungen werden alle Anforderungen, die direkt die Funktion und ihre Auswirkung beschreiben in Gruppe der Funktionalen Anforderungen einsortiert und alle

[6] 9 Christina Zehnter, Alexander Burger und Jivka Ovtcharova. *Key-Performance-Analyse von Methoden des Anforderungsmanagements*. Techn. Ber. Karlsruhe, 2012. URL: `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000028693`, S. 2.3.1.

[7] Gerhard [Hrsg.] Versteegen und Alexander Heßeler, Hrsg. *Anforderungsmanagement : formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl; [mit Praxisbeiträgen von BearingPoint (KPMG), HOOD Group und Telelogic; mit 11 Tab.]* Xpert.press. Berlin: Springer, 2004.

[8] Grande, *100 Minuten für Anforderungsmanagement : Kompaktes Wissen nicht nur für Projektleiter und Entwickler*.

[9] Niebisch, *Anforderungsmanagement in sieben Tagen*.

anderen Anforderungen in die Kategorie der Nicht-funktionalen Anforderungen. Diese werden dann noch in weitere Kategorien unterteilt.

- Anwender- & Systemanforderungen
  Bei der Gliederung in Anwenderanforderungen und Systemanforderungen, werden in die Kategorie der Anwenderanforderung alle einsortiert die beschreiben wie eine Software für den jeweiligen Anwender bedienbar sein sollte, sie beschreiben ein konkretes Problem,[10] Systemanforderungen beschreiben wie die Lösung eines Problems umgesetzt werden soll, was es an Schnittstellen zu anderen Prozessen zu gibt. Anwenderanforderungen werde in der Regel durch die späteren Nutzer erhoben. Unterschiedliche Nutzergruppen, werden auch unterschiedliche Anforderungen hervor bringen, da sie den Prozess aus ihrer eigenen für sie erforderlichen Sicht betrachten und entsprechend die Anforderungen erheben. Systemanforderungen werden dagegen eher von den Entwicklern erhoben, die das System als ganzen betrachten müssen, was sind an Schnittstellen zu anderen Prozessen gegeben, wie können bestimmte Anforderungen umgesetzt werden, dass vom gewünschten Input auch der vom Anwender geforderte Output erzeugt wird. Die Systemanforderungen Teil man dann noch in Funktionale und nicht-funktionale Anforderungen auf. Bei den Funktionalen Anforderungen werden dann die Anforderungen eingegliedert, welche die zu realisierenden Funktionen des System beschreiben (z.B. Jeder Nutzer muss zu Beginn seinen Benutzernamen und Passwort für den LogIn zum System eingeben). Zu den nicht Funktionalen Anforderungen werden die zugeordnet, welche zwar für den Betrieb des Systems erforderlich sind aber im Untergrund laufen und dem Benutzer verborgen bleiben (z.B. Nach einer Eingabe von Benutzername und Passwort muss überprüft werden, ob diese im System hinterlegt sind)

Es ist für den Erfolg eines Projektes nicht ausschlaggebend, an welche Art der Unterteilung man sich hält. Wichtig dabei ist, dass man seine Anforderungen in

[10] Versteegen und Heßeler, *Anforderungsmanagement : formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl; [mit Praxisbeiträgen von BearingPoint (KPMG), HOOD Group und Telelogic; mit 11 Tab.]*

irgendeiner Art und weise unterteilt und so die in großer Stückzahl auftretenden Anforderungen strukturiert.

### 2.6.3 Attribute von Anforderungen

### 2.6.4 Methoden zum Erheben von Anforderungen

### 2.6.5 •

# 3 Literaturverzeichnis

Grande, Marcus. *100 Minuten für Anforderungsmanagement : Kompaktes Wissen nicht nur für Projektleiter und Entwickler*. 2., aktualisierte Aufl. 2014. Springer Vieweg, 2014.

Niebisch, Thomas. *Anforderungsmanagement in sieben Tagen*. Springer-Verlag, 2013.

Versteegen, Gerhard [Hrsg.] und Alexander Heßeler, Hrsg. *Anforderungsmanagement : formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl; [mit Praxisbeiträgen von BearingPoint (KPMG), HOOD Group und Telelogic; mit 11 Tab.]* Xpert.press. Berlin: Springer, 2004.

Zehnter, Christina, Alexander Burger und Jivka Ovtcharova. *Key-Performance-Analyse von Methoden des Anforderungsmanagements*. Techn. Ber. Karlsruhe, 2012. URL: `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000028693`.

# Eidesstattliche Erklärung

Studierender: Maik Ledwina
Matrikelnummer: 880767

Hiermit erkläre ich, dass ich diese Arbeit selbstständig abgefasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

..... .....

Ort, Abgabedatum Unterschrift (Vor- und Zuname)