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	Text
SPS	Sepicher Programmierbare Steuerung

Bezeichnungen und Begrifflichkeiten

Repository	Text
Mergen	text
Push	text
Klonen	text
Open-Source	Text

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 beiden 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.

$Teil berdie Controller$

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 Tachos. An jeder Messmaschine werden Messköpfe zur Aufnahme von Tastpunkten angebracht. Aufgabe des Controllers ist es diese Komponenten zusammen zu betreiben, synchronisieren und zu überwachen.

$Teil berSPSProgramme$

2.3.3 SPS-Programme

$Teil berdie Grundlagen von Macros$

2.3.4 Macros

Zur Software des Controller im Untersuchungsgegenstand gehören eine Vielzahl von Macros, welche als Erweiterungen bzw. zusätzliche Optionen zur Software des Controller gehören und welche mit in die Versionierung integriert werden müssen. Die Macros 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

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)