C:\Users\haller\Desktop\Logo_HFU.tif

Bachelor-Thesis

in

Wirtschaftsinformatik

Verteilte Versionsverwaltung

Eine Tool-Integration in Java-Programmier-Vorlesungen

|  |  |
| --- | --- |
| Referent: | Prof. Dipl.-Inform. J. Anton Illik |
| Korreferent: | Prof. Dr.-Ing. Stefan Noll |
| Vorgelegt am: | 31.08.2017 |
| Vorgelegt von: | Ertugrul Özkara |
|  | 245269 |
|  | Steigstraße 54  78554 Aldingen |
|  | ertugrul.oezkara@hs-furtwangen.de |

# Abstract

[Englisch, 100 -120 Worte]

[Deutsch, 100 – 120 Worte]

# Inhaltsverzeichnis

[Abstract III](#_Toc481708990)

[Inhaltsverzeichnis V](#_Toc481708991)

[Abbildungsverzeichnis IX](#_Toc481708992)

[Tabellenverzeichnis XI](#_Toc481708993)

[Abkürzungsverzeichnis XIII](#_Toc481708994)

[1 Einleitung 15](#_Toc481708995)

[2 Grundlagen 16](#_Toc481708996)

[2.1 Definition 16](#_Toc481708997)

[2.2 Was ist Versionsverwaltung 17](#_Toc481708998)

[2.3 Vorteile 18](#_Toc481708999)

[2.4 Historie 20](#_Toc481709000)

[2.4.1 SCCS 20](#_Toc481709001)

[2.4.2 CVS 20](#_Toc481709002)

[2.4.3 SVN 20](#_Toc481709003)

[2.5 Funktionsweise 20](#_Toc481709004)

[2.5.1 Probleme(nicht sicher) 20](#_Toc481709005)

[2.5.2 Lösungswege(nicht sicher) 20](#_Toc481709006)

[2.6 Arten 20](#_Toc481709007)

[2.6.1 Lokale Versionsverwaltung 20](#_Toc481709008)

[2.6.2 Zentrale Versionsverwaltung 20](#_Toc481709009)

[2.6.3 Lokal versus zentral 20](#_Toc481709010)

[2.6.4 Verteilte Versionsverwaltung 20](#_Toc481709011)

[2.7 Konzepte 20](#_Toc481709012)

[2.7.1 Lock Modify Write 20](#_Toc481709013)

[2.7.2 Copy Modify Merge 20](#_Toc481709014)

[2.7.3 Objektbasierte Versionisierung 20](#_Toc481709015)

[2.8 Strategien 20](#_Toc481709016)

[3 Git 21](#_Toc481709017)

[3.1 Entstehung 21](#_Toc481709018)

[3.2 Grundlagen 21](#_Toc481709019)

[3.3 Eigenschaften 21](#_Toc481709020)

[3.4 Interner Aufbau 21](#_Toc481709021)

[3.5 Funktionsweise 21](#_Toc481709022)

[3.6 Vorteile 21](#_Toc481709023)

[3.7 Wichtige Begriffe innerhalb der Systeme 21](#_Toc481709024)

[3.8 Branches kurz erklärt(nicht sicher ob hier oder in 3.7) 21](#_Toc481709025)

[3.9 Branching und Merging(nicht sicher ob hier oder in 3.7) 21](#_Toc481709026)

[3.10 Protokollierung 21](#_Toc481709027)

[3.11 Wiederherstellung 21](#_Toc481709028)

[3.12 Archivierung 21](#_Toc481709029)

[3.13 Koordinierung 21](#_Toc481709030)

[3.14 Gleichzeitiges Entwickeln 21](#_Toc481709031)

[3.15 Verteilte Abläufe 21](#_Toc481709032)

[3.16 Hosting über GitHub 21](#_Toc481709033)

[3.17 Befehle 21](#_Toc481709034)

[3.18 Kommandozeile zur Bedienung von Git(nicht sicher) 21](#_Toc481709035)

[4 Versionsverwaltungssysteme 23](#_Toc481709036)

[4.1 Ziele 23](#_Toc481709037)

