Seminar-Ausarbeitung

Werkzeuge für eine gemeinsame Softwareentwicklung

-

„Projektgruppe Modulares Enterprise Architecture Management System“

Department für Informatik - Carl von Ossietzky Universität Oldenburg

Vorgelegt von:

Christian Zillmann

Betreuer:

Dipl. Inform. Matthias Postina

30. Oktober 2007

Inhaltsverzeichnis

Zusammenfassung

Diese Arbeit, die im Rahmen der Projektgruppe „Modulares Enterprise Architecture Management System“ an der Carl von Ossietzky Universität Oldenburg entsteht, beschäftigt sich mit dem Thema „Werkzeuge für eine gemeinsame Softwareentwicklung“. Dabei wird auf die Notwendigkeit und den Nutzen von Werkzeugen bei der Softwareentwicklung eingegangen. Weiterhin werden die verschiedenen Arten von Werkzeugen vorgestellt. Es wird konkret auf Werkzeuge eingegangen, die für diese Projektgruppe zur Unterstützung der Entwicklung im Team interessant sein könnten.

1 Einleitung

1.1 Motivation und Problemstellung

Große Softwareprojekte werden in der Regel im Team entwickelt, bei denen Mitarbeiter an verschiedenen Orten und zu verschiedenen Zeiten an dem Projekt arbeiten. Nach Altmann u.Pomberger (1999) erfolgt die Zusammenarbeit der Mitarbeiter eines Softwareprojektes durch die zeitliche und räumliche Aufteilung des Projektes asynchron. Demnach sind eine regelmäßige Synchronisation und die Möglichkeit der Entwickler aktuelle Zustände und Ereignisse zu erfassen, für eine effiziente Zusammenarbeit unerläßlich. Zehler (2004) schreibt auf Seite 1 in Zeile 12-17: „Die Entwicklung und Wartung immer komplexer werdender Softwaresysteme schließen die Zusammenarbeit vieler Menschen mit ein, wodurch das Programmieren, Verstehen und Modifizieren der Software immer schwieriger wird. Der Einsatz von Werkzeugen im Software-Entwicklungsprozess spielt mittlerweile eine große Rolle, um die Entwickler bei der Bewältigung dieser Aufgaben zu unterstützen.“ Auch die Software, welche im Laufe der Projektgruppe „Modulares Enterprise Architektur Management System“ entstehen soll, wird in einem größeren Team entwickelt. Auch hier ist zu erwarten, dass nicht immer zur gleichen Zeit und am gleichen Ort an dem Projekt gearbeitet werden kann. Daher ist es wichtig, dass jedes Mitglied sich zu jeder Zeit über den aktuellen Stand der Entwicklung informieren kann. Weiterhin werden Werkzeuge benötigt, welche die Kommunikation zwischen den einzelnen Teammitgliedern unterstützen. Dazu zählen Modellierungstools genauso wie z.B. eine Plattform zur direkten Kommunikation zwischen den Mitgliedern oder zur Dokumentation des Projektverlaufs.

1.2 Zielsetzung

Ziel dieser Arbeit ist es eine praxisnahe Vorstellung von Entwicklungswerkzeugen zu geben, welche im Rahmen der Projektgruppe zur Anwendung kommen könnten, um den Entwicklungsprozess im Team zu unterstützen. Dabei soll es sich nicht um einen alternativen Vergleich von Werkzeugen handeln, sondern es soll aus den verschiedenen Bereichen von Entwicklungswerkzeugen, auf die im Laufe der Arbeit noch eingegangen wird, jeweils ein Werkzeug näher vorgestellt und betrachtet werden. Werkzeuge zum Projektmanagement, Dokumentation oder Review und Test werden in dieser Ausarbeitung bewusst nicht näher behandelt, da sie bereits Thema von anderen Ausarbeitungen sind.

1.3 Aufbau der Arbeit

Bei diesem Kapitel handelt es sich um die Einführung mit der Motivation und Problemstellung, der Zielsetzung und dem Aufbau dieser Arbeit. Das zweite Kapitel beschäftigt

sich dagegen vorwiegend mit den Grundlagen, die in dieser Arbeit benötigt werden. Dabei wird auf die verschiedenen Arten von Entwicklungswerkzeugen und ihre Verwendung eingegangen. Im dritten Kapitel werden dann repräsentativ Werkzeuge aus den verschiedenen Kategorien vorgestellt. Die Arbeit schließt dann mit einem Fazit ab.

2 Grundlagen

In diesem Kapitel soll zuerst kurz der Begriff Werkzeug erörtert werden und anschließend ein Blick auf die Vor- und Nachteile der Verwendung von Werkzeugen bei der Softwareentwicklung geworfen werden. Im Anschluss daran, werden verschiedene Kategorien von Softwareentwicklungswerkzeugen, die der Entwicklung im Team dienlich sein könnten, vorgestellt. Hierzu zählen im Rahmen dieser Ausarbeitung die Kategorien Versionskontrolle, Modellierungstools, Requirements-Engineering, Entwicklungsumgebungen sowie Bugtracking.

2.1 Begriffserklärung

Forbrig (2003) schreibt auf Seite 22 in Zeile 1-5: „Schon seit langem existiert die Wunschvorstellung, den Softwareentwicklungsprozess Werkzeug unterstützt zu betreiben. Die Klasse von Werkzeugen (Tools), die hierbei eine Rolle spielen, sind die sogenannten CASE-Werkzeuge. Computer Aided Software Engineering (CASE) verwendet eine Klasse von Werkzeugen die, die ingenieurmäßige Erstellung von Software unterstützt."

- Nach Rechenberg (1985) versteht man unter Werkzeugen im Kontext der Software-Entwicklung „Programme, die die Herstellung, Prüfung, Wartung und Dokumentation von Programmen vereinfachen, beschleunigen oder in ihrer Qualität verbessern."
- Nach Glinz (2007) ist ein Werkzeug für die Softwareentwicklung, ein rechnergestütztes Hilfsmittel für die Entwicklung von Software.

Die grundlegenden Ziele der Verwendung von Werkzeugen bei der Softwareentwicklung sind nach Sauer (1998):

- Erhöhung der Produktivität
- Erhöhung der Qualität
- Erleichterung des Softwaremanagements

2.2 Vor- und Nachteile der Verwendung von Entwicklungswerkzeugen

Partsch (2004) zählt zu den **Vorteilen** der Verwendung von Entwicklungswerkzeugen, folgende Punkte auf:

- Die Produktivität erhöht sich, da wiederholende Arbeitsgänge automatisiert werden können.
- Der Entwickler kann sich auf die inhaltlichen Aspekte konzentrieren.
- Der Werkzeugeinsatz erzeugt standardisierte Daten und Dokumente, welche die Kommunikation parallel arbeitender Entwickler erleichtern und die Einheitlichkeit und Konsistenz der Teilprodukte und des Endprodukts verbessern.

- Mit der Auswahl eines Werkzeuges legt man sich in der Regel auf ein Vorgehensmodell und spezielle Methoden fest. Der Einsatz des Werkzeuges garantiert weitgehend die korrekte Anwendung der unterstützten Verfahren und verbessert somit die Qualität des Entwicklungsprozesses und des Softwareproduktes.
- Durch den Einsatz von Test- und Debugging-Werkzeugen lässt sich die Fehlerhäufigkeit senken, was eine Verbesserung der Produkt-Qualität zur Folge hat.

Bei den **Nachteilen** zählt Partsch (2004) dagegen folgende Punkte auf:

- Die Existenz eines Werkzeugs und die Erfahrung im Umgang mit ihm verführen leicht dazu, es auch für Anwendungen einzusetzen, für die es methodisch nicht geeignet ist.
- Die meisten negativen Erfahrungen beim Einsatz von Werkzeugen im Software-Entwicklungsprozess lassen sich auf mangelhafte Auseinandersetzung mit dem Werkzeug zurückführen.
- Es existiert gegenwärtig kein Werkzeug, welches sämtliche Phasen der Software-Entwicklung gleichermaßen und integriert unterstützt.
- Ein großes Handicap bei der Zusammenarbeit mehrerer unterschiedlicher Werkzeuge in einem Entwicklungsprozess liegt oftmals darin, dass die Schnittstellen sowie die Modelle der einzelnen Werkzeuge nicht miteinander vereinbart und interoperabel sind.

Zusammengefasst lässt sich demnach sagen, dass sich für die Entwickler damit letztendlich der Vorteil ergibt, dass sie von Trivialarbeiten entlastet werden und eine bessere Übersicht über ihre Leistung erhalten. Dem entgegengesetzt ist natürlich immer der Einarbeitungsaufwand der speziellen Werkzeuge.

2.3 Kategorien von Softwarewerkzeugen

In diesem Abschnitt werden nun einige Kategorien von Entwicklungswerkzeugen vorgestellt, die im Rahmen der Projektgruppe „Modulares Enterprise Architecture Management“ zur Unterstützung der gemeinsamen Entwicklung hilfreich sein könnten.

2.3.1 Versionskontrolle

Nach Baerisch (2005) sind Versionskontrollwerkzeuge heutzutage, die am meisten verbreiteten Werkzeuge bei der Entwicklung von Software. Versionskontrollwerkzeuge erlauben die Verwaltung von unterschiedlichen Revisionen einer Datei, sie koordinieren und unterstützen die Entwicklung eines Teams an einem gemeinsamen Datenbestand. Typischerweise wird mit diesen Werkzeugen die Versionierung von Quelltexten verwaltet. Aber nach Baerisch (2005) kann man dieses Werkzeug für alle Arten von Textdateien verwenden. Dabei werden Änderungen an den Datenbeständen erfasst und protokolliert. Es soll damit sichergestellt sein, dass alle Benutzer die Änderungen nachvollziehen können und immer auf dem aktuellsten Stand der Entwicklung sind. Ein weiterer Vorteil ist, dass jederzeit zu alten Versionen gewechselt werden kann, z.B. wenn eine Änderung fehlerhaft war und man diese gerne rückgängig machen will. Dem Benutzer fallen nach Baerisch (2005) dabei lediglich die Aufgabe zu, zu entscheiden, wann der aktuelle Zustand einer Datei als neue Version betrachtet wird. In der Regel ist es möglich neue Versionen mit Kommentaren zu versehen. Baerisch (2005) schreibt auf Seite 7 in Zeile 11-13:„Dies gestattet es auch lange nach einer Veränderung festzustellen, mit welcher Intention, wann und von wem diese vorgenommen

wurde.“ Zusammengefasst kann man demnach sagen, dass Versionsverwaltungssysteme dem Benutzer zwei grundlegende Funktionen anbieten:

- eine Dokumentationsfunktion um Änderungen nachvollziehen zu können
- eine Wiederherstellungsfunktion, die es ermöglicht unerwünschte Änderungen rückgängig zu machen

Da ein Versionsverwaltungswerkzeug nicht nur von einer Person alleine verwendet werden kann, sondern auch von einer Gruppe von Benutzern, die wie auch in dieser Projektgruppe geplant, gemeinsam an einer Software arbeiten, kann es natürlich zu Konflikten kommen, wenn mehrere Personen gleichzeitig Änderungen an einer Datei vornehmen. So könnte man sich vorstellen, dass die Änderungen des einen Nutzers, sofort wieder vom nächsten Nutzer überschrieben wird, ohne dass dieser von der ersten Änderung erfährt. Ein Versionsverwaltungswerkzeug muss also mit solchen Konflikten umgehen können. Baerisch (2005) nennt dafür folgende Verfahren:

- das generelle Ausschließen von potentiellen Konfliktfällen
- den Versuch, Konfliktfälle automatisch zu beheben
- die Unterstützung bei der Auflösung von Konflikten durch die Benutzer

Ein weiteres Feature von Versionsverwaltungssystemen sind die sogenannten Branches. Man spricht von Branches, wenn sich die Entwicklung einer Software an einem Punkt in zwei Richtungen entwickelt, also aus einer Version, zwei Versionen werden, vergleichbar mit einer Astgabel. Baerisch (2005) nennt dafür als Beispiel, dass bei der ersten Version nur noch Fehler korrigiert und kleine Änderungen vorgenommen werden und bei der zweiten Version grundlegende Änderungen vorgenommen werden, wie zum Beispiel neue Funktionen. Der Benutzer hat in diesem Fall beim Hinzufügen einer Änderung die Möglichkeit, die Änderung einem der beiden Zweige zuzuordnen.

2.3.2 Modellierung

Nach Floyd u.Klischewski (1998) kommt der Modellierung in der Informatik eine zentrale Stellung zu. In der Informatik kommt den Modellen die Aufgabe zu, eine Art „Fenster zur Wirklichkeit“ zu sein. Modelle sollen Zusammenhänge aus der Wirklichkeit in Datenstrukturen abbilden. Typische Modellierungssprachen bzw. Techniken sind z.B. ereignisgesteuerte Prozessketten (EPK). Ereignisgesteuerte Prozessketten dienen der statischen Darstellung von Prozessen. Eine andere Form sind die sogenannten Entity-Relationship-Modelle, die im Rahmen der Datenmodellierung einen Ausschnitt aus der realen Welt beschreiben. ER-Diagramme kommen zum Beispiel bei der Anwendungsentwicklung zur Verständigung zwischen Entwickler und Anwender zur Anwendung aber auch bei der Implementierung von Software, zur Modellierung des Datenbankdesigns. Zur Modellierung von objektorientierter Software hat sich nach Forbig (2006) die Unified Modeling Language (UML) als Standard durchgesetzt. Die erste Version von UML wurde ca. 1990 von der Object Management Group entwickelt. Im Jahre 2005 entstand eine grundlegend überarbeitete Version von UML, die als UML2 bezeichnet wurde. UML2 kennt nach Forbig (2006) zur Zeit sechs Strukturdiagramme:

- Klassendiagramm
- Kompositionsstrukturdiagramm

- Komponentendiagramm
- Verteilungsdiagramm
- Objektdiagramm
- Paketdiagramm

sowie sieben Verhaltensdiagramme:

- Anwendungsfalldiagramm
- Aktivitätsdiagramm
- Sequenzdiagramm
- Kommunikationsdiagramm
- Interaktionsübersichtsdiagramm
- Zeitverlaufsdiagramm
- Zustandsdiagramm

Modellierung ist auch bei der gemeinsamen Softwareentwicklung zur Kommunikation zwischen den einzelnen Teammitgliedern ein wichtiges Werkzeug. Zur Modellierung von z.B. EPKs, ER-Diagrammen oder UML2-Diagrammen existieren zahlreiche Tools, von denen jeweils eins im nächsten Kapitel Werkzeuge näher vorgestellt wird.

2.3.3 Requirements-Engineering

Requirements-Engineering beschreibt das ingenieurmäßige Vorgehen bei der Erhebung von Anforderungen und ist damit ein wesentlicher Bestandteil bei der Softwareentwicklung. Die Anforderungen, die ein Anwender bzw. ein Auftraggeber an die Software bzw. an das zu entwickelnde System hat, müssen im Vorfeld ermittelt werden. Diese Anforderungen werden dann in dem sogenannten Anforderungskatalog festgehalten. Nach Partsch (2004) ist Requirements-Engineering im weiteren Sinn eine Disziplin, die zur systematischen Entwicklung einer vollständigen, konsistenten und eindeutigen Spezifikation, in der beschrieben wird, was ein softwaregestütztes Produkt tun soll (aber nicht wie), und die als Grundlage für Vereinbarungen zwischen allen Betroffenen dienen kann. Weiterhin schreibt Partsch (2004), dass der Begriff Requirements-Engineering in dem etwas engerem Sinn dazu verwendet wird, die Tätigkeiten am Beginn eines Projektes zu charakterisieren. Demnach müssen in dieser anfänglichen Analyse- und Definitionsphase im traditionellen Phasenmodell die Anforderungen an das zu entwickelnde Produkt ermittelt, festgelegt, beschrieben und analysiert werden. Da nach Partsch (2004) die Anforderungen an ein neues Produkt i.d.R. zuerst vage, verschwommen, mehrdeutig, unzusammenhängend, unvollständig und sogar widersprüchlich sind, ist es daher Ziel des Definitionsprozesses ausgehend von den anfänglich verfügbaren Information die Anforderungen an das neue Produkt präzise, eindeutig, konsistent und vollständig zu beschreiben. Dies dient demnach als Grundlage für die spätere Systementwicklung. Der Einsatz von RE-Werkzeugen hat zum Ziel, Anforderungen an laufende oder gewünschte Software-Systeme systematisch in den Griff zu bekommen. Ein Werkzeug zur Unterstützung der Anforderungsanalyse sollte laut Partsch (2004) folgende Funktionen beinhalten:

- Beschreibung des Requirement über Formulare mit den geeigneten Features einer Textverarbeitung (z.B. MS WORD)

- Darstellung der Anforderungen als Use Case, auch über ein Use-Case-Diagramm in einem grafisch orientierten Analyse-Tool
- Möglichkeit der Verfeinerung der Anforderung über ein Aktivitätsdiagramm in der Unified Modeling Language (UML)
- Darstellung der Anforderungen als Beschreibung von Use Cases, über Use-Case-Diagramme und in der weiteren Verfeinerung über Aktivitätsdiagramme in grafisch orientierten Analyse-Tools erfolgen; hier ist die UML das geeignete Mittel für die grafische Darstellung in der Anforderungsanalyse
- Klassifizierung der Anforderung mit entsprechenden Selektionsmöglichkeiten
- Priorisierung der Anforderung mit entsprechenden Selektionsmöglichkeiten
- Verwaltung und Darstellung der strukturellen Abhängigkeiten zwischen den Anforderungen
- Zuordnung der Anforderungen zu einem Projektportfolio
- Verknüpfung der Anforderung mit den in einem CASE-Tool (Computer Aided Software Engineering Tool) verwalteten Objekten der Anwendungsentwicklung

Auch hierzu wird im nächsten Kapitel ein Werkzeug vorgestellt, welches den Prozess der Anforderungsanalyse unterstützen soll.

2.3.4 Integrierte Entwicklungsumgebung

Eine integrierte Entwicklungsumgebung ist ein Programm, welches die Entwicklung von Software unterstützen soll. Der Begriff „integrierte Entwicklungsumgebung“ wird oft mit IDE abgekürzt, was das Kürzel für die englische Bezeichnung „integrated development environment“ ist. IDEs fassen mehrere Programme zusammen und beinhalten in der Regel folgende Komponenten:

- Compiler und/oder Interpreter
- Linker
- Debugger
- Texteditor

Größere IDEs beinhalten darüber hinaus noch viele weitere Features wie z.B. eine Quelltextformatierungsfunktion oder einen „Content-Assistent“ bzw. „Autocomplete-Funktion“, die dem Programmierer Schreibarbeit abnehmen soll.

2.3.5 Bugtracking

Sogenannte Bug-Tracker oder auch Bugtracking-Tools sind nach Suhr (2007) in der Regel webbasierte Anwendungen, die als Aufgabe die Verwaltung von verschiedenartigen Aufgaben haben. Dazu zählt Suhr (2007) zum Beispiel die Fehlerbeseitigung oder Änderungswünsche. Dabei ist es möglich neue Aufgaben zu erstellen, diesen den Teammitgliedern zuzuordnen, den Verlauf einer Aufgabe zu beobachten oder die Erstellung komplexer Berichte, die den Fortschritt des Projektes bzw. der verwalteten Aufgaben dokumentieren. Die Realisierung dieses Werkzeuges als Webanwendung bietet sich demnach an, da jedes Teammitglied außer einem herkömmlichen Internet-Browser keine zusätzliche Software benötigt.

3 Werkzeuge

Dieses Kapitel stellt nun einzelne Werkzeuge zu den eben näher betrachteten Kategorien vor. Es wird darauf geachtet, dass die Werkzeuge im Idealfall Open-Source und für möglichst viele Betriebssysteme verfügbar sind. Leider kann dies nicht bei allen Werkzeugen eingehalten werden, da einige Werkzeuge nur für Windows bzw. nicht frei verfügbar sind.

3.1 Das Versionskontrollwerkzeug Subversion

Subversion[1] (abgk.SVN) wurde entwickelt, um das bis dahin klassische Versionskontrollsystem CVS zu ersetzen. SVN korrigiert einige Schwächen von CVS. So ist der Umgang mit Binärdaten optimiert wurden, die Arbeit mit Verzeichnissen wurde stark verbessert und auch das Umbenennen von Dateien ist unter SVN zuverlässiger als unter CVS. Um mit SVN zu arbeiten benötigt man einen Server, auf dem der SVN-Server installiert wird und auf dem die Repositorys liegen. Das Respository beinhalt immer den aktuellen Stand eines Projektes. Dementsprechend gibt es pro Projekt genau ein Respository. Auf den Clients muss ein SVN-Client installiert werden. Mit Hilfe dessen kann man sich dann den Inhalt des Repositorys auf den Client laden. Diese Kopie des Respositorys nennt man Arbeitskopie. Arbeitskopien kann es beliebig viele geben und an diesen werden dann Änderungen vorgenommen und wieder ins Repository geladen (committen). Als SVN-Client wird hier der Client RapidSVN vorgestellt. RapidSVN ist Open Source und für alle gängigen Betriebssysteme verfügbar. Die nächsten Kapitel zeigen nun, wie man Repsositorys auf dem Server anlegt und wie man mit Hilfe von RapidSVN mit diesen arbeiten kann.

3.1.1 Einrichten eines Repositorys

Ist der SVN-Server installiert, steht auf dem Server auf der Kommandozeile das Werkzeug svnadmin bereit. Dies kann man testen, indem man einfach auf der Kommandozeile *svnadmin* eintippt und Enter drückt. Es sollte sich nun das Subversion-Werkzeug melden. Um nun ein Repository für ein neues Projekt anzulegen, wechselt man in den Ordner, in dem man das Repository anlegen will. In diesem Beispiel legen wir ein Repository mit dem Namen *PGEAM* im Verzeichnis */usr/local/svn/* an, mit Hilfe des *svnadmin* Befehls:

```
svnadmin␣create␣pgeam
```

Wenn der Befehl ohne Fehler ausgeführt wurde, befindet sich nun ein Ordner *pgeam* im Verzeichnis */usr/local/svn/*. Im Verzeichnis *pgeam* befindet sich nun ein Ordner *conf*, in dem sämtliche Konfigurationsdateien vorhanden sind. Mit Hilfe dieser können dann auch Benutzer angelegt werden und diesen Rechten zugewiesen werden, z.B. ob sie Lese- und Schreib-Zugriff haben oder nur Leserechte. Da die Konfiguration und Einrichtung des Repositorys i.d.R. vom Administrator übernommen wird, wird hier nicht näher drauf eingegangen. Für weiterführende Informationen speziell zur Einrichtung von Nutzern und deren Rechte, sei auf das *svnbook* verwiesen, welches unter `http://svnbook.red-bean.com/` eingesehen werden kann.

[1] `http://subversion.tigris.org/`

3.1.2 Konfiguration von RapidSVN

RapidSVN ist für Windows, Linux und MAC OS auf der Seite `http://rapidsvn.tigris.org/` downloadbar. Da eine volle Funktionsbeschreibung dem Umfang dieser Arbeit sprengen würde, kann hier nur ein kleiner Einblick in die grundlegenden Funktionen gegeben werden. Für weiterführende Informationen sei hier auf die Online-Hilfe zu RapidSVN verwiesen, zu finden unter `http://www.rapidsvn.org/index.php/OnlineHelp:Contents`. Nach der Installation muss RadpidSVN konfiguriert werden, um Zugriff auf das Repository zu haben. Dafür startet man RapidSVN und klickt anschließend im Menu auf Repository → Auschecken. Das sich daraufhin öffnende Fenster füllt man wie in Abbildung 1 auf Seite 9 zu sehen aus. Die URL des Repositorys ist i.d.R. in der Form svn://ipdesservers/usr/local/svn/pgeam. Beim Zielverzeichnis stellt man einen lokalen Ordner ein, in dem die Arbeitskopie geladen werden soll. Klickt man nun auf OK, wird man eventuell noch nach seinem Benutzernamen

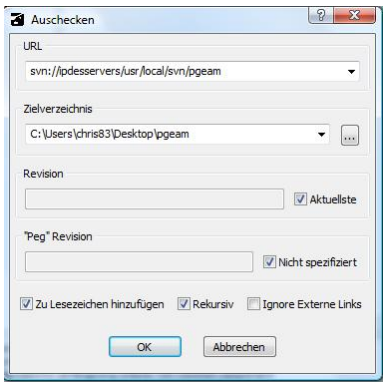

Abbildung 1: rapidsvn

und Passwort gefragt, hier gibt man dann die Benutzerdaten ein, die man vom Administrator des SVN-Servers bekommen hat. Anschließend wird die Arbeitskopie in das lokale Verzeichnis kopiert.

3.1.3 Hinzufügen, Ändern und Löschen von Dateien

In seiner lokalen Arbeitskopie auf seinem eigenen Rechner kann man nun beliebig Dateien erstellen oder auch bereits existierende Dateien ändern. RapidSVN erkennt diese Änderungen automatisch und zeigt neue Dokumente durch ein Fragezeichen-Symbol an und geänderte Dokumente durch ein rotes Symbol. Neue Dateien müssen zuerst durch einen Rechtsklick auf die Datei und dem Punkt Hinzufügen, im Kontextmenu, hinzugefügt werden. Möchte man ganze Ordner inklusive Unterordnern und Dateien hinzufügen, kann man alternativ auch den Punkt „rekursiv Hinzufügen“ wählen. Hat man alle neuen Dateien hinzugefügt, muss man die Änderungen bzw. neuen Dateien ins Repository laden. Dies bezeichnet man als comitten. Dafür wählen wir im Menu Bearbeiten den Punkt Committen aus. Anschließend wird man aufgefordert, einen kurzen Kommentar zu schreiben, der beschreibt, was genau geändert wurde. Von dieser Funktion sollte man unbedingt gebrauch machen, da es sonst schwer wird, die Änderungen nach längerer Zeit nachzuvollziehen. Mit einem Klick auf OK werden die Änderungen ins Repository übernommen.

3.1.4 Update der Arbeitskopie

Bevor man Änderungen an seiner lokalen Arbeitskopie vornimmt, sollte man immer sicherstellen, dass man auf der aktuellen Version arbeitet. Dafür sollte man im Vorfeld immer ein Update seines Arbeitsverzeichnisses durchführen. Um ein Update durchzuführen, klickt man in RapidSVN auf Bearbeiten und dann auf Aktualisieren. Nun kann man sicher sein, dass man auf der aktuellsten Version arbeitet.

3.1.5 Änderungen rückgängig machen

Hat man an einer Datei lokal eine Änderung vorgenommen und möchte dies rückgängig machen, kann man in RapidSVN das Kontextmenu der entsprechenden Datei mittels eines Rechtsklick öffnen und dort Rückgängig auswählen. Die Datei wird dann wieder in den Zustand gebracht, die sie im Repository hat.

3.2 Modellierung mit Microsoft Visio 2007 und Dia

Im Open-Source Bereich ist es schwer ein brauchbares Modellierungstool zu finden. Es gibt zwar einige UML-Tools, diese haben aber oft viele Schwächen, so dass die Meisten z.B. kein UML2 oder nur Klassendiagramme unterstützen.

3.2.1 Dia vs. Visio

Mit Dia findet sich ein Open-Source Produkt, was verschiedene Modellierungen erlaubt, von UML-Diagrammen, ER-Diagrammen, Geschäftsprozessen bis zu einfachen Zeichnungen. Dia gehört eigentlich zum Gnome-Projekt kann aber durch die Portierung des GIMP-Toolkits ebenfalls unter Windows betrieben werden. Dia ist vergleichbar mit dem proprietären Visio von Microsoft, ist aber Visio im Bereich Bedienungsfreundlichkeit unterlegen, bietet weniger vorgefertigte Diagrammtypen und unterstützt standardmäßig auch kein UML2. Visio ist zwar kein Open-Source Produkt, kann aber von Informatik-Studenten der Universität Oldenburg kostenlos über das MSDN-AA-Programm[2] von Microsoft bezogen werden. Informatik-Studenten der Universität Oldenburg können sich über `http://wwwphp.informatik.uni-oldenburg.de/elms/index.php` für das MSDN-AA-Programm registrieren und so Visio kostenlos beziehen.

3.2.2 Diagrammarten von Visio

Visio bietet im Softwareentwicklungsbereich u.a. folgende Arten von Diagrammtypen an:

- UML-Modelldiagramm
- Datenbankmodelldiagramm
- Programmstrukturdiagramm
- Datenflussmodelldiagramm
- ORM-Diagramm (Object Role Modeling - Diagramm)
- ROOM-Diagramme (erläutern wie ein System funktioniert)
- Konzeptionelle Website - Diagramme

[2] `http://www.e-academy.com/`

- COM und OLE - Diagramme
- Unternehmensanwendungs-Diagramme

Da Visio gegenüber Dia einige Vorteile hat und ebenfalls kostenlos bezogen werden kann, bietet es sich an, Visio gegenüber Dia den Vorzug zu geben. Für andere Betriebssysteme als Microsoft Windows ist Dia aber eine brauchbare Alternative. Hilfe und eine Anleitung zu Visio, sowie nähere Informationen zu den einzelnen Diagrammtypen findet man unter `http://office.microsoft.com/de-de/visio/CH010191591031.aspx`.

3.3 Requirements-Engineering mit Rational RequisitePro

Im Bereich Requirements-Engineering lässt sich leider im Open-Source Bereich kein anspruchsvolles Produkt finden. Die Firma IBM besitzt allerdings ein Produkt namens Rational RequisitePro, welches über das IBM-Scholars-Program[3] von Studenten kostenlos bezogen werden kann, allerdings muss sich die Hochschule erst für dieses Programm registrieren. RequisitePro bietet eine Unterstützung bei der Erstellung von Anforderungsdokumenten durch Word-Vorlagen. So lassen sich u.a. Produktanforderungen, Softwareanforderungen, zusätzliche Anforderungen oder Test-Entwürfe verwalten und erstellen. Es handelt sich dabei um eine Integration von Word mit einer Anforderungsdatenbank. Es ist z.B. eine Anforderungsverfolgung und -verknüpfung mit verschiedenen Sichten wie z.B. einer Matrix- oder Baum-Sicht möglich. Hierarische Anforderungen sind möglich, Funktionen zum Änderungsmanagement sind vorhanden und die Gruppenarbeit bzw. die Arbeit im Team wird explizit unterstützt.

Abbildung 2: Integration in Word
Quelle [4]

3.4 Die Entwicklungsumgebung Eclipse

Als Entwicklungsumgebung soll hier Eclipse[6] vorgestellt werden. Eclipse ist eine sehr umfangreiche Entwicklungsumgebung, die Open-Source ist und damit kostenlos bezogen wer-

[3] `http://www-304.ibm.com/jct09002c/university/scholars/ur/index.html`
[4] Quelle: `http://web.inf.tu-dresden.de/TU/Informatik/ST2/ST/lv_WS98-99/st2/uebung/reqmanagementtools.pdf`
[5] Quelle: `http://web.inf.tu-dresden.de/TU/Informatik/ST2/ST/lv_WS98-99/st2/uebung/reqmanagementtools.pdf`
[6] `http://www.eclipse.org`

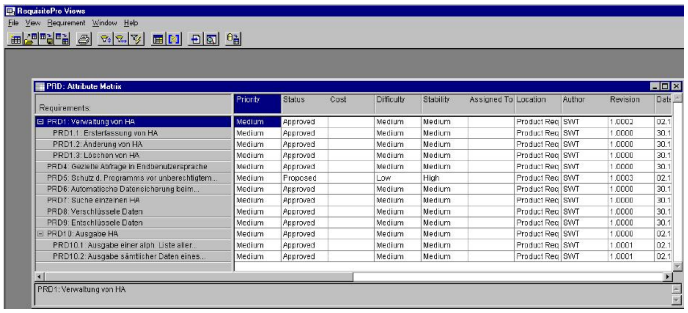

Abbildung 3: Anforderungsdatenbank
Quelle [5]

den kann. Eclipse hat sich in sehr kurzer Zeit einen Namen gemacht, eine sehr leistungsfähige Entwicklungsumgebung für Java zu sein. Eclipse ist aber laut eclipse.org nicht nur eine Entwicklungsumgebung sondern ebenfalls eine universelle Werkzeugplattform. Das zugrundeliegende Ziel bei der Entwicklung von Eclipse war, eine plattformunabhängige, mehrere Programmiersprachen unterstützende Plattform zu schaffen, in der sich neue Werkzeuge leicht integrieren lassen. So gibt es heute für fast jede Programmiersprache ein Eclipse-Plugin und so gut wie jedes namenhafte Werkzeug lässt sich in Eclipse integrieren. So gibt es zum Beispiel ein Plugin für Subversion oder für das Bugtracking-System Mantis, welches im nächsten Abschnitt vorgestellt wird. Bei den Programmiersprachen wird Java oder PHP genauso wie z.B. Ruby, C++ oder Python unterstützt. Eclipse liegt zur Zeit in der Version 3.3.1 vor. Es empfiehlt sich aber, eventuell auf eine Version früher zurückzugreifen, da viele Plugins noch nicht mit der neuen Version kompatibel sind. Eclipse ist wie bereits erwähnt unter allen Betriebssystemen lauffähig, auf denen ein JDK[7] installiert ist. Die Abbildung 4 auf Seite 13 zeigt einen Screenshot von Eclipse. Der mit 1 gekennzeichnete Bereich, ist standardmäßig der Navigationsbereich, hier kann durch die Dateistruktur eines oder mehrerer Projekte navigiert werden. Natürlich kann dieser Bereich auch mit anderen Funktionen belegt werden, die dann über eine Tab-Schaltfläche aktiviert bzw. gewechselt werden können. In dem mit 2 gekennzeichneten Bereich befindet sich i.d.R. der Editor zum Bearbeiten der Dateien. Der Bereich, der mit 3 gekennzeichnet ist enthält standardmäßig z.B. eine Konsole oder eine History-Ansicht, kann aber ebenfalls frei mit den benötigen Sichten oder Tools belegt werden, genauso wie der Bereich 4. Entwickelt man mit verschiedenen Programmiersprachen so kann man unter Eclipse verschiedene Sichten wählen, die auf die jeweilige Programmiersprache optimiert sind. Weiterhin besitzt Eclipse neben einen integrierten Debugger noch eine menge weitere Funktionen, wie z.B. Quellcodeformatierung oder eine Autocomplete-Funktion.

3.5 Bugtracking mit Mantis

Bei Mantis handelt sich um ein freies und sehr populäres Bugtracking-System, welches mit PHP geschrieben wurde und eine MySQL-Datenbank oder PostgreSQL-Datenbank verwen-

[7] Java Development Kit
[8] Quelle: `http://www.sigs.de/publications/js/2003/cebit/weyerhaeuser_JS_cebit_03.pdf`

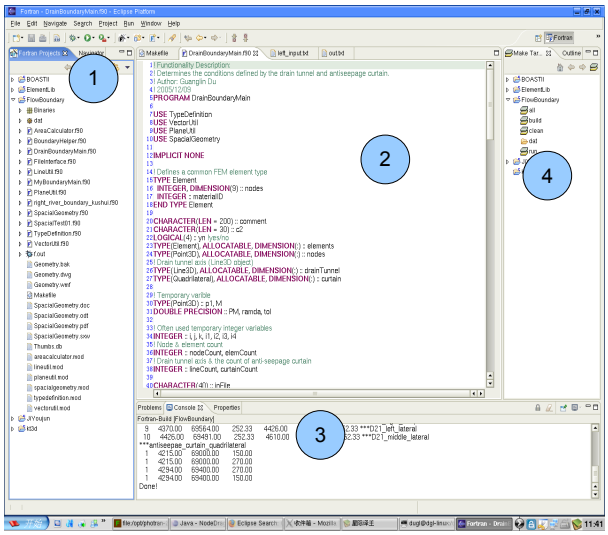

Abbildung 4: Eclipse
Quelle [8]

det. Mantis wird auf einem Webserver installiert und benötigt auf Client-Seite nur einen handelsüblichen Browser. Mantis ist zum einem kostenlos erhältlich, einfach zu installieren, komplett Web basiert, Plattform unabhängig, besitzt einfache Suchfunktionen, ist sehr gut dokumentiert und unterstützt nicht nur die Verwaltungs von Bugs sondern ebenfalls die Verwaltung von Aufgaben. Bugs und Aufgaben können verschiedenen Benutzern und verschiedenen Zuständen zugeordnet werden. Mantis unterscheidet zwischen sechs Rechtestufen von Benutzern:

- Betrachter
- Reporter
- Entwickler
- Tester
- Administrator
- Manager

Abbildung 5 auf Seite 14 zeigt den Lebenszyklus eines Bugs und wie dieser mit Mantis verwaltet wird. Die Abbildung 6 auf Seite 14 zeigt die Übersicht über ein Projekt, welches von Mantis verwaltet wird. Man erhählt einen schnellen Überblick, welche Probleme noch offen sind, welche kürzlich bearbeitet wurden, in welchem Stadium sie sich befinden und welche Bugs und Fehler einem selber zugeordnet sind.

[9] Quelle: `http://www.se.fh-heilbronn.de/~permantier/Vorlesungen/Labor%20SW-Projekte/Vorstellung%20Mantis%20BugTracking%20(Ambiel,%20Doneit).pdf`

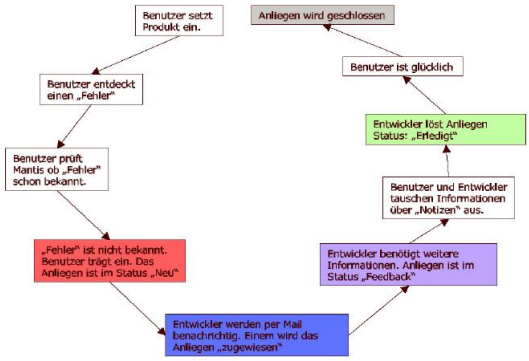

Abbildung 5: Lebenszyklus eines Bugs
Quelle [9]

Abbildung 6: Mantis-Übersicht

4 Fazit

Abschließend lässt sich sagen, dass eine Unterstützung durch Werkzeuge, bei der gemeinsamen Softwareentwicklung eine große Hilfe sein kann. Durch Werkzeuge lässt sich die Qualität des Produktes erhöhen und die Entwicklungszeit verkürzen. Dem Nutzen der Werkzeugunterstützung muss immer der Aufwand zur Auswahl eines geeigneten Werkzeuges und die Einarbeitungszeit in die entsprechenden Werkzeuge entgegengesetzt werden. Im Hinblick auf die Projektgruppe lässt sich sagen, dass ein Versionsverwaltungswerkzeug unbedingt erforderlich ist, da es sonst sehr aufwendig wäre, die Beiträge der verschiedenen Teilnehmer sinnvoll zu verwalten und zusammenzuführen. Als eine gemeinsame Entwicklungsumgebung ist mit Eclipse auch eine gute Lösung gefunden, da Eclipse beliebig mit Plugins erweitert werden kann und so auf die kommenden Anforderungen der Projektgruppe angepasst werden kann. Die Aufgaben und Fehlerverwaltung lässt sich sehr gut mit Mantis realisieren, da auf den Rechnern der Teilnehmer der Projektgruppe nur ein Browser benötigt wird und der

Funktionsumfang von Mantis meiner Einschätzung nach für die Projektgruppe ausreichend ist. Mit Mantis lassen sich dann die Fehler und Aufgaben dokumentieren und koordinieren, so dass jeder Teilnehmer sich schnell einen Überblick über seine Aufgaben schaffen kann und weiß, welche Fehler noch wie dringend korrigiert werden müssen. Auch ein Modellierungstool bringt einen großen Nutzen für die Projektgruppe mit. Diese Werkzeug unterstützt die Dokumentation der Software durch verschiedene Arten von Diagrammen. Diese Diagramme können neben der Dokumentationsfunktion auch noch Grundlage für die Kommunikation unter den Teammitgliedern sein. Kostenlose Tools sind z.B. das weiter oben vorgestellte Dia oder zum Beispiel Agro-UML[10]. Microsoft Visio bietet genau wie auch Dia die Möglichkeit neben UML viele weitere Diagrammarten zu modellieren und kann ebenfalls kostenlos bezogen werden. Im Bereich Requirements-Engineering ließ sich leider kein leistungsfähiges Werkzeug finden. IBM bietet neben vielen anderen guten Werkzeugen auch ein Requirements-Engineering Werkzeug, was aber nicht Open-Source ist. Dieses Werkzeug lässt sich aber wie bereits beschrieben, über ein Universitäts-Programm kostenlos beziehen, was aber eine Anmeldung der Universität erfordert. Meiner Einschätzung nach, ist aber das Fehlen eines Requirement-Engineering Werkzeug, im Gegensatz zu dem Fehlen eines der anderen Werkzeuge, verschmerzbar, da mit einer Standard-Textverarbeitung, einem Versionskontrollsystem und einem Aufgabenmanagement-System das Erstellen einer Anforderungsanalyse schon weitestgehend unterstützt wird. Ob eine weitergehende Unterstützung erforderlich ist, sollte im Team besprochen werden.

Literatur

Altmann, J.; Pomberger, G.: Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge. In: *Nüttgens, Markus (Hrsg.): Electronic Business Engineering/4. Internationale Tagung Wirtschaftsinformatik. Heidelberg, Germany: Physica-Verlag* (1999), S. 643–664

Baerisch, S.: *Versionskontrollsysteme in der Softwareentwicklung.* Informationszentrum Sozialwissenschaften, Bonn, 2005 `http://www.gesis.org/Publikationen/Berichte/IZ_Arbeitsberichte/pdf/ab_36.pdf`. – zuletzt besucht am 19.10.2007

Floyd, C.; Klischewski, R.: *Modellierung - ein Handgriff zur Wirklichkeit.* Universität Hamburg Facbbereich Informatik, AB Softwaretechnik, 1998 `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-9/Floyd.ps`. – zuletzt besucht am 19.10.2007

Forbig, P.: *Objektorientierte Softwareentwicklung mit UML.* Carl Hanser Verlag, München, 2006

Forbrig, P.: *Lehr- und Übungsbuch Softwareentwicklung.* Carl Hanser Verlag München, 2003

Glinz, M.: *Vorlesungsskript - Software Engineering.* Universität Zürich, 2007 `http://www2.staff.fh-vorarlberg.ac.at/~hv/Semester4/OOAD/glinz.pdf`. – zuletzt besucht am 19.10.2007

Partsch, H.: *Softwaretechnik, Skript zur Vorlesung.* Universität Ulm, 2004

[10] `http://argouml.tigris.org/`

Rechenberg, P.: Editorial zu Werkzeuge der Softwaretechnik. In: *Elektronische Rechenanlagen* 27 (1985)

Sauer, J.: *Software Engineering II, Skript zur Vorlesung*. Carl von Ossietzky Universität - Oldenburg, 1998 `http://www-is.informatik.uni-oldenburg.de/~sauer/lehre/skripte/se_2_98.pdf`. – zuletzt besucht am 19.10.2007

Suhr, G.: *Werkzeuge zur Aufgabenverwaltung in der dezentralen Systementwicklung*, Technische Universität Berlin, Diplomarbeit, Februar 2007. `http://user.cs.tu-berlin.de/~tron/das/issueman-suhr.pdf`. – zuletzt besucht am 19.10.2007

Zehler, T.: *Werkzeuge in der Software-Entwicklung*. VSEK Projekts FKZ 01 IS C39, 2004 `http://www.softwarekompetenz.de/servlet/is/21995/VSEK-Report_006D.pdf?command=downloadContent&filename=VSEK-Report_006D.pdf`. – 19.10.2007