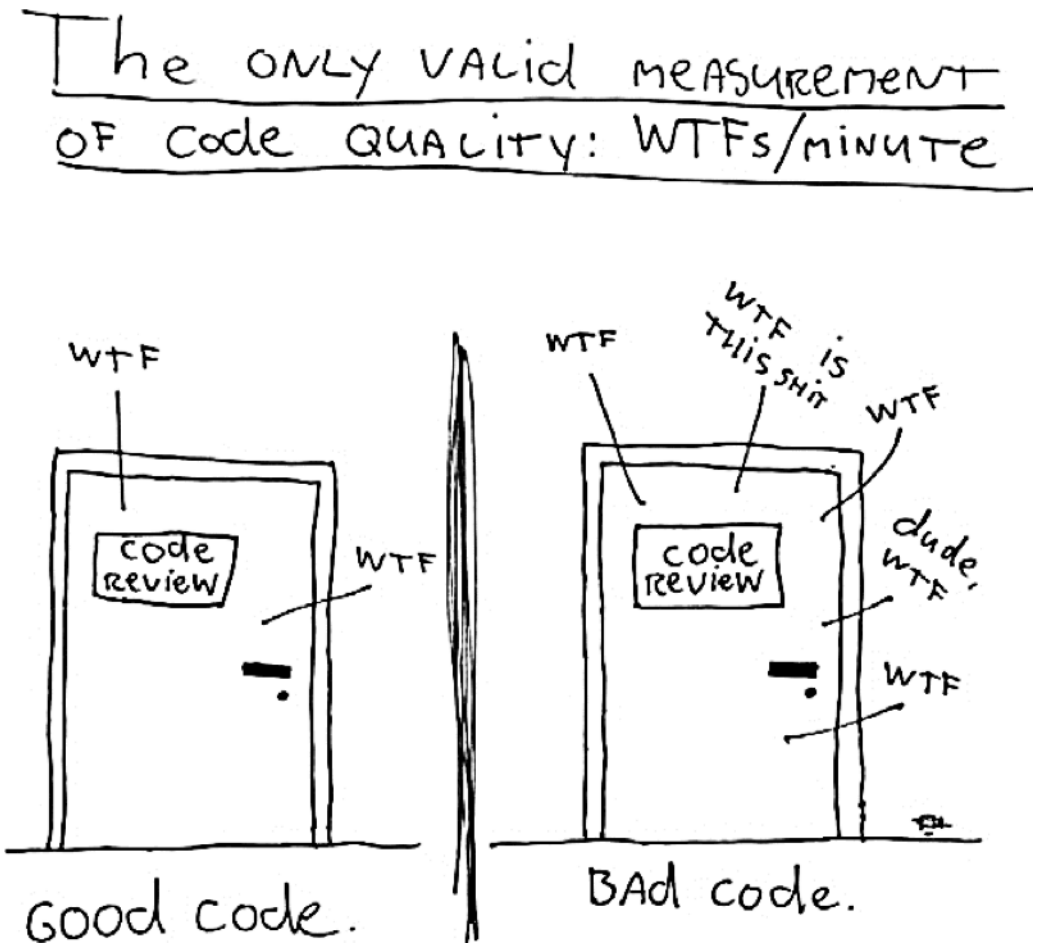
Software Construction

Florian Thiévent

3. Semester (HS 2018)



If you use this documentation for an exam, you should offer a beer to the authors!

Inhaltsverzeichnis

1 Software Construction

Problem Definition	**Detail Plan**	Team Management
Anforderungen spezifizieren	**Coding and Debugging**	Maintenance (Betrieb)
Planung der Anwendung	**Unit Testing**	**Integration**
SoftwareArchitektur		**Integration Testing**
		System Testing

1.1 Definition

Software construction is a fundamental act oft software engineering: the construction of working, **meaningful software** through a combination of **coding, validation and testing** by a programmer.

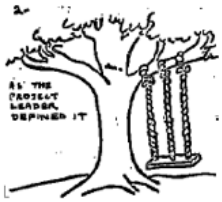
1.2 Low Level SWC

- Testing definieren
- Design und Klassen schreiben
- Erstellen und Naming der Variablen und Konstanten
- Unit Testing, Integration Testing und Debugging
- Review von anderem Code von Teammitgliedern
- Integration von Software
- Formatierung von Code und Kommentaren

1.3 Wieso ist SWC wichtig

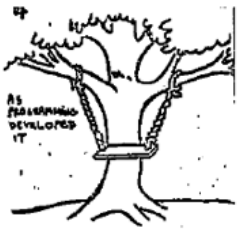
Construction is the only activity that's guaranteed to be done

As Management requested it.

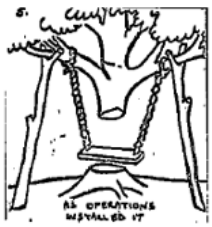


As the Project Leader defined it.

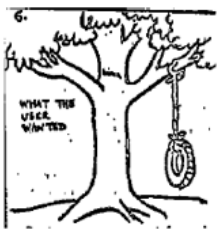
As Systems designed it.



As Programming developed it.



As Operations installed it.



What the user wanted.

1.4 Software Qualität

- Reliability: Zuverlässige Software welche fehlerfrei betrieben werden kann
- Reusability: Code Teile auch für andere Projekte nutzbar machen
- Extendibility: Erweiterbarkeit soll einfach möglich sein
- Understandability: Code soll verständlich für andere sein
- Efficiency: Geschwindigkeit
- Usability: Software einfach nutzen, auf das Zielpublikum ausgerichtet
- Portability: Einfach in eine andere Umgebung zügeln
- Functionality: Software soll funktional sein

Definition ISO: The totality of features and characteristics of a product or service that bear on its ability stated or implied needs
Definition IEEE: The degree to which a system, component, or process meets specified requirements.

1.5 Ziele

Die Software muss den Anforderungen des Kunden entsprechen. **"Good enough Software"not excellent software!**

1.6 Extreme Programming

1. Der Kunde/Auftraggeber ist immer verfügbar
2. Code wird gemäss vereinbarten Standards programmiert
3. Zuerst die Tests programmieren, dann den eigentlichen Code
4. Der produktive Code wird immer zu zweit programmiert
5. Nur ein Programmierer-Paar darf gleichzeitig Code integrieren
6. Integriere häufig
7. Jeder hat auf den gesamten Code Zugriff
8. Optimiere so spät wie möglich
9. Keine Überstunden
10. Das Team folgt gemeinsamen Code-Richtlinien, so dass es aussieht, als wenn der Code von einer einzigen Person geschrieben worden wäre
11. XP Projekte werden in sehr kurzen Abständen released (von täglich bis zu maximal alle 3-4 Wochen)

2 Version Control System

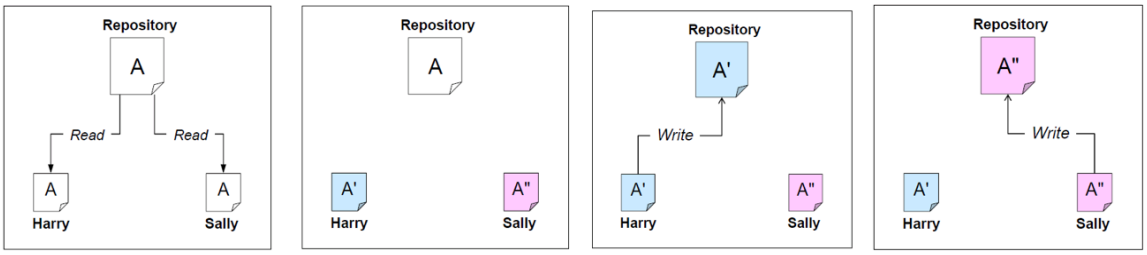
Es geht grundsätzlich um sich ständig ändernde Artefakte welche verwaltet werden müssen. Jede Änderung soll eindeutig nachvollziehbar sein.

2.1 Motivation für eine VCS

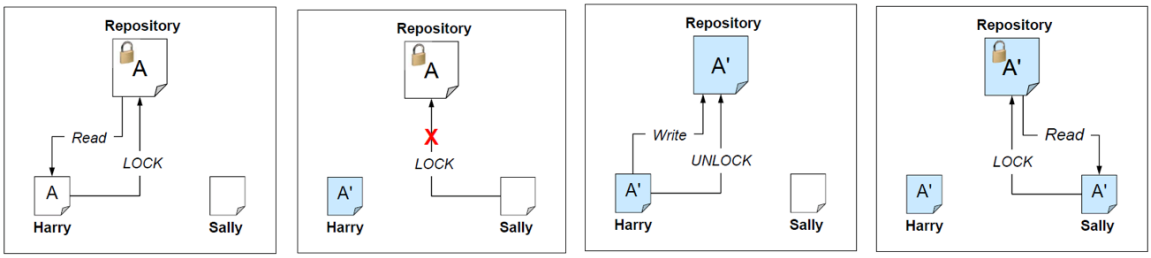
Ohne Versionsmanagement sieht der Alltag von Entwicklern so aus:

- Bugs die behoben wurden tauchen plötzlich wieder auf
- Dateien gehen verloren
- Frühere Releases der Software können nicht mehr erstellt werden
- Dateien werden äuf mysteriöse Art und Weise" verändert
- Gleicher oder ähnlicher Code existiert mehrfach in verschiedenen Projekten
- Zwei Entwickler ändern dieselbe Datei gleichzeitig ohne es zu merken

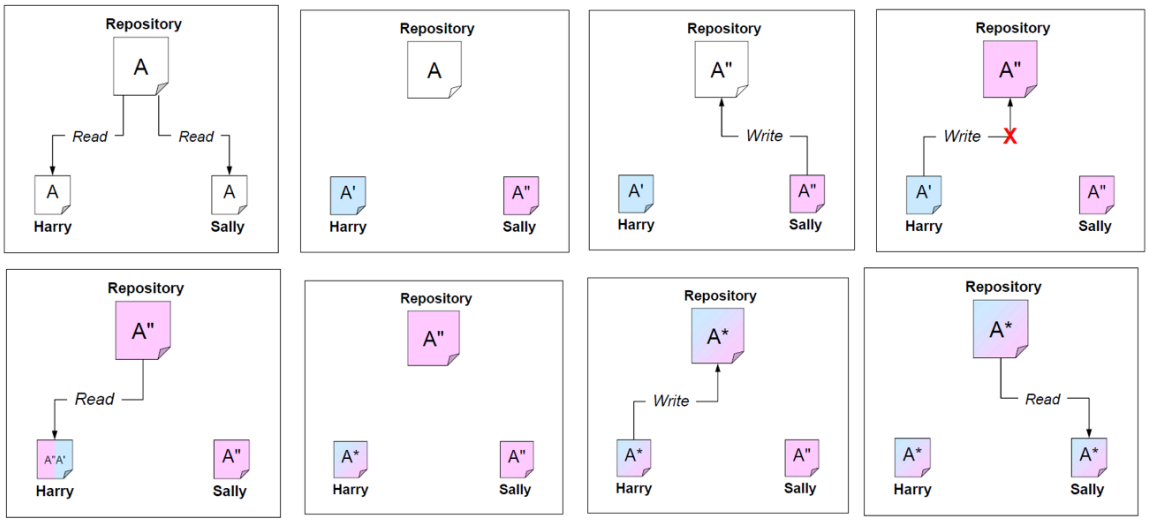
2.2 Problem of Filesharing



2.3 Lock-Modify-Unlock Solution



2.4 Copy-Modify-Merge Solution



2.5 Grundbegriffe Versions- und Release-Management

2.5.1 Version

Ein Zustand eines Konfigurationselementes mit einem klar definierten Funktionsumfang.

2.5.2 Release

Bezeichnet eine veröffentlichte Version eines Konfigurationselementes. Oft wird ein Release auch ausserhalb der Software-Entwicklungsorganisation zusammengestellt.

2.5.3 Major

Die Idee, das Major-Versionen (mindestens teilweise) inkompatibel sind

2.5.4 Minor

Änderungen dieser Nummer sind rückwärts kompatibel.

2.5.5 Patch

Bug-Fixes, Änderungen sind vorwärts- und rückwärts kompatibel.

2.5.6 Build

Diese Nummer gibt eine Neukompilierung von Sourcecode an, z.B. aufgrund von Prozessor-, Plattform- oder Compileränderungen.

2.5.7 Revision

Kleine Änderung an einer Version, welche Fehler behebt, jedoch keine Einfluss auf den Funktionsumfang haben.

2.5.8 Variante

Eine Variation einer Version welche entwickelt wurde um z.B. auf einer anderen Hardware oder unter einem anderen Betriebssystem zu laufen. Oder auch um für verschiedene Benutzergruppen feine Anpassungen vorzunehmen. Beispiele: Anpassungen für Tablets, Sehbehinderte, Touchscreens.

2.6 Grundbegriffe VCS

2.6.1 Repository

Eine Datenbank in welcher Projektdateien gespeichert werden. Ein Repository vergisst nichts, d.h. es ist nicht möglich eine Datei zu überschreiben. Vielmehr wir einfach eine neue Version der Datei gespeichert, die alte bleibt weiterhin im Repository und es kann auch weiterhin zugegriffen werden.

2.6.2 Working Copy

Eine *lokale Kopie* aller relevanten Projektdateien. Der Entwickler arbeitet immer auf dieser lokalen Kopie. Es wird also *nie* direkt auf den Dateien im Repository gearbeitet.

2.6.3 Checkout / Clone

Ist die Bezeichnung für einen Vorgang wenn eine Working Copy vom Repository bezogen wird. Dies ist eine reine Leseoperation auf dem Repository. Dabei wird auf der Entwicklermaschine eine neue Working Copy angelegt.

2.6.4 Commit / Push

Bezeichnet den Vorgang wenn eine Datei oder ein ganzes Set an Dateien (neu oder geändert) *mit einer Beschreibung* ins Repository gespeichert wird. Man spricht auch davon diese unter Versionskontrolle zu stellen. Dies ist eine Schreiboperation auf dem Repository die *atomar* erfolgt, d.h. pro Commit werden entweder alle oder gar keine Dateien ins Repository gespeichert. Damit ist sichergestellt, das das Set an Dateien konsistent ins Repository gespeichert wird. Es ist nicht möglich, dass durch zwei gleichzeitige Commits die Dateien durcheinandergebracht werden.

2.6.5 Update / Fetch / Pull

Bezeichnet den Vorgang wenn Dateien aus dem Repository mit der eigenen Working Copy abgeglichen werden. Indem andere Entwickler ihre Arbeit committen erhält das Repository neue Versionen welche auf den Working Copies der anderen Entwickler noch nicht vorhanden sind. Der Abgleich findet auf der Working Copy statt, für das Repository ist ein Updatevorgang somit eine reine Leseoperation.

2.6.6 Revision, Version

Jeder Commit verändert den Inhalt des Repositories und erzeugt somit eine neue Version oder Revision des Repositories die eindeutig identifizierbar sein muss. Dies ist notwendig um später wieder auf einen bestimmten Stand der Arbeit zurückzukehren zu können. Manche Repositories verwenden mehrstellige Versionsnummern (z.B. CVS), andere nummerieren die Commits einfach durch (z.B. Subversion) oder vergeben einen Hash (z.B. Git) als Identifikation.

2.6.7 Entwicklungsverlauf (Baseline, Codeline, Line of Development

TODO

2.6.8 Branch

TODO

2.6.9 Merging

TODO

2.6.10 Tag, Label

2.7 Configuration Items

Unter VCS	Nicht im VCS	Grenzfall
Sourcen	Testprotokolle (generiert)	DB
Dokumentation	Testreports (generiert)	Video
Kommunikation	Persönliche Konfig (IDE)	Audio
Tests	Javadoc → html	
Geteilte Config	DIE	
Video / Audiofiles (kleine)	Compilate	
	(.exe, .class, .jar	
	Files aus anderen Projekten	
	(Libraries, Vorlagen)	

3 Build - Automation

Die Komponenten einer Software werden per Knopfdruck erstellt. Macht meist mehrer Dinge auf einmal (Code compilieren, Tests durchlaufen, Checkout, Kopie auf Server, E-Mail). Der Buildvorgang kann auch automatisch gestartet werden.

3.1 Wieso wird dies benötigt, Probleme

TODO

3.2 Build Prozess