[4.2 Verschiedene Versionsverwaltungssysteme im Überblick 23](#_Toc481709038)

[4.2.1 Darcs 23](#_Toc481709039)

[4.2.2 Git 23](#_Toc481709040)

[4.2.3 GNU arch 23](#_Toc481709041)

[4.2.4 Apache Subversion 23](#_Toc481709042)

[4.2.5 Fossil 23](#_Toc481709043)

[4.2.6 Mercurial 23](#_Toc481709044)

[4.2.7 Bazaar 23](#_Toc481709045)

[4.2.8 Gegenüberstellung verschiedener Systeme 23](#_Toc481709046)

[5 Umsetzung/Integration des Tools in die Vorlesung 23](#_Toc481709047)

[5.1 Git installieren 23](#_Toc481709048)

[5.2 GitHub 23](#_Toc481709049)

[5.3 Repository aufbauen 23](#_Toc481709050)

[5.4 Veränderung an Repository melden 23](#_Toc481709051)

[5.5 Commit-Verlauf einsehen 23](#_Toc481709052)

[5.6 Rückgängig machen 23](#_Toc481709053)

[5.7 Wartung und Wiederherstellung 23](#_Toc481709054)

[5.8 Git als PlugIn in Eclipse(nicht sicher) 23](#_Toc481709055)

[6 Resumee 23](#_Toc481709056)

[Literaturverzeichnis 25](#_Toc481709057)

[Stichwortverzeichnis 27](#_Toc481709058)

[Eidesstattliche Erklärung 29](#_Toc481709059)

[A. [Anhang] 31](#_Toc481709060)

[7 Literaturverzeichnis 31](#_Toc481709061)

# Abbildungsverzeichnis

Das Abbildungsverzeichnis muss vor der Finalen Abgabe entfernt werden, wenn keine Abbildungen in die Arbeit eingefügt wurden.

# Tabellenverzeichnis

Das Tabellenverzeichnis muss vor der Finalen Abgabe entfernt werden, wenn keine Tabellen in die Arbeit eingefügt wurden.

# Abkürzungsverzeichnis

|  |  |
| --- | --- |
| VCS | Version Control System |
| SCCS |  |
| CVS |  |
| SVN |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Einleitung

Versionsverwaltungen oder auch Versionskontrollsysteme (kurz: VCS) verwalten die Änderungen an beliebig vielen Dateien, die im Laufe der Zeit entstehen. Sie ermöglichen es diese in eine ältere Version zurückzuführen ohne dass es dabei eine Rolle spielt, ob es sich um Bilddateien, Quellcode oder jegliche andere Art von Dateien handelt.[[1]](#footnote-1)

Versionskontrolle ist insbesondere im Bereich der Software-Entwicklung ein immer wichtiger werdender Begriff. Höchstwahrscheinlich erreicht jedes Projekt irgendwann den Punkt, an dem ältere oder gelöschte Stände noch einmal eingesehen werden müssen, es zu verstehen gilt, warum zu einem bestimmten Zeitpunkt eine bestimmte Funktion in ein Programm eingefügt worden ist oder das etwas kaputt ist und man in die Ausgangssituation zurück möchte.[[2]](#footnote-2) Darüber hinaus besteht jederzeit die Gefahr, dass man vor allem in großen Projekten mit mehreren Mitgliedern, die zuvor verrichtete Arbeit eines anderen überschreibt und diese damit verloren geht.

Mit immer größer und anspruchsvoller werdenden Programmen steigt auch die Anzahl der benötigten Entwickler. Diese wiederum haben unter anderem die Aufgabe Fehler so gut wie möglich zu verhindern. Dies jedoch stellt eine immer größer werdende Herausforderung dar. Abhilfe schaffen hier Versionsverwaltungssysteme, welche sich deshalb auch in dieser Branche am schnellsten verbreitet haben.[[3]](#footnote-3) Diese bieten nicht nur einen Ausweg für die beschriebenen Probleme, sondern sind ebenso eine Erleichterung im Voranschreiten des Entwicklungsprozesses in Solo- und auch in Großprojekten.[[4]](#footnote-4)

Software-Entwicklung findet nicht nur in Unternehmen statt, sondern wird auch in den verschiedensten Einrichtungen gelehrt. Dabei reicht das Spektrum von weiterführenden berufsbezogenen Schulen, bis hin zu Universitäten. Daher ist es nicht unwahrscheinlich, dass die oben aufgeführten Probleme sich auch in Unterrichtsräumen und Hörsälen zu erkennen geben und Lehrende sowie auch Lernende vor Herausforderungen stellen.

Aufgrund dieser Erkenntnis ist es das Ziel dieser Arbeit Versionsverwaltung Studenten, Schülern sowie Lehrkörpern näher zu bringen, diese dafür zu begeistern und die benötigte Versionskontrolle in Programmier-Vorlesungen zu verankern.

Gegenstand von Kapitel 2 ist es dem Leser fundamentales Wissen über Versionsverwaltung zu vermitteln und zu verstehen aus welcher Notwendigkeit heraus diese entstanden ist. Des Weiteren werden verschiedene Konzepte und auch Funktionsweisen angesprochen. Nachfolgend werden verschiedene und gängige Versionsverwaltungssysteme unter die Lupe genommen und entsprechende wichtige Begriffe erklärt, um sich ein Bild darüber machen zu können, welche Systeme am geeignetsten für die Arbeit in Programmier-Vorlesung sind. Anschließend wird auf die Versionsverwaltungssoftware Git eingegangen Den Abschluss bildet die Integration von Git in die Vorlesung und die zugehörige Handhabung.

# Grundlagen

Dieses Kapitel geht auf wichtiges und fundamentales Wissen im Bereich von Versionsverwaltung und auch verteilte Versionsverwaltung ein. Es wird erklärt was Versionsverwaltung ist, warum es Notwendig ist, welche Konzepte dahinter stecken und warum heutzutage verteilte Versionsverwaltung bevorzugt wird.

## Definition

„Eine Versionsverwaltung hat die Aufgabe, den Werdegang eines Projektes zu protokollieren. Sie übernimmt die Versionierung [sic] und die Archivierung der Daten und ermöglicht den gemeinsamen Zugriff darauf. Dabei kann jede Änderung und jeder Stand verfolgt, rückgängig gemacht oder wiederhergestellt werden.“ (Versionsverwaltung 2016, S. 1)

## Was ist Versionsverwaltung

„Arbeit, die Sie nicht mit einem Versionskontrollsystem verwalten, ist Arbeit, die Sie möglicherweise noch einmal machen müssen - sei es weil Sie versehentlich eine Datei löschen oder Teile als obsolet betrachten, die Sie später doch wieder benötigen.“ (Haenel/Plenz 2011, S. 14)

Das obige Zitat unterstreicht das Argument, wie wichtig die Arbeit mit Versionsverwaltung in manchen Bereichen sein kann. Versionsverwaltung kann unter anderem dafür sorgen, dass bei einer sorgfältigen Planung viel Zeit gespart wird. Beispielsweise können verlorengeglaubte Dateien wiederhergestellt werden. Dadurch muss bei einem unterlaufenen Fehler, die bereits getane Arbeit nicht wiederholt werden.

Versionsverwaltung bedeutet im Prinzip nichts anderes, als das von Dateien Versionen erstellt werden können, um diese dann je nach Gebrauch zu verwalten. Jede noch so kleine Änderung an einer Datei erstellt eine neue Version dieser. Der Begriff »Version« kann vieles Bedeuten, doch wenn man im Bereich der Versionskontrolle davon spricht, handelt es sich um die Version einer einzelnen Datei oder der Ansammlung mehrerer Dateien.[[5]](#footnote-5)

Dadurch soll das gemeinsame Arbeiten an einer Vielzahl von Dokumenten ermöglicht werden. Dabei gilt die höchste Priorität der Vermeidung von Datenverlust bei gleichzeitiger Entwicklung durch mehrere Personen. [[6]](#footnote-6)

Egal ob eine reine Textdatei bearbeitet wird, ein Bild oder ein Video, es ist üblich, dass der aktuelle Stand immer wieder gesichert wird, damit die bereits verrichtete Arbeit nicht verloren geht oder damit man bei Bedarf auf eine ältere Version zurückgreifen kann. Die Vorgehensweise bei der Benennung dieser "Sicherheitskopien" ist von Mensch zu Mensch unterschiedlich. Manche geben die aktuelle Versionsnummer dem Dateinamen bei, andere wiederum erzeugen verschiedene Ordner mit dem entsprechenden Datum.  
Diese Vorgehensweisen sind zwar nicht falsch, doch weder sind sie praktisch noch sind sie resistent gegen Fehleranfälligkeit. Wenn zu viele Dateien versioniert werden müssen und diese keine eindeutigen Namen zugewiesen bekommen, dann steht man vor einem Problem.[[7]](#footnote-7)

Zur Lösung dieser Probleme wurden Versionsverwaltungssysteme entwickelt, die es erlauben, genau nachzusehen wer etwas wann und weshalb geändert hat, um bei Bedarf die entsprechenden Änderungen rückgängig zu machen. Das System notiert sich nämlich, wer eine Änderung an einer Datei vorgenommen hat mit der dazugehörigen Uhrzeit und verweist auf den Abschnitt in der Datei, die geändert worden ist. So entsteht nach und nach mit jeder Änderung eine Änderungshistorie, die dem Arbeiten an mehreren Dateien Erleichterung bringt. Diese Änderungshistorie erlaubt es genau nachzuverfolgen ab welchem Punkt ein Projekt in die falsche Richtung läuft oder Fehler aufweist, sodass diese Fehler schnell ausfindig gemacht und auch behoben werden können.[[8]](#footnote-8)

## Vorteile

Vor allem bei großen Projekten ist die Nutzung eines Versionsverwaltungssystems fast unumgänglich. Diese bringen nämlich erhebliche Erleichterungen mit an Bord. Die meisten Vorteile die eine Versionskontrolle mit sich bringt wurden zwar weiter oben schon fast alle genannt, dennoch werden hier noch einmal einige wichtige Punkte zusammengefasst.

Wie schon erwähnt, können Änderungen lokalisiert und rückgängig gemacht werden. Der große Vorteil hierbei ist, dass diese Änderungen innerhalb des gesamten Projekts vorgenommen werden können. Durch Fehler wird nicht das endgültige Schicksal eines Projektes beschlossen. Das Wissen, dass man jederzeit zu einem bestimmten Zeitpunkt zurückspringen kann, verleiht Mitarbeitern innerhalb eines Projektes eine gewisse Ruhe. Das Team arbeitet dadurch ungezwungener und schneller, denn die Möglichkeit Fehler schnell wieder beheben zu können, verleiht das Gefühl von Sicherheit. Versionsverwaltungssysteme verhalten sich in diesem Rahmen so ähnlich wie eine Zeitmaschine, die es uns erlaubt an einem bestimmten Datum zu einem bestimmten Punkt des Projektes zurückzuspringen.[[9]](#footnote-9)

Ein weiterer Vorteil, den Versionskontrolle mit sich bringt, ist das gleichzeitige Entwickeln. Mehrere Entwickler können gleichzeitig und kontrolliert an demselben Quellcode arbeiten ohne dass die Arbeit des jeweils anderen dadurch überschrieben wird.[[10]](#footnote-10)

Des Weiteren hilft die Protokollierung dabei gemachte Änderungen ausfindig zu machen und zu verstehen wer im Team welche Änderung aus welchem Grund und an welcher Stelle des zu bearbeitenden Programms gemacht hat.[[11]](#footnote-11) Dies ist unter anderem auch dann nützlich, falls man ein bestimmtes Fragment des selbst erzeugten Quellcodes nicht mehr nachvollziehen kann.

Nicht nur gleichzeitiges Entwickeln ist möglich sondern auch das gleichzeitige Veröffentlichen von Software. Das Team ist dadurch nicht gezwungen kurz vor einem Release seine Arbeiten zu unterbrechen und kann ungehindert auf einem anderen Entwicklungszweig des Projektes die Arbeit fortsetzen.[[12]](#footnote-12)

## Historie

### SCCS

### CVS

### SVN

## Funktionsweise

### Probleme(nicht sicher)

### Lösungswege(nicht sicher)

## Arten

### Lokale Versionsverwaltung

### Zentrale Versionsverwaltung

### Lokal versus zentral

### Verteilte Versionsverwaltung

## Konzepte

### Lock Modify Write

### Copy Modify Merge

### Objektbasierte Versionisierung

## Strategien

# Git

## Entstehung

## Grundlagen

## Eigenschaften

## Interner Aufbau

## Funktionsweise

## Vorteile

## Wichtige Begriffe innerhalb der Systeme

## Branches kurz erklärt(nicht sicher ob hier oder in 3.7)

## Branching und Merging(nicht sicher ob hier oder in 3.7)

## Protokollierung

## Wiederherstellung

## Archivierung

## Koordinierung

## Gleichzeitiges Entwickeln

## Verteilte Abläufe

## Hosting über GitHub

## Befehle

## Kommandozeile zur Bedienung von Git(nicht sicher)

# Versionsverwaltungssysteme

## Ziele

## Verschiedene Versionsverwaltungssysteme im Überblick

### Darcs

### Git

### GNU arch

### Apache Subversion

### Fossil

### Mercurial

### Bazaar

### Gegenüberstellung verschiedener Systeme

# Umsetzung/Integration des Tools in die Vorlesung

## Git installieren

## GitHub

## Repository aufbauen

## Veränderung an Repository melden

## Commit-Verlauf einsehen

## Rückgängig machen

## Wartung und Wiederherstellung

## Git als PlugIn in Eclipse(nicht sicher)

# Resumee

# Literaturverzeichnis

Fügen Sie hier ihre verwendete Literatur ein. Beachten Sie dabei die Vorgaben zu den Zitierstilen

# Stichwortverzeichnis

Das Stichwortverzeichnis ist optional. Wenn Sie kein Stichwortverzeichnis in Ihrer Arbeit verwenden wollen können Sie dieses Kapitel entfernen!

# Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Ort, Datum Name]

Auch hier müssen die Platzhalter mit den korrekten Daten ersetzt werden.

# [Anhang]

Der Anhang ist optional. Wenn Sie keinen Anhang in Ihrer Arbeit verwenden wollen können Sie dieses Kapitel entfernen!

1. Vgl. Denker, Merlin/Stefan Srecec, »Versionsverwaltung mit Git. Fortgeschrittene Programmierkonzepte in Java, Haskell und Prolog« (2015), S. 1 [↑](#footnote-ref-1)
2. Vgl. Vijayakumaran, Sujeevan, *Versionsverwaltung mit Git*, 1. Auflage, Frechen 2016, S. 14 [↑](#footnote-ref-2)
3. Vgl. Bechara, John, »Revisionssichere Archivierung. Verteiltes Dokumentenmanagement« (2009), S. 5 [↑](#footnote-ref-3)
4. Vgl. Vijayakumaran, *Versionsverwaltung mit Git* [wie Anm. 2], S. 14 [↑](#footnote-ref-4)
5. Vgl. ebd., S. 17 [↑](#footnote-ref-5)
6. Vgl. Taentzer, »Versionsverwaltung von Softwareartefakten« (2014), S. 43–74, hier: S. 43 [↑](#footnote-ref-6)
7. Vgl. Vijayakumaran, *Versionsverwaltung mit Git* [wie Anm. 2], S. 17 [↑](#footnote-ref-7)
8. Vgl. ebd., S. 17–18 [↑](#footnote-ref-8)
9. Vgl. Thomas, David/Andrew Hunt/Falk Lehmann/Uwe Petschke, »Der pragmatische Programmierer«, *Der pragmatische Programmierer, Hunt, Andrew, [2. Aufl.]. - München [u.a.] Hanser* (2003), S. 1–7, hier: S. 1–2 [↑](#footnote-ref-9)
10. Vgl. ebd., S. 1 [↑](#footnote-ref-10)
11. Vgl. ebd., S. 1–2 [↑](#footnote-ref-11)
12. Vgl. ebd., S. 2 [↑](#footnote-ref-12)