

# Projektangebot

## Holistische Anzeige von Softwaresystemen

---

Version: 1.0

Erstellt am: 17.12.2010

Letzte Änderung: 27.01.2011

## Inhalt

Einleitung .....	4
Zweck, Abgrenzung .....	4
Motivation.....	4
Formale Grundlagen .....	4
Leistungen der Vertragspartner .....	4
Lieferumfang .....	4
Lizenzierung der Software.....	5
Leistungen des Auftraggebers.....	5
Berichtung des aktuellen Stands.....	5
Meilensteine .....	5
Geplanter Ablauf .....	5
Funktionaler Umfang .....	7
Umfang des Kernprojekts.....	7
Umfang des Erweiterungsprojekts.....	8
Entwicklungsrichtlinien .....	9
Konfigurationsmanagement .....	9
Design- und Programmierrichtlinien.....	9
Verwendete Software .....	9
Entwicklungsprozess .....	9
Aufteilung in zwei Projekte .....	9
Phasen im Kernprojekt.....	10
Spezifikation .....	10
Entwurf.....	10
Implementierung .....	10
Test.....	10
Phasen im Erweiterungsprojekt .....	10
Spezifikation .....	10
Entwurf.....	11
Implementierung .....	11
Test.....	11
Auslieferung .....	11
Dokumentationsplan.....	11
Prüfungen.....	11

Projektorganisation.....	11
Schnittstelle zum Auftraggeber.....	11
Schnittstellen zu anderen Projekten.....	12
Schlüsselpersonen.....	12
Entwicklungsplan .....	13
Kostenplan .....	13
Risiken und ihre Bewertung .....	13
Versionshistorie .....	17

## Einleitung

### Zweck, Abgrenzung

Gegenstand dieses Angebots ist der Projektplan, nach dem die Umsetzung des Projekts bei Annahme durch den Auftraggeber geplant ist. Dieser Projektplan legt die Vorgehensweisen fest, nach denen das Softwareprojekt „Holistische Anzeige von Softwaresystemen“ durchgeführt wird. Der Projektablauf wird aufgeteilt in zwei einzelne, aufeinanderfolgende Iterationen. Für die einzelnen Iterationsschritte werden die Phasen beschrieben und Meilensteine für die Phasenabschlüsse definiert. Außerdem wird festgehalten, welche externen Schnittstellen für die Iterationen relevant sind, welche Forderungen an die Schnittstellen und die Software gestellt wird und welche Maßnahmen zur Konfigurationsverwaltung ergriffen werden.

### Motivation

Oft sind Strukturen in Softwaresystemen oder Relationen auf Codebasis unklar. Es soll eine Software entwickelt werden, die die Möglichkeit bietet, vorhandene Softwaresysteme auf unterschiedliche Zusammenhänge, beispielsweise Vererbungsstruktur, Aufrufe etc., zu analysieren und zu visualisieren. Die Software soll in der Lage sein, Strukturen in vorliegendem Code (in Form eines Repositoriums) zu finden und in verschiedenen Graphen darzustellen.

### Formale Grundlagen

Die Software ist in der Programmiersprache C# im Umfeld .NET 4.0 zu entwickeln. Zur Erstellung der grafischen Benutzungsschnittstelle sind die Techniken der Windows Presentation Foundation (WPF) einzusetzen. Die Gewinnung der Daten für die darzustellenden Graphen soll per Reflection und/oder durch Microsofts Phoenix SDK, dessen Verwendung in der Expertengruppe entschieden wird, geschehen.

Der zu analysierende Code liegt in C# vor, jedoch soll die Software für andere Sprachen erweiterbar gestaltet werden.

Die von der Anwendung verwendete Sprache ist Englisch.

## Leistungen der Vertragspartner

### Lieferumfang

Zum Lieferumfang der Software gehören

- der Quellcode des Programms
- die ausführbare Anwendung
- die hinter der Benutzungsschnittstelle liegende Bibliothek zur Datengewinnung und Graphenrepräsentation
- die Spezifikation und der Entwurf der Software
- die Protokolle der Tests □ die Resultate des Vorprojekts □ das Handbuch.

## Lizenzierung der Software

Die Komponente zur Graphenvisualisierung wird unter der BSD-Lizenz geliefert. Die Lizenz der Gesamtsoftware hängt von den verwendeten Drittkomponenten ab.

## Leistungen des Auftraggebers

Nach Abschluss des Projekts wird die Software vom Auftraggeber gewartet. Außerdem stellt er die zur Ausführung nötige Hardware zur Verfügung.

Der Auftraggeber steht dem Entwicklerteam während des gesamten Projekts zur Klärung offener Fragen zur Verfügung. Er verfügt über Expertenwissen zu den zu implementierenden Graphentypen. Außerdem nimmt er an den Reviewsitzungen zur Spezifikation und auch zum Entwurf teil.

Der Auftraggeber stellt dem Entwicklerteam Beispieldaten in Form von Repositorien anderer Projekte zur Verfügung.

## Berichtung des aktuellen Stands

Der Auftraggeber erhält wöchentlich Berichte vom Projektleiter. In den Berichten sind jeweils der aktuelle Stand des Projekts und die Entwicklung seit dem letzten Bericht festgehalten.

## Meilensteine

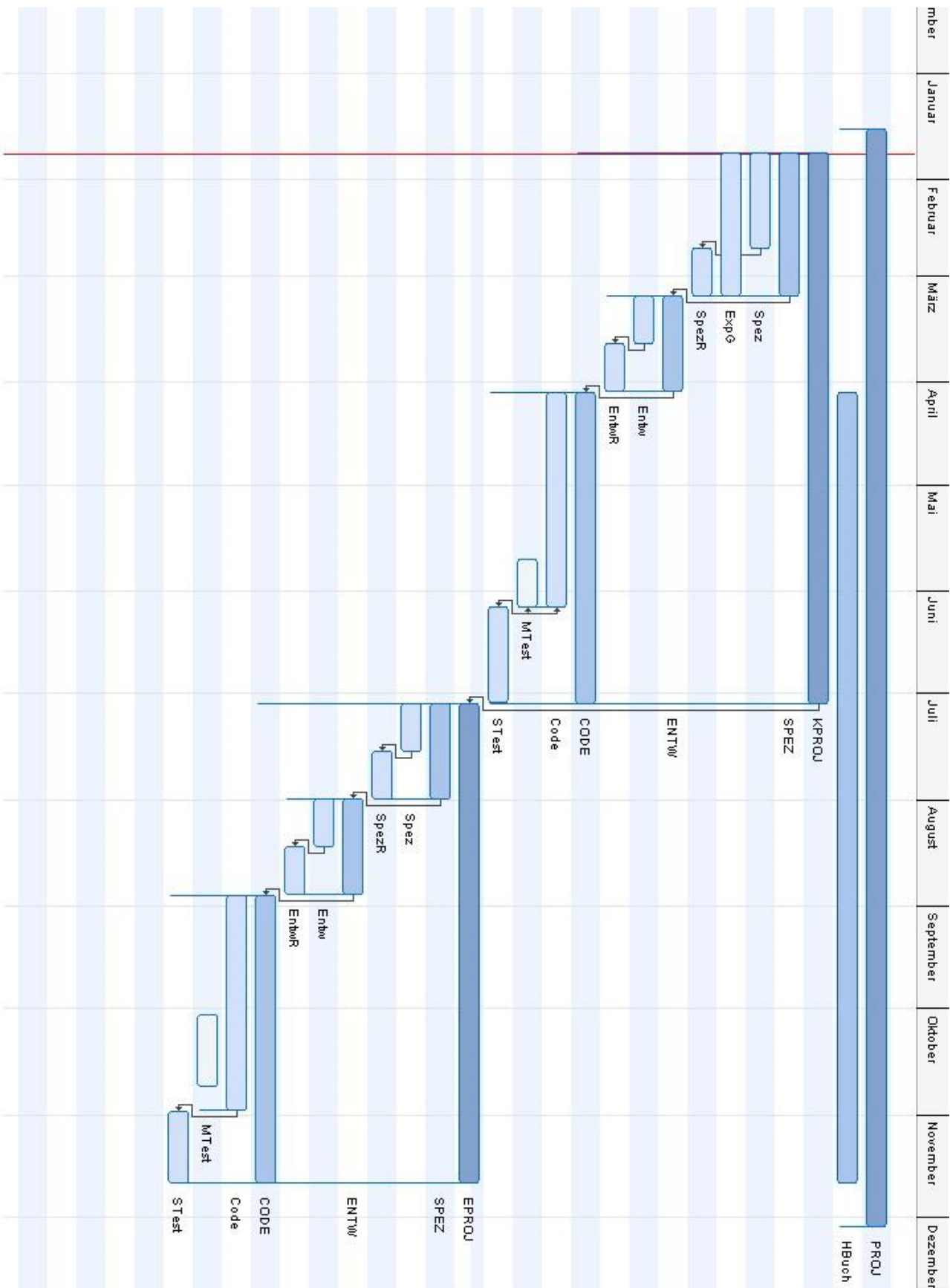
Im Folgenden werden die Meilensteine mit ihrem jeweiligen Fälligkeitsdatum und den beteiligten Dokumenten erklärt. Außerdem ist gezeigt, zu welchem Teilprojekt die Meilensteine gehören. Für den Kunden relevante Meilensteine sind durch farbige Hinterlegung angezeigt.

Proj.	#	Meilenstein	Dokumente	Termin
Kernprojekt	1	Projektbeginn		22.11.2010
	2	Kundengespräch	Analysenotizen, Projektplan	23.11.2010
	3	Spezifikation	Spezifikation	14.02.2011
	4	Spezifikation	Korrigierte Spezifikation	28.02.2011
	5	Entwurf	Entwurf	28.03.2011
	6	Systemtestplan	Systemtestplan	25.04.2011
	7	Implementierung	Implementierung, Modultest	30.05.2011
	8	Kernsystem	Implementierung, Systemtestprotokoll	27.06.2011
Erweiterungsprojekt	9	Projektbeginn		27.06.2011
	10	Spezifikation	Spezifikation	11.07.2011
	11	Entwurf	Korrigierte Spezifikation, Entwurf	08.08.2011
	12	Systemtestplan	Systemtestplan	19.09.2011
	13	Implementierung	Implementierung, Modultest	24.10.2011
	14	Auslieferung	Release Candidat, Systemtestprotokoll	14.11.2011
	15	Abnahme		27.11.2011
	16	Projektabschluss		27.11.2011

## Geplanter Ablauf

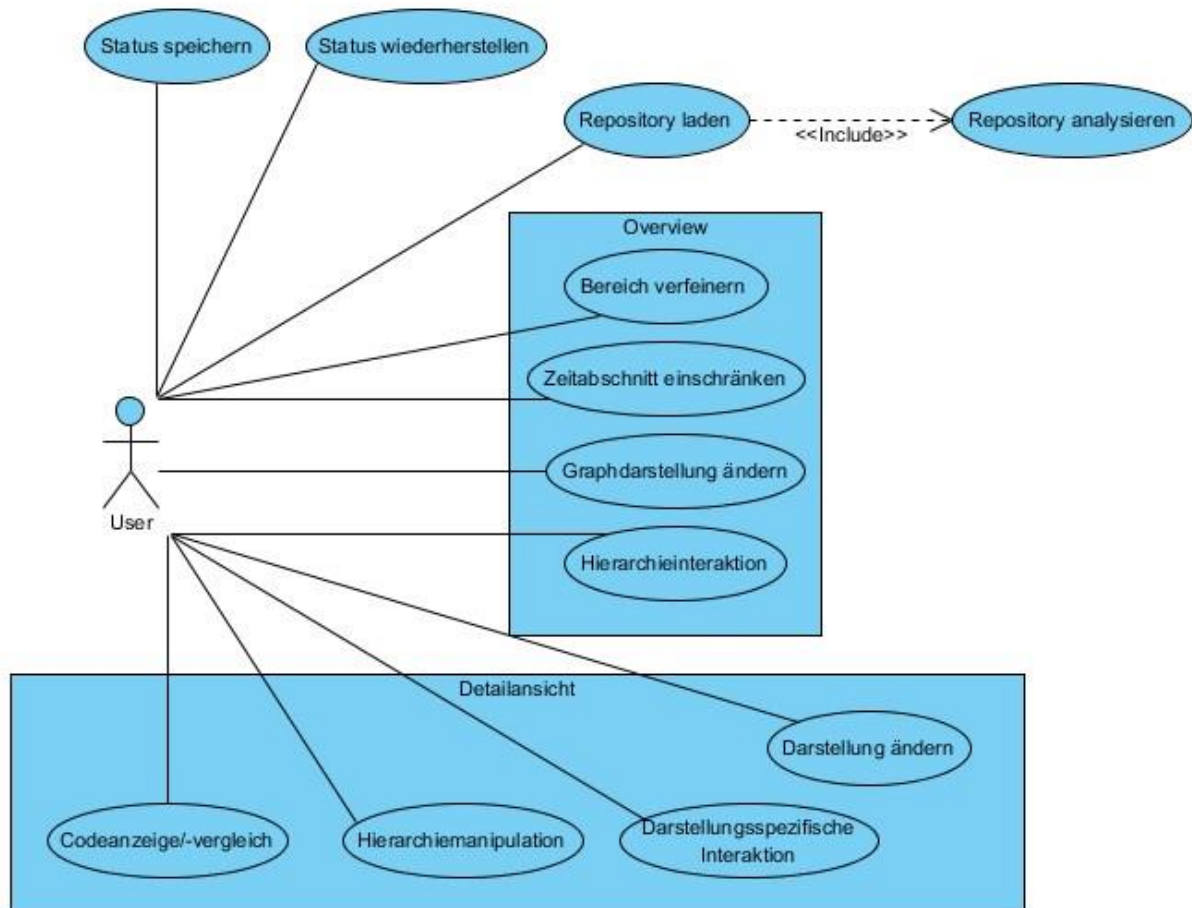
Der durch die Meilensteine geplante Ablauf wird im folgenden Gantt-Diagramm dargestellt.

Sollten oben genannte Meilensteine wegen Verzögerungen in der Projektarbeit nicht eingehalten werden können, wird in Absprache mit dem Kunden entschieden, wie weiter verfahren werden soll.



## Funktionaler Umfang

Das folgende Use-Case-Diagramm zeigt den angestrebten Funktionsumfang der kompletten Software, also nach Beendigung von Kern- und Erweiterungsprojekt.



## Umfang des Kernprojekts

Mit dem Abschluss des Kernprojekts bestehen die Graphen- und Datenextraktionsbibliotheken. Auf diesen Bibliotheken setzt eine grafische Benutzungsschnittstelle auf, die im Kernprojekt mit den nötigsten Funktionen ausgestattet wird. Zu den Funktionen dieses Iterationsschrittes gehören:

1. Repositorien können durch das Programm analysiert und dargestellt werden. Um die für die Graphen benötigten Daten zu extrahieren, werden die Möglichkeiten der Reflection und möglicherweise des Phoenix SDK genutzt. Die Daten können in der dann vorliegenden Form auch persistent abgespeichert werden.
2. In der Übersicht können verschiedene Arten von Graphen dargestellt werden mit dem Ziel, in den Programmabläufen und -strukturen Anomalien, Trends, Gegentrends, periodische Verhalten oder temporäre Verschiebungen festzustellen. Hierzu kann in der Übersicht einer der folgenden Graphentypen ausgewählt werden: Ablaufdiagramm, Aufrufdiagramm und Vererbungsstruktur. Grundsätzlich ist die Übersicht zweigeteilt in eine hierarchische Struktur der analysierten Software und die Darstellung der Entwicklung der Revisionen, beispielsweise in Form einer Pixelmap.

3. Der Benutzer kann in der dargestellten Übersicht kleinere Zeitabschnitte und Hierarchiebreiten auswählen und verfeinern oder die Ansicht auf eine breitere Betrachtung zurückstellen. Das Ziel hierbei ist in der Regel die konkrete Findung der in der Übersicht gezeigten Anomalien.
4. Der Benutzer kann zwischen den verschiedenen Graphendarstellungen umschalten.
5. In der feinsten Ansicht kann eine Codedatei, die einem Element im Graphen entspricht, angezeigt und mit ihren Vor- und Nachfolgeversionen verglichen werden.
6. Die Hierarchieinteraktion beschränkt sich auf Selektion und die Reduzierung und Erweiterung von Hierarchieknoten.
7. Der Status des Programms wird beim Beenden gespeichert und beim nächsten Start wiederhergestellt. Darin enthalten sind das analysierte Repository und die darin betrachteten Dateien in der letzten Ansicht.
8. Dem Benutzer ist es möglich, mehrere Ansichten auf Ausschnitte des Repositoriums gleichzeitig geöffnet zu haben. Die Darstellungen sind durch Linking und Brushing miteinander verbunden, sodass Selektionen und Änderungen in einer Ansicht alle anderen Ansichten beeinflussen.
9. Der Benutzer kann die Darstellung der Graphen in eine Grafikdatei exportieren, beispielsweise im PNG- oder XPS-Format.

Der geplante Umfang der im Kernprojekt entwickelten Bibliotheken umfasst die Speicherung und Darstellung der extrahierten Daten in Form verschiedener Graphen. Dazu gehören

- ☐ Knoten-Kanten-Diagramme,
- ☐ TimeRadarTrees.

Die Bibliothek zur Graphendarstellung besitzt eine sehr allgemeine Schnittstelle an die Benutzungsoberfläche. Die Art der Datenhaltung ist ebenfalls mit einer möglichst allgemeinen Schnittstelle versehen, um die Erweiterung um Module, die in anderen Programmiersprachen verfasste Software oder andere Repository-Modelle analysieren, einfach zu gestalten.

## Umfang des Erweiterungsprojekts

Ziel des Erweiterungsprojekts ist es, dem Benutzer komfortablere Interaktionsmöglichkeiten zu bieten. Das beinhaltet folgende Funktionen:

1. Weiterführende Interaktionstechniken mit den Detailgraphen.
2. Dem Benutzer ist es möglich, Knoten in der Hierarchie auf Detailebene umzuordnen.
3. Zusätzliche Implementierung von TimeLineTrees und TimeArcTrees als Darstellungsformen.
4. Eine Hilfefunktion mit Erläuterungen zur Benutzungsschnittstelle.
5. Der Benutzer wird in der Auswahl der Darstellungsform durch die Benutzungsschnittstelle unterstützt.

Es wird in der Entwicklung darauf Wert gelegt, die Usability der Software im Erweiterungsprojekt möglichst zu optimieren. Darin ist auch eine gute Performance, beispielsweise durch Verwendung von Caching, enthalten.



## Entwicklungsrichtlinien

### Konfigurationsmanagement

Zum Einsatz kommt ein Subversion-Repository, das vom Kunden zur Verfügung gestellt wird. Die Repository-Struktur sieht grundlegend zwei Ordner „documents“ und „code“ vor. Im Ordner „documents“ werden alle entstandenen Dokumente abgelegt, im Ordner „code“ werden die Erzeugnisse der Implementierung gespeichert. Abgeschlossene Dokumente, die dem Kunden vorgelegt wurden, werden entsprechend markiert und nicht mehr verändert. Solche Dokumente dürfen nur begründet und nach Absprache mit der Team- und Projektleitung überschrieben werden.

Jeder SVN-Commit ist durch einen Commit-Kommentar zu erklären.

Dateien, die im Ordner „documents . stable“ abgelegt werden, müssen im PDF-Format vorliegen.

### Design- und Programmierrichtlinien

Für die Entwicklung des Codes in C# halten sich die Entwickler an den offiziellen, von Microsoft vorgeschlagenen Styleguide für die Programmiersprache C#. Alle Klassen, Methoden etc. sind aussagekräftig, vollständig und korrekt zu kommentieren mit den Möglichkeiten der von C# gebotenen Dokumentationskommentaren.

### Verwendete Software

Als Entwicklungsumgebung wird das Microsoft Visual Studio 2010 verwendet. Zur Analyse der Repositorien wird die Bibliothek SharpSVN und eventuell die Möglichkeiten von Microsofts Phoenix SDK verwendet. Die Verwendung des Phoenix SDK kann dazu führen, dass eine frühere Version (2008) des Microsoft Visual Studio verwendet werden muss.

Für die Kommunikation im Team wird eine Wiki-Seite mit DekiWiki verwendet sowie Skype zur direkteren Kommunikation.

Von allen Projektmitarbeitern wird eine Aufwandserfassung verlangt. Dazu wird die Onlinesoftware Kimai verwendet.

Als Issue-Tracker wird Redmine verwendet.

Die Reviews werden werkzeugunterstützt durch die Software RevAger geführt.

Die Formulierung der Dokumente erfolgt in Microsoft Office Word. Für die Erstellung von finalen Dokumenten mit der entsprechenden Markierung im Repository muss die Möglichkeit, Dateien im PDF-Format zu speichern, gegeben sein.

Die Verwendung des SVN-Repositories erfolgt über die Funktionen des Microsoft Visual Studios.

## Entwicklungsprozess

### Aufteilung in zwei Projekte

Das Gesamtprojekt wird aufgeteilt in zwei Iterationen, die jeweils nach dem Wasserfallmodell ablaufen. In der ersten Iteration („Kernprojekt“) wird die Bibliothek, die den Systemkern darstellt, sowie eine grundlegende grafische Benutzungsschnittstelle entwickelt. In der zweiten Iteration

(„Erweiterungsprojekt“) wird die Benutzungsschnittstelle im Hinblick auf Funktionsumfang und Benutzungskomfort erweitert sowie evtl. weitere Features umgesetzt.

## Phasen im Kernprojekt

### Spezifikation

1. Analyse: Es werden Kundengespräche geführt und die Rahmenbedingungen sowie die Verbindlichkeiten festgelegt.
2. Spezifikation: Es wird ein umfangreiches Dokument erstellt, das alle vom Kunden geforderten Anforderungen und ein Begriffslexikon zur Bereinigung von Mehrdeutigkeiten enthält. Das Dokument ist möglichst präzise zu formulieren.
3. Spezifikationsