TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Wirtschaftswissenschaften

Professur für Wirtschaftsinformatik,   
insb. Informationssysteme in Industrie und Handel

Das ist das Thema der Arbeit

Bachelor/Master/Diplomarbeit

zur Erlangung des akademischen Grades

„Bachelor/Master of Science/DipomXYZ“

|  |  |
| --- | --- |
| Name: | Musterfrau, Erika |
| Adresse: | Musterstraße, 01234 Musterstadt |
| Matrikelnummer: | 1234567 |
| Übermittelt an: | Prof. Dr. Muster |
| Übermittlungsdatum: | 17.11.2018 |

Inhaltsverzeichnis

[Abbildungsverzeichnis II](#_Toc516228888)

[Tabellenverzeichnis III](#_Toc516228889)

[Formelverzeichnis III](#_Toc516228890)

[Abkürzungsverzeichnis III](#_Toc516228891)

[1 Einleitung (Überschrift 1) 3](#_Toc516228892)

[2 Verwendung der Vorlage 3](#_Toc516228893)

[2.1 Überschriften (Überschrift 2) 3](#_Toc516228894)

[2.2 Standardtext und Hervorhebungen 3](#_Toc516228895)

[2.3 Aufzählungen und Nummerierungen 3](#_Toc516228896)

[2.4 Abbildungen, Tabellen und Formeln 3](#_Toc516228897)

[2.4.1 Abbildungen (Überschrift 3) 3](#_Toc516228898)

[2.4.2 Tabellen 3](#_Toc516228899)

[2.4.3 Formeln 3](#_Toc516228900)

[2.5 Fußnoten 3](#_Toc516228901)

[2.6 Kopfzeile 3](#_Toc516228902)

[2.7 Abkürzungen 3](#_Toc516228903)

[2.8 Literaturverweise 3](#_Toc516228904)

[2.9 Verzeichnisse 3](#_Toc516228905)

[2.10 Eidesstattliche Erklärung 3](#_Toc516228906)

[Literaturverzeichnis 3](#_Toc516228907)

[Anhangsverzeichnis III](#_Toc516228908)

[Anhang III](#_Toc516228909)

###### Abbildungsverzeichnis

[Abbildung 1: Aussagekräftige Bildunterschrift (Quelle, 1970, S. 25) 3](#_Toc516228910)

[Abbildung 2: Hinweis zum Setzen der korrekten Seitenzahl im Anhang 3](#_Toc516228911)

###### Tabellenverzeichnis

[Tabelle 1: Aussagekräftige Tabellenbeschriftung 3](#_Toc516228912)

[Tabelle 2: Einsatz arabischer und römischer Seitenzahlen 3](#_Toc516228913)

###### Formelverzeichnis

[Formel 1: Biomialformel 3](#_Toc516228914)

###### Abkürzungsverzeichnis

|  |  |
| --- | --- |
| Abk. | Abkürzung |
|  |  |

# Einleitung

Ziel ist es, einen Marktüberblick der verfügbaren Versionskontrollsysteme für die Softwareentwicklung darzustellen. Als Grundlage dafür soll zunächst erläutert werden, was VCS leisten können und warum sie in der modernen Softwareentwicklung eine große Bedeutung haben. Dabei sollen die wichtigsten Begriffe geklärt und eine grundlegende Systematisierung der verschiedenen Konzepte vorgestellt werden.

# Theoretische Grundlagen von Versionskontrollsystemen

Hier sollen die Grundlagen und Prinzipien dargestellt werden, auf denen die Versionskontrolle basiert, welche unterschiedlichen Konzepte existieren

## Grundlegendes Konzept von VCS

Wenn zwei oder mehr Programmierer auf die selbe Datei zugreifen wollen, kommt es unweigerlich zu einem Konflikt. Es existieren unterschiedliche Lösungen für diese Konkurrenzszenarios. Der klassische Ansatz besteht in dem *Lock-Modify-Unlock*-Prinzip, das auch als pessimistischer Ansatz bezeichnet wird. Dabei wird die relevante Ressource durch einen Vorgang gesperrt (*Check-Out*), woraufhin eine Bearbeitung der Ressource ausnahmslos durch den Vorgang möglich ist. Nach der Bearbeitung wird die Sperre entfernt, wodurch die Ressource wieder für alle Nutzer wieder verfügbar wird. Unter Verwendung dieses Prinzips ist eine Zusammenführung (*Merg*) nicht erforderlich, da eine Datei zu jedem Zeitpunkt von nur maximal einem Nutzer bearbeitet werden kann (vgl. Otte, 2009, S. 3). Für den Einsatz in VCS ist dieses Modell bis auf wenige Ausnahmen allerdings ungeeignet. Der restriktive Charakter des Modells für verteilten Softwareentwicklung ein Hindernis dar (vgl. Otte, 2009, S. 3).

Bei dem Prinzip *Copy-Modify-Merge* (optimistischer Ansatz) können mehrere Nutzer Kopien einer Ausgangsdatei erstellen und diese parallel bearbeiten, ohne dabei mit den anderen Nutzern in Kontakt stehen zu müssen. Im Anschluss an die Bearbeitung ist ein *Merge*-Prozess notwendig, der die modifizierten Dateien mit den enthaltenen Veränderungen letztendlich zu einer neuen Version der Datei zusammenführt. Durch diesen Prozess wird zum einen der administrative Aufwand im Szenario einer kollaborativen Arbeitsweise deutlich verringert, zum anderen sind deutlich flexiblere Workflows möglich als beim *Lock-Modify-Unlock*-Prinzip möglich. Aus diesem Grund basiert der Großteil der VCS auf dem *Copy-Modify-Merge-*Prinzip (vgl. Otte, 2009, S. 3). Unerlässlich ist dabei jedoch der effiziente Umgang mit Konflikten, welche beim Zusammenführen der unterschiedlichen Versionen auftreten. Daher stellen die Lösung von inhaltlichen Konflikten eine wichtige Funktion von VCS dar (vgl. Louridas, 2006, S. 105).

Die einfachste Art der Versionsverwaltung ist die lokale Versionsverwaltung. Mit Hilfe von SCCS und RCS.

In den folgenden Kapiteln werden die zwei grundlegenden Architekturen von VCS vorgestellt.

## Zentrale Versionskontrollsystemen (CVCS)

Zentrale Versionskontrollsysteme (*Central Version Control System*, CVCS) gehören zu den ersten Ansätzen einer Versionsverwaltung, die das Ziel hatte die kollaborative Zusammenarbeit zu unterstützen. Der grundlegende Aufbau entspricht einer Client-Server-Architektur. Dabei wird auf dem Server ein für alle Nutzer zugängliches Speicherverzeichnis angelegt (engl. *repository*). Dieser Server muss für das Lesen, Abrufen oder Einbringen (*Commit*) von Veränderungen (*Changes*) kontaktiert werden, wofür die entsprechenden Benutzerrechte vergeben werden müssen. Eine Veränderung wird vorgenommen, in dem entweder eine einzelne Datei oder das gesamte Repository „Ausgecheckt“ wird und damit in eine lokale Version erstellt wird. Dabei kann sowohl die jeweils aktuellste als auch jeder vorhergehende Version abgerufen werden. Das somit erstellte lokale Repository (auch: *working copy* oder Arbeitsverzeichnis) wird dann genutzt, um Veränderungen am Code vorzunehmen. Im Anschluss an die Bearbeitung, z.B. die Entwicklung eines neuen *Features*, werden die veränderten Dateien als *Commit* an das zentrale *Repository* übergeben, wodurch die Versionsnummerierung um einen Wert erhöht wird. Dabei wird der Speicherplatz des Servers geschont, indem nur die Veränderungen zwischen konsekutiven Versionen vom VCS dokumentiert werden (vgl. Louridas, 2006, S. 104). Zusammen mit der modifizierten Datei werden bei einem *Commit* verschiedene Metadaten übergeben, dazu gehört eine Kurzbeschreibung der durchgeführten Anpassung, die zu einer besseren Nachvollziehbarkeit und Transparenz beiträgt und für die Zusammenarbeit im Entwicklerteam wichtig ist.

Um nach dem Commit von Entwickler A die aktuelle Version der Datei/des Verzeichnisses zu erhalten, muss ein beliebiger Entwickler B ein Update vom zentralen Repository abrufen. Ein *Merge* ist notwendig wenn eine parallele Bearbeitung von einer Datei durch zwei (oder mehr) Entwickler stattgefunden hat. Bei kompatiblen Veränderungen am Code, können die zwei Versionen automatisch vom CVS zusammengeführt werden. Im Falle von inhaltlichen Konflikten muss eine manuelle Lösung dieser erfolgen. Zur Identifikation von Konflikten können unstrukturierte, strukturierte oder semi-strukturierte Werkzeuge verwendet werden, wobei entweder auf eine reine Textprüfung, auf eine Syntaxprüfung oder eine Kombination von beidem zurückgegriffen wird (vgl. Cavalcanti, 2017, S. 19).

## Dezentrale Versionskontrollsysteme (DVCS)

## Versionskontrollsystem Softwareanbieter

# Vergleich von „relevanten“ Versionskontrollsystemen

## Vergleichskriterien

###### Literaturverzeichnis

Das Literaturverzeichnis ist mit der Formatvorlage *Quellen* zu verwenden. Angehängt finden Sie ein beispielhaftes Literaturverzeichnis im APA 6th Standard.

Bacharach, S. B. (1989). Organizational Theories: Some Criteria for Evaluation. *Academy of Management Review*, *14*(4), 496–515. <https://doi.org/10.5465/AMR.1989.4308374>

Burton-Jones, A., McLean, E. R., & Monod, E. (2015). Theoretical perspectives in IS research: from variance and process to conceptual latitude and conceptual fit. *European Journal of Information Systems*, *24*(6), 664–679. <https://doi.org/10.1057/ejis.2014.31>

Gregor, S. (2006). The Nature of Theory in Information Systems. *MIS Quarterly*, *30*(3), 611–642.

Shapira, Z. (2011). „I’ve Got a Theory Paper--Do You?“: Conceptual, Empirical, and Theoretical Contributions to Knowledge in the Organizational Sciences. *Organization Science*, *22*(5), 1312–1321. <https://doi.org/10.1287/orsc.1100.0636>

Suddaby, R. (2010). Editor’s Comments: Construct Clarity in Theories of Management and Organization. *Academy of Management Review*, *35*(3), 346–357. <https://doi.org/10.5465/AMR.2010.51141319>

Sutton, R. I., & Staw, B. M. (1995). What Theory is Not. *Administrative Science Quarterly*, *40*(3), 371–384.

Weber, R. (2012). Evaluating and Developing Theories in the Information Systems Discipline. *Journal of the Association for Information Systems*, *13*(1), 1–30.

###### Anhangsverzeichnis

[A 1 Anhang Ebene 1 III](#_Toc516218198)

[A 1.1 Anhang Ebene 2 III](#_Toc516218199)

###### Anhang

#### Anhang Ebene 1

##### Anhang Ebene 2

**Eidesstattliche Erklärung**

Hiermit versichere ich, die vorliegende Arbeit mit dem Titel

„Thema“

selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.

Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt und wurde noch nicht veröffentlicht.

Dresden, 17.11.2018

|  |  |
| --- | --- |
| Vorname Nachname |  |