Projektmanagment und Teamarbeit

Autoren: Berger Dominik
Bernard Simon
Dalvai Ragnioli Peter
Di Pauli Maximilian Stefan
Domaneg Andreas
Gobbi Viktor
Hopfgartner Christopher
Mussner Marcel
Niederegger Andreas
Niedrist Alexander
Nocker Thomas
Oberprantacher Hannes
Pirpamer Damian
Pazeller Daniel
Prader Matthias
Rassele Phillip
Romen Rene
Rizzolli Thomas
Rufinatscha Samuel

Contents

DokuWiki

Philipp Rassele

May 21, 2015

1 DokuWiki

1.1 Was ist DokuWiki?

DokuWiki ist eine freie Wiki-Software,die in der Programmiersprache PHP geschrieben und unter der GPL 2 lizenziert wurde. Dokuwiki wurde von Andreas Gohr 2004 ins Leben gerufen. Um Inhalte und Metadaten zu speichern, werden einfache Textdateien genutzt und keine SQL-Datenbanken, wie bei anderen Wiki-Engines. Um die Wikiquellseiten gut leserlich zu halten, werden Inhalt und Metadaten von Wikiseiten bei DokuWiki strikt getrennt. Anfangs zur einfachen Dokumentation von Projekten gedacht, wird Dokuwiki mittlerweile aufgrund seiner Einfachheit und Funktionen für eine Vielzahl von Anwendungen eingesetzt. Auf Basis einer übersichtlichen Struktur lassen sich mit Erweiterungen (Plugins) weitere Funktionen hinzufügen, etwa für Blogs, Mediendaten oder Arbeitsgruppen.

1.2 Merkmale

1.2.1 Versionsverwaltung

Die Versionverwaltung speichert alle Versionen einer Wikiseite. Es ist möglich, ältere Versionen mit der aktuellen Version zu vergleichen. Außerdem wird verhindert, dass mehrere Benutzer gleichzeitig eine Seite verändern können.

1.2.2 Zugriffskontrolle

Die Zugriffsrechte lassen sich für Kombinationen von Benutzern, Gruppen und Namespaces vergeben. Die Einstellung ist via Webinterface (Usermanager) oder manuell per Konfigurationsdatei möglich (Access Control List).

1.2.3 Add-ons

DokuWiki hat einen einfachen Add-on-Mechanismus.Dadurch ist es möglich Erweiterungen (Plugins) in PHP zu schreiben. Es gibt inzwischen eine ganze Reihe an Erweiterungen (¿300). Über den Plug-in-Manager können diese über die Web-Oberfläche in das eigene Wiki integriert und verwaltet werden.

1.2.4 Templates

Das Aussehen des Wikis kann der Administrator über Templates festlegen. Es wurden inzwischen unterschiedliche Templates von der Entwicklergemeinde zur Verfügung gestellt.

1.2.5 Internationalisierung

Als Standard-Zeichencodierung wird UTF-8 verwnedet. Somit sind auch Sprachen wie Chinesisch, Thai oder Hebräisch darstellbar. Das Wiki selber kann momentan in 39 Sprachen konfiguriert werden.

1.2.6 Caching

Um den Server des Wikis zu entlasten, speichert ein Cache geparste Seiten. Bei einem erneuten Aufruf der Seite werden die gespeicherten Daten geliefert, anstatt die Wikiseite nochmals zu parsen.

1.2.7 Volltextsuche

DokuWiki hat eine Volltextsuche integriert, mit der in dem gesamten Wiki nach Stichwörtern gesucht werden kann.

1.2.8 WYSIWYG-Editor

Der Wiki-Philosophie einer einfachen Markup-Syntax entsprechend hat DokuWiki in der Grundausstattung keinen WYSIWYG-Editor. Diese Funktion kann aber über ein Plugin nachgerüstet werden; alternativ gibt es eine Quickbuttonleiste ähnlich MediaWiki.

1.2.9 Datenspeicherung

DokuWiki speichert alle Daten (aktuelle und alte Seiteninhalte, Indizes, Caches) in Textdateien. Dadurch ist keine separat laufende Datenbank (etwa MySQL) notwendig.

1.2.10 Versionierung/Synchronisation

Jede Wiki-Seite wird in einer Textdatei im Verzeichnis dokuwiki-JJJJ-MM-TT/data/pages gespeichert, der Name der Datei bleibt trotz Versionierung gleich. Vorherige Versionen befinden sich unter dokuwiki-JJJJ-MM-TT/data/attic. Es erfolgt kein Umbenennen/Neuanlegen der Originaldatei (z. B. Revision00011, Revision00012). Dies macht Dokuwiki ideal für Synchronisations-Tools mit beidseitigem Abgleich und diff-Funktion wie Unison.

1.2.11 Portable Version

Für Windows-Rechner ist DokuWiki auch als portable Version zusammen mit einem portablen Apache Webserver für die Verwendung auf einem USB-Stick vorhanden.

1.2.12 HTML5

Seit Release 2012-10-13 “Adora Belle” parst DokuWiki HTML5-Seiten.

Weitere Information über DokuWikie und ähnlicher Software findet man auf http://www.wikimatrix.org/.

2 Was ist Gradle

Gradle ist ein Build-Management Tool wie Maven, dass in Java geschrieben wurde. Anders als Maven verwendet Gradle Groovie um die zu bauenden Projekte zu beschreiben. Im Gegensatz zu Maven, das XML verwendet, sind Gradle-Skripts direkt ausführbarer Code.

Gradle wurde für Builds von Projekten konzipiert, die aus einer Vielzahl von Unterprojekten besteht. Da Builds umfangreicher Projekte sehr viel Zeit in Anspruch nehmen, unterstützt Gradle sowohl inkrementelles(st"uck f"ur st"uck) als auch Bauen der Software durch parallel ablaufende Build-Prozesse. Dadurch k"onnen nur Teile eines Projektes gebaut werden, die ge"andert wurden, ohne das Ganze Projekt neu bauen zu m"ussen. Des Weiteren k"onnen Tests parallel zum laufenden Build gestartet werden. Dadurch wird die Geschwindigkeit des Bauens des Projektes um ein Vielfaches erh"oht

Viele bekannte Frameworks verwenden Gradle, wie zum Beispiel Hibernate, Groovy, Grails und Spring Security. Auch Android ist seit Mitte 2013 hinzugekommen. Auch sogenannte "nativer" Systeme, welche nicht auf der Java-Plattform basieren, werden unterstützt. Dazu gehören die Programmiersprachen C++, C, Objective C und Assembler.

3 Aufbau

In Gradle braucht man in den meisten F"allen nur die build.gradle Datei. Gradle verf"ugt jedoch "Uber 2 weitere Dateien, die aber nicht unbedingt erforderlich sind.

- build.gradle - enth"allt die Definition der Tasks und der ganzen Abh"angigkeiten des Projektes
- settings.gradle - hier werden die teilnehmenden Unterprojekte eines Multiprojekts definiert.
- gradle.properties - enth"allt eine Liste von Eigenschaften f"ur eine projektspezifische Gradle-Initialisierung eines Builds

Weiteres wird in Gradle zwischen Tasks und Eigenschaften(Properties) unterschieden.

Tasks sind ausf"uhrbarer Code, der "uber die Kommandozeile aufgerufen wird, indem man dort "gradle [den auszuf"uhrenden Task]" eingibt.

Durch Properties werden die Eigenschaften des Builds, wie z.B. Pfad der Klassen und Abh"angigkeiten definiert werden. Eigenschaften k"onnen in zwei verschiedenen Weisen definiert werden:

1. `sourceSets.main.java.srcDir "src/main/java"`

2.
```
sourceSets{
      main{
          java{
              srcDir "src/main/java"
          }
      }
  }
```

Beide Schreibweisen funktionieren, wobei meistens eine Kombination von 1. und 2. bevorzugt wird.

4 Einfache Grundtasks

Das Build-Script wird in der "build.gradle" geschrieben.
Zum Anfang ein einfaches Hello-World Programm:
In der build.gradle:

```
task helloworld {
    String hallo = 'Hallo'
    String welt = ' Welt!'
    println hallo + welt.toUpperCase()
}
```

5 Erstellen eines einfachen Java-Builds

Wie bei den Tasks, werden auch Build-Scripts in der build.gradle definiert:

```
apply plugin: "java"

sourceSets {
    main.java.srcDir "src/main/java"
}

jar {
    manifest.attributes "Main-Class": "net.tfobz.test.Test"
}
```

- Als erstes wird apply plugin: java gesetzt, damit Gradle mit Java-Projekten überhaupt umgehen kann.
- Durch sourceSets.main.java.srcDir , kann der Pfad für die Java-Packages gesetzt werden. Wird diese Eigenschaft nicht gesetzt, dann wird standartm"asig "src/main/java" als Pfad gew"ahlt.
- Die Eigenschaft jar.manifest.attributes erg"anzt die Manifest-Datei mit den gesetzten Attributen(in diesem Fall der Pfad der Main-Klasse).

5.1 JUnit-Tests mit Gradle durchf"uhren

Die Tests werden in Gradle, "ahnlich wie in Maven, getrennt vom Quellcode abgelegt. Standardm"a"sig ist der Pfad "src/main/test" eingestellt. Man kann diesen Pfad aber, wie es auch im vorherigen Beispiel gemacht wurde, "andern.

Folgendes Beispiel soll zeigen, wie man JUnit in seinem Projekt integrieren kann:

```
repositories {
    mavenCentral()
}

sourceSets {
    main.java.srcDir "src/main/java"
    test.java.srcDir "src/main/test"
}

dependencies {
    testCompile 'junit:junit:4.12'
}

test {
    testLogging {
        events 'started', 'passed', 'failed'
    }
}
```

- Die Funktion mavenCentral() in der Eigenschaft repositories holt die erforderlichen Bibliotheken f"ur JUnit(Internetverbindung erforderlich).
- In sourceSets.test.java.srcDir kann der Pfad der JUnit Testklassen ge"andert werden.
- In der Eigenschaft testCompile in dependencies wird die zu verwendende JUnit-Version angegeben.
- In der Eigenschaft test.testLogging.events werden die anzuzeigenden JUnit-Meldungen angegeben. Wird z.B. 'failed' weggelassen, so werden die gescheiterten JUnit-Tests nicht in der Konsole ausgegeben (Build schl"agt aber trotzdem fehl).

5.2 Java Projekt eclipsf"ahig machen

Um ein Gradle-Projekt eclipsf"ahig zu machen braucht man nur das Plugin "eclipse" hinzuzuf"ugen.

```
apply plugin: "java"
apply plugin: "eclipse"
```

- Gibt man in der Kommandozeile den Befehl “gradle eclipse” ein, werden die ben”otigten Dateien f’ur Eclipse erzeugt. Das Projekt an sich wird aber weder compiliert noch getestet.
- Mit dem Befehl “gradle cleanEclipse” werden die erzeugten Dateien wieder entfernt.

6 Microsoft Dynamics AX

6.1 Einführung

Microsoft Dynamics® AX ist eine objektorientierte ERP-Lösung von Microsoft® für mittelständische Unternehmen und Großunternehmen, die es ermöglicht, Wandel und Veränderungen aus dem Geschäftsumfeld aufzunehmen und mitzugestalten und so den Geschäftserfolg voranzutreiben. Es handelt sich um eine leistungsstarke Lösung, mit der man in kürzester Zeit einen nachhaltigen Nutzen erzielen kann. Dank der 36 verfügbaren Landesversionen eignet sich die Software insbesondere für den Einsatz in multinationalen Organisationen. Microsoft Dynamics AX 2012 stellt den Beginn eines neuen Produktivitätszeitalters bei Unternehmenslösungen dar. Es handelt sich um die neueste Version der erfolgreichen ERP Software Reihe und beeindruckt mit einem deutlichen Zuwachs an Funktionalität, einer völlig neuen Agilität und einer Benutzeroberfläche, die überzeugt und Produktivität beim Anwender fördert.

6.2 Ausgangssituation zur Entwicklung der ERP Software

Zur Entwicklung dieser ERP Software wurden relevante aktuelle Business und IT-Anforderungen einbezogen: steigender globaler Wettbewerb, verbunden mit signifikantem Kosten- und Preisdruck, Organisationsänderungen, aber Prozessoptimierungen, das Entstehen von komplexen internationalen Netzwerken oder erhöhte Anforderungen durch steigende gesetzliche Kontrolle sind wichtige Trends, der sich Unternehmen stellen müssen.Es wird absolute Transparenz für schnelle und richtige Entscheidungen gefordert. Viele Organisationen haben aber zunehmend Schwierigkeiten, mit diesen Trends bzw. Beweggründen Schritt zu halten, da ein altes und überholtes ERP System weder die nötige Flexibilität noch Prozessorientierung aufweist, um hier effizient agieren zu können.

6.3 Die vier Prinzipien von AX

6.4 Leistungsstartk

Folgende Eigenschaften tragen zur hohen Performanz der Microsoft ERP Software bei:

- Umfassende ERP Software-Funktionen sowohl für administrative als auch für operative Unternehmensbereiche
- Reports können durch direkte Integration der Business Intelligence Funktionen des SQL-Servers rasch und effektiv verarbeitet werden.
- Tasks eines Benutzers werden in einem zentralen Aufgabenbereich zusammengeführt und gegebenenfalls einer funktionalen Eingabewarteschlange zugewiesen.

6.5 Agil

Die Geschäftstätigkeit des Unternehmens kann durch eine Sammlung von einheitlichen Organisationsmodellen, überwacht, analysiert, bewertet und gegebenenfalls modifiziert werden. Das Microsoft Dynamics ERP System löst zwei essentielle Fragen auf überzeugende Art und Weise: Wie gut bildet das ERP System die reale Welt ab und wie schnell lässt sie sich an das Unternehmen anpassen?

6.6 Einfach

Das Prinzip der Einfachheit spiegelt sich bei Installation, Implementierung, Anpassung, der Anwendungsoberfläche sowie bei Updates wieder, wobei die Grundlage zur Einfachheit die vertraute Benutzeroberfläche bildet.

6.7 Global

Eine globale Ausrichtung der aktuellen Microsoft ERP Software wird durch folgende Funktionen erreicht:

- Ab dem AX 2009-Release wurden Lokalisierungen für 38 Länder zur Verfügung gestellt.
- Zentrale Stammdatenhaltung für Produkte und Geschäftspartneradressen wurden mit dem aktuellsten AX 2012 Release hinzugefügt.

7 Activiti

Activiti ist ein freies Workflow-Management-System mit dem man Businessprozesse definieren und ausführen kann. Es ist in Java geschrieben und damit Plattform unabhängig. Es ist ausgerichtet auf Geschäfts Leute, Entwickler und System Administratoren. Die Activiti Engine ist leight weight und auf Java Entwickler ausgelegt.

7.1 Übersicht

In folgender Grafik sind alle Teile der Activiti Software in den nächsten Kapitel werden die wichtigsten erklärt.

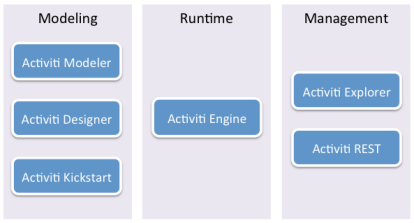

Figure 1: Teile der Activiti Software

7.2 Activiti Engine

Die Activiti Engine ist das Herzstück des Projekts. Es ist eine Business Process Model and Notation 2 Engine mit API für Java. Es gibt dem Entwickler die Möglichkeit mit Listener bei bestimmten Ereignissen eigenen Java Code einzubauen um so mehr technische Details ins Diagramm zu bringen. Des weiteren können eigene Aktivitäten für das Diagramm definiert werden. Die ganze Engine ist von Anfang an darauf ausgelegt Cloud fähig zu sein.

7.3 Activiti Explorer

Der Activiti Explorer ist eine Webanwendung mit der das System Verwaltet werden kann. Sie ermöglicht alles vom erstellen neuer Aufgaben bis zum Abschließen jener. Es können Arbeiter verwaltet werden und deren Aufgaben ebenfalls. Für User zeigt die Webanwendung ihre aktuellen Aufgaben an und welche sie als nächstes machen müssen. Die Anwendung führt dabei History wer wann was macht.

7.4 Activiti Modeler

Der Activiti Modeler ist eine Webanwendung die es ermöglicht Grafische BPMN 2.0 Diagramme zu erstellen die ausgeführt werden können von der Activiti Engine.

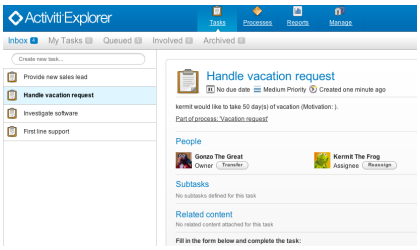

Figure 2: Der Activiti Explorer

7.5 Activiti Designer

Der Activiti Designer ist im Grunde das selbe wie der Activiti Modeler, es ist aber ein Eclipse plugin. Entwickler können damit direkt in Eclipse BPMN Diagramme für ihre Programme erstellen.

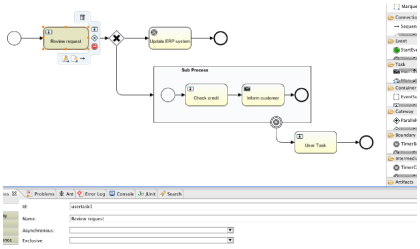

Figure 3: Der Activiti Designer

8 GIT Versionskontrollsystem

8.1 Geschichte

Torvalds wünschte sich ein verteiltes System, das wie BitKeeper genutzt werden konnte und die folgenden Anforderungen erfüllte:
1. Unterstützung verteilter, BitKeeper-ähnlicher Arbeitsabläufe
2. Sehr hohe Sicherheit gegen sowohl unbeabsichtigte als auch böswillige Verfälschung
3. Hohe Effizienz
Ein bereits existierendes Projekt namens Monotone entsprach den ersten zwei Anforderungen, das dritte Kriterium wurde jedoch von keinem bestehenden System erfüllt. Torvalds entschied sich dagegen, Monotone an seine Anforderungen anzupassen, und begann stattdessen, ein eigenes System zu entwickeln. Einer der Hauptgründe für diesen Schritt war die Arbeitsweise, für die Monotone nach Torvalds Ansicht optimiert ist. Torvalds argumentierte, dass einzelne Revisionen von einem anderen Entwickler in den eigenen Entwicklungszweig zu importieren zu Rosinenpickerei und unordentlichen“ Repositories führen würde. Wenn hingegen immer ganze Zweige importiert werden, wären Entwickler gezwungen aufzuräumen. Dazu seien Wegwerf-Zweige“ notwendig. Gits Gestaltung verwendet einige Ideen aus Monotone sowie BitKeeper, aber keinen Quellcode daraus. Es soll ausdrücklich ein eigenständiges Versionsverwaltungssystem sein.

8.2 Eigenschaften

8.2.1 Nicht-lineare Entwicklung

Sowohl das Erstellen neuer Entwicklungszweige (branching), als auch das Verschmelzen zweier oder mehrerer Zweige (merging) sind integraler Bestandteil der Arbeit mit Git und fest in die Git-Werkzeuge eingebaut. Git enthält Programme, mit deren Hilfe sich die nicht-lineare Geschichte eines Projektes einfach visualisieren lässt und mit deren Hilfe man in dieser Geschichte navigieren kann. Ein Branch stellt nur eine Reference, eine Textdatei mit einer Commit-ID, dar, die in einem Repository im Verzeichnis .git/refs/heads liegt und auf einen bestimmten Commit verweist. Über dessen Elterncommits, lässt sich die Branchstruktur rekonstruieren. Durch diese Eigenschaften lassen sich weiterhin sehr große und effiziente Entwicklungsstrukturen realisieren, bei denen jedes Feature und jeder Entwickler einen Branch oder ein eigenes Repository haben, aus dem der Maintainer dann Commits über Merge oder Cherry-pick (Nutzen einzelner Commits) in den Hauptzweig des Projekts (master) übernehmen kann.

8.2.2 Kein zentraler Server

Jeder Benutzer besitzt eine lokale Kopie des gesamten Repositorys, inklusive der Versionsgeschichte (history). So können die meisten Aktionen lokal und ohne Netzwerkzugriff ausgeführt werden. Es wird nicht zwischen lokalen Entwicklungszweigen und Entwicklungszweigen entfernter Repositories unterschieden.

Obwohl es keinen technischen Unterschied zwischen verschiedenen Repositories gibt, gilt die Kopie, auf die von einer Projekt-Homepage aus verwiesen wird, häufig als das offizielle Repository“, in das die Revisionen der Entwickler übertragen werden.

8.2.3 Datentransfer zwischen Repositories

Daten können neben dem Übertragen auf Dateisystemebene (file://) mit einer Reihe verschiedener Netzwerkprotokolle zwischen Repositories übertragen werden. Git besitzt ein eigenes sehr effizientes Protokoll, das den TCP-Port 9418 nutzt (git://), allerdings nur zum Fetchen und Clonen genutzt werden kann, also dem Lesen eines Repositorys. Ebenso kann der Transfer über SSH (ssh://), HTTP (http://), HTTPS (https://) oder über (weniger effizient) FTP (ftp://) erfolgen. Die Übertragung in das offizielle Repository“ eines Projekts erfolgt häufig in Form von Patches.] Für Projekte, die auf Webseiten wie GitHub oder BitBucket gehostet werden, kann eine Änderung einfach durch das Pushen eines Branches vorgeschlagen werden, der dann bei Bedarf durch ein Merge in das Projekt integriert wird.

8.2.4 Speichersystem und Dateiversionierung

Im Gegensatz zu CVS, wo jede Datei eine eigene Revisionsnummer besitzt, speichert Git bei einem Commit das gesamte Verzeichnis ab. Wenn eine Datei in einem Commit nicht geändert wird, ändert sich auch die Checksumme nicht und sie muss nicht nochmals gespeichert werden. Die Objekte liegen im Projekt unter .git/objects.

Figure 4: Maven Logo

9 Apache Maven

9.1 Einleitung

Maven ist ein Build-Management-Tool der Apache Software Foundation und basiert auf Java. Mit ihm kann man insbesondere Java-Programme standardisiert erstellen und verwalten. Der Name Maven kommt aus dem Jiddischen und bedeutet so viel wie "Sammler des Wissens".

9.2 Design

Maven basiert auf einer Plugin-Architektur, die es ermöglicht, Plugins für verschiedene Anwendungen auf das Projekt anzuwenden, ohne diese explizit installieren zu müssen.
Die Bandbreite reicht von Plugins, die es ermöglichen, direkt aus Maven heraus eine Webanwendung zu starten, um sie im Browser zu testen, über welche, die es ermöglichen, Datenbanken zu testen oder zu erstellen, bis hin zu solchen, die Web Services generieren. Die dafür nötigen Tätigkeiten beschränken sich häufig nur darauf, das gewünschte Plugin zu ermitteln und einzusetzen.

9.3 Lebenszyklus

Der Standard-Lebenszyklus Maven geht von einem Zyklus aus, der bei der Softwareerstellung häufig durchlaufen wird. Dabei wird nicht unterstellt, dass jedes Softwareprojekt alle Phasen des im Folgenden dargestellten Zyklus verwendet:

- archetype (Scaffolding): Damit kann ein Template für ein Softwareprojekt erstellt werden. Abhängigkeiten werden aufgelöst und bei Bedarf heruntergeladen.
- validate (Validieren): Hier wird geprüft, ob die Projektstruktur gültig und vollständig ist.
- compile (Kompilieren): In dieser Phase wird der Quellcode kompiliert.
- test (Testen): Hier wird der kompilierte Code mit einem passenden Testframework getestet. Maven berücksichtigt dabei in späteren Zyklen, dass Testklassen normalerweise nicht in der auszuliefernden Software vorhanden sind.
- package (Verpacken): Das Kompilat wird – ggf. mit anderen nichtkompilierbaren Dateien – zur Weitergabe verpackt. Häufig handelt es sich dabei um eine Jar-Datei.
- integration-test (Test der Integrationsmöglichkeit): Das Softwarepaket wird auf eine Umgebung (anderer Rechner, anderes Verzeichnis, Anwendungsserver) geladen und seine Funktionsfähigkeit geprüft.
- verify (Gültigkeitsprüfung des Software-Pakets): Prüfungen, ob das Softwarepaket eine gültige Struktur hat und ggf. bestimmte Qualitätskriterien erfüllt.
- install (Installieren im lokalen Maven-Repository): Installiere das Softwarepaket in dem lokalen Maven-Repository, um es in anderen Projekten verwenden zu können, die von Maven verwaltet werden. Dies ist insbesondere für modulare Projekte interessant.
- deploy (Installieren im fernen Maven-Repository): Stabile Versionen der Software werden auf einem fernen Maven-Repository installiert und stehen damit in Umgebungen mit mehreren Entwicklern allen zur Verfügung.

9.4 Unterstützte Entwicklungsumgebungen

Für die gängigsten Entwicklungsumgebungen (z. B. Eclipse, IntelliJ IDEA oder NetBeans) sind Plugins vorhanden, über die sich Maven direkt aus der Entwicklungsumgebung heraus bedienen lässt. Zusätzlich sind Maven-Plugins vorhanden, die Dateien erzeugen, welche den Import eines reinen Maven-Projekts in die Entwicklungsumgebung ermöglichen

9.5 Teilprojekte

Die Entwicklung von Maven ist in verschiedene Teilprojekte untergliedert.

- Maven 1.x pflegt die älteren Versionen von Maven.
- Maven 2 entwickelt die aktuelle Produktlinie von Maven weiter.
- Plugins entwickelt die meisten Maven-Plugins.
- Shared Components stellt Softwarekomponenten bereit, die von den anderen Teilprojekten verwendet werden können.
- Ant Tasks ermöglicht es, Maven-Funktionalität aus Ant-Skripten heraus zu verwenden.
- Doxia ist ein Framework zum Generieren von Content aus den Formaten Almost Plain Text (APT), Confluence, DocBook, FML (FAQ Markup Language), LaTeX, Markdown, Rich Text Format (RTF), TWiki, XDoc und XHTML.
- SCM (Source Code Management) entwickelt Software für die Anbindung von Apache an verschiedene Systeme zur Versionsverwaltung wie CVS oder Subversion.
- Surefire entwickelt ein Testframework für Maven.
- Wagon stellt eine Abstraktionsschicht für Kommunikationsprotokolle wie "Dateizugriff", HTTP oder FTP bereit.

9.6 Versionsgeschichte

Die erste Version, Maven 1.x, wurde im Jahre 2003 eingeführt und am 13. Juli 2004 als Version 1.0 fertiggestellt. Die Umsetzung passierte jedoch sehr schnell, sodass einige Eigenheiten nicht bedacht wurden. Beispielsweise gibt es Performanceprobleme und zu viele Konfigurationsdateien und -angaben.

Deshalb wurde das Konzept überarbeitet und seit dem Jahre 2005 parallel begonnen, Maven 2.x zu entwickeln, welches in Version 2.0 am 19. Oktober 2005 fertiggestellt wurde.

Maven 1.x wird nicht mehr weiterentwickelt und beschränkt sich auf Support und Bugfixes. Die letzte Version von Maven 1.x (Version 1.1) wurde am 25. Juni 2007 ausgeliefert.

Die Entwicklung von Maven 3.0 begann im Jahr 2008. Nach acht Alpha-Releases wurde die erste Beta-Version von Maven 3.0 im April 2010 veröffentlicht. Besonderes Augenmerk lag auf der Kompatibilität zwischen Maven 2 und 3.

10 IRC

Internet Relay Chat (oder kurz IRC) bezeichnet ein vollkommen textbasiertes Chat-System. Hier treffen sich die Teilnehmer in sogenannten Channels und können in diesen Channels miteinander schreiben. Jeder Benutzer kann jederzeit einen neuen Channel erstellen. Außerdem ist es möglich sogenannte Querys zu erstellen, welche die Kommunikation zwischen nur zwei Gesprächsteilnehmern ermöglicht.

10.1 Geschichte

Die erste Version eines solchen Chat-Netzwerks entstand 1985 im BITNET, einem Netzwerk von Großrechnern in den USA. Damals noch unter dem Namen Relay Chat. Zwei Studenten von der Universität Oulu, in Finnland übertrugen dieses System 1988 auf das Internet. Die erste Version dieses Chat-Netzwerks erlangte schnell Beliebtheit, sodass bald weitere Unabhängige IRC Netze entstanden.

10.2 Allgemein

Ein IRC-Netzwerk besteht aus mehreren miteinander verbundenen Servern, welche auch Relay-Station genannt werden. Jeder Chat Teilnehmer verbindet sich immer mit einem Server und klingt sich somit in das Netz ein. Ein besonderes Merkmahl des IRC-Netzwerks ist, dass zwischen zwei Teilnehmern des Chats immer nur eine Verbindung besteht. Dies war historisch sehr bedeutsam, da früher die Leitungskapazität sehr begrenzt war. So wird jede Nachricht die von den Benutzern, die mit einem Server verbunden sind, nur über einen Kanal an einen weiteren Server und somit an deren Teilnehmer geschickt. Dieser Server schickt seine Nachricht daraufhin an den nächsten Server weiter. Diese Funktionsweise sorgt für eine sehr geringe Menge an Datenverkehr, jedoch fehlt gleichzeitig jede Art von Redundanz. So kann es beim Ausfall eines Servers passieren, dass sich ein Netz in zwei kleinere Netze aufspaltet zwischen denen keine Verbindung mehr besteht.

10.3 Technische Eigenschaften

IRC ist ein auf IP und TCP basierendes Protokoll. Die gesamte Kommunikation besteht aus Textnachrichten mit einer maximalen Länge von 512 Zeichen. Diese Textnachrichten enthalten einen Absender, einen Befehl und je nach Befehl eventuelle Befehlsparameter. Auf einen Befehl kann eine Antwort des Servers folgen, dies ist jedoch von Befehl zu Befehl und von Server zu Server unterschiedlich. Im Laufe der Zeit entwickelten sich außerdem verschiedene Erweiterungen zum IRC Protokoll, welche unter anderem den Aufbau der Nachrichten ändern.

10.4 Verschlüsselung

Ob und wie Nachrichten verschlüsselt werden ist von Netz zu Netz unterschiedlich. In den meisten Fällen wird entweder gar nicht verschlüsselt oder eine über SSL/TLS verschlüsselte Verbindung benutzt. Verschiedene Erweiterungen benutzen hier ebenfalls verschiedene Verschlüsselungen.

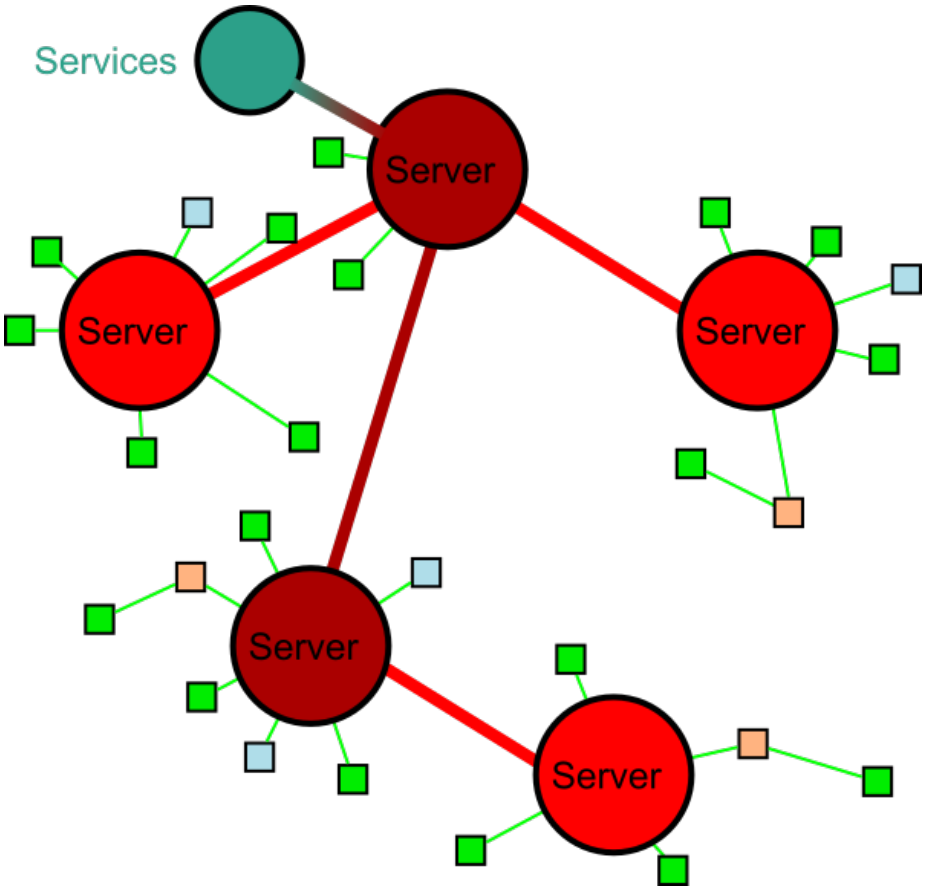

Figure 5: Schematischer Aufbau eines IRC-Netzes.

Figure 6: Maven Logo

11 Apache Maven

11.1 Einleitung

Maven ist ein Build-Management-Tool der Apache Software Foundation und basiert auf Java. Mit ihm kann man insbesondere Java-Programme standardisiert erstellen und verwalten. Der Name Maven kommt aus dem Jiddischen und bedeutet so viel wie "Sammler des Wissens".

11.2 Design

Maven basiert auf einer Plugin-Architektur, die es ermöglicht, Plugins für verschiedene Anwendungen auf das Projekt anzuwenden, ohne diese explizit installieren zu müssen.
Die Bandbreite reicht von Plugins, die es ermöglichen, direkt aus Maven heraus eine Webanwendung zu starten, um sie im Browser zu testen, über welche, die es ermöglichen, Datenbanken zu testen oder zu erstellen, bis hin zu solchen, die Web Services generieren. Die dafür nötigen Tätigkeiten beschränken sich häufig nur darauf, das gewünschte Plugin zu ermitteln und einzusetzen.

11.3 Lebenszyklus

Der Standard-Lebenszyklus Maven geht von einem Zyklus aus, der bei der Softwareerstellung häufig durchlaufen wird. Dabei wird nicht unterstellt, dass jedes Softwareprojekt alle Phasen des im Folgenden dargestellten Zyklus verwendet:

- archetype (Scaffolding): Damit kann ein Template für ein Softwareprojekt erstellt werden. Abhängigkeiten werden aufgelöst und bei Bedarf heruntergeladen.
- validate (Validieren): Hier wird geprüft, ob die Projektstruktur gültig und vollständig ist.
- compile (Kompilieren): In dieser Phase wird der Quellcode kompiliert.
- test (Testen): Hier wird der kompilierte Code mit einem passenden Testframework getestet. Maven berücksichtigt dabei in späteren Zyklen, dass Testklassen normalerweise nicht in der auszuliefernden Software vorhanden sind.
- package (Verpacken): Das Kompilat wird – ggf. mit anderen nichtkompilierbaren Dateien – zur Weitergabe verpackt. Häufig handelt es sich dabei um eine Jar-Datei.
- integration-test (Test der Integrationsmöglichkeit): Das Softwarepaket wird auf eine Umgebung (anderer Rechner, anderes Verzeichnis, Anwendungsserver) geladen und seine Funktionsfähigkeit geprüft.
- verify (Gültigkeitsprüfung des Software-Pakets): Prüfungen, ob das Softwarepaket eine gültige Struktur hat und ggf. bestimmte Qualitätskriterien erfüllt.
- install (Installieren im lokalen Maven-Repository): Installiere das Softwarepaket in dem lokalen Maven-Repository, um es in anderen Projekten verwenden zu können, die von Maven verwaltet werden. Dies ist insbesondere für modulare Projekte interessant.
- deploy (Installieren im fernen Maven-Repository): Stabile Versionen der Software werden auf einem fernen Maven-Repository installiert und stehen damit in Umgebungen mit mehreren Entwicklern allen zur Verfügung.

11.4 Unterstützte Entwicklungsumgebungen

Für die gängigsten Entwicklungsumgebungen (z. B. Eclipse, IntelliJ IDEA oder NetBeans) sind Plugins vorhanden, über die sich Maven direkt aus der Entwicklungsumgebung heraus bedienen lässt. Zusätzlich sind Maven-Plugins vorhanden, die Dateien erzeugen, welche den Import eines reinen Maven-Projekts in die Entwicklungsumgebung ermöglichen

11.5 Teilprojekte

Die Entwicklung von Maven ist in verschiedene Teilprojekte untergliedert.

- Maven 1.x pflegt die älteren Versionen von Maven.
- Maven 2 entwickelt die aktuelle Produktlinie von Maven weiter.
- Plugins entwickelt die meisten Maven-Plugins.
- Shared Components stellt Softwarekomponenten bereit, die von den anderen Teilprojekten verwendet werden können.
- Ant Tasks ermöglicht es, Maven-Funktionalität aus Ant-Skripten heraus zu verwenden.
- Doxia ist ein Framework zum Generieren von Content aus den Formaten Almost Plain Text (APT), Confluence, DocBook, FML (FAQ Markup Language), LaTeX, Markdown, Rich Text Format (RTF), TWiki, XDoc und XHTML.
- SCM (Source Code Management) entwickelt Software für die Anbindung von Apache an verschiedene Systeme zur Versionsverwaltung wie CVS oder Subversion.
- Surefire entwickelt ein Testframework für Maven.
- Wagon stellt eine Abstraktionsschicht für Kommunikationsprotokolle wie "Dateizugriff", HTTP oder FTP bereit.

11.6 Versionsgeschichte

Die erste Version, Maven 1.x, wurde im Jahre 2003 eingeführt und am 13. Juli 2004 als Version 1.0 fertiggestellt. Die Umsetzung passierte jedoch sehr schnell, sodass einige Eigenheiten nicht bedacht wurden. Beispielsweise gibt es Performanceprobleme und zu viele Konfigurationsdateien und -angaben.

Deshalb wurde das Konzept überarbeitet und seit dem Jahre 2005 parallel begonnen, Maven 2.x zu entwickeln, welches in Version 2.0 am 19. Oktober 2005 fertiggestellt wurde.

Maven 1.x wird nicht mehr weiterentwickelt und beschränkt sich auf Support und Bugfixes. Die letzte Version von Maven 1.x (Version 1.1) wurde am 25. Juni 2007 ausgeliefert.

Die Entwicklung von Maven 3.0 begann im Jahr 2008. Nach acht Alpha-Releases wurde die erste Beta-Version von Maven 3.0 im April 2010 veröffentlicht. Besonderes Augenmerk lag auf der Kompatibilität zwischen Maven 2 und 3.

12 Mercurial

Figure 7: Logo von Mercurial

Mercurial ist ein verteiltes, programmunabhängiges Versionskontrollsystem.

12.1 Geschichte

Angekündigt wurde es von Matt Mackall auf der Linux-Kernel-Mailingliste am 19. April 2005. Diese Ankündigung war die Folge, dass die Firma BitMover, die z. B. für den Linux-Kernel als Versionskontrollsystem eingesetzte Software BitKeeper nicht mehr in einer kostenlosen Version bereitstellte. Zur selben Zeit startete Linus Torvalds das Projekt Git, das sich in der Folgezeit besser etablieren konnte als Mercurial.

12.2 Implementierung

Es ist nahezu in vollständig in Python entwickelt, weshalb es nicht unbedingt empfehlenswert wäre, Mercurial zu benutzen. Lediglich eine diff-Implementierung, die mit binären Dateien umgehen kann, ist in C entwickelt, da das in Python nicht möglich ist. Nichtsdestotrotz sind Effizienz, Skalierbarkeit und robuste Handhabung von Text- und Binärdateien einige Punkte, die bei der Entwicklung als Schwerpunkte festgelegt wurden.

Ähnlich wie Git ist Mercurial kein zentralisiertes Versionskontrollsystem, d.h. man kann ein Repository klonen kann und auf einer lokalen Kopie darauf arbeiten. Auf dieser Kopie kann man ganz normal auf den die Funktionen von Mercurial verwenden. Ebenso ist die Fähigkeit des Erstellens und Zusammenfügens von Entwicklungszweigen ein fester Bestandteil dieses Versionskontrollsystems. Des Weiteren kann man einfach und schnell Unterschiede zwischen zwei unterschiedlichen Versionen anzeigen lassen. Zudem ist es möglich bei der Version Sekunden anzugeben, die man zurückspringen will.

12.3 Zugriff

Man kann sowohl über eine grafische Oberfläche als auch über die Komandozeile auf die Funktionen von Mercurial zugreifen. Eine grafische Oberfläche wird häufig bei Microsoft Windows, Gnome/Nautilus (jeweils TortoiseHg) und bei Mac OS X (MacHg und Murky). Bei gängigen Entwicklungsumgebungen wie Netbeans, Eclipse, Android Studio oder der Qt Creator erlauben es, ein externes Plugin zu installieren und das User-Interface der Entwicklungsumgebung ist es möglich, auf die Funktionen von Mercurial zuzugreifen.

12.3.1 Terminal

Über die Kommandozeile kann man über folgende Befehle auf Mercurial zugreifen:

- **Clonen**: hg clone <URL>
- **Dateien hinzufügen**: hg add <Datei>
- **Änderung**: hg revert <Datei>
- **Änderungen bestätigen**: hg commit -m <Änderungstext>
- **Repository auf den aktuellen Stand bringen**: hg update
- **Branch mischen**: hg merge
- **Versionsgeschichte des Repository erkunden**: hg log -v

Wie beim bereits im Unterricht behandelten GIT, muss man bei der Bestätigung von Änderungen eine Beschreibung der Änderung anfügen. Wenn das automatische Mischen nicht gelingt, muss man manuell die Konflikte lösen.

12.4 Anwendungs-Beispiele

Einige namhafte Firmen haben und bei bekannten Projekten wurde Mercurial eingesetzt. Dazu gehören Facebook, Mozilla (Firefox, Thunderbird), SourceForge, Google Inc. (Google Chrome, Google Code), Atlassian (Bitbucket), Microsoft (Codeplex), Oracle (OpenJDK), Xen, NetBeans IDE, Python und ClearCanvas.

12.5 Fazit

Zusammenfassend kann man sagen, dass Mercurial ähnlich dem Versionskontrollsystem Git ist, es gibt keine signifikante Unterschiede. Der größte Unterschied zwischen beiden Systemen ist der Performanceverlust durch den Python-Interpreter, weshalb es besser ist, Git als Versionskontrollsystem zu verwenden.

13 Bitucket

Bitbucket ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte, der die Versionsverwaltungssysteme Git und Mercurial unterstützt. Der Dienst wurde ursprünglich als reines Mercurial-System von Jesper Nøhr entwickelt und 2010 von dem australischen Unternehmen Atlassian gekauft und am 3. Oktober 2011 umUnterstützung für Git erweitert.

13.1 Eigenschaften

Im Gegensatz zu anderen Open-Source-Hostern wie SourceForge ist auf Bitbucket nicht das Projekt als Sammlung von Quelltext zentral, sondern der Nutzer mit seinen Repositorys (Verzeichnissen, die vom jeweiligen VCS kontrolliert werden). Gleichzeitig wird das Erstellen (Branchen) und Wiedervereinigen (Mergen) von Abspaltungen (Forks) besonders propagiert. Forks dienen weiterhin dazu, einfach bei anderen Projekten mitentwickeln zu können. Benutzer werden dazu ermutigt, in Teams zusammenzuarbeiten. Anders als bei anderen VCS-Diensten wie GitHub stellt Bitbucket grundsätzlich jedem Benutzer eine unbegrenzte Anzahl privater, also nicht öffentlich sichtbarer Repositorys kostenlos zur Verfügung, außerdem können Teams aus bis zu fünf Mitgliedern ein kostenfreies Abonnement abschließen. Größere Teams müssen einen monatlichen Betrag entrichten.

13.2 Verwendung

2014 arbeiteten über 330.000 Teams aus über 2,5 Millionen Entwicklern mit Bitbucket, 200 Terabyte Code wurden gehostet. Zu den Unternehmen, die Gebrauch von Bitbucket machen, zählen neben Atlassian: DHL PayPal Tesla Motors The New York Times Auch die Open-Source-Projekte Eigen und OGRE sind bei Bitbucket gehostet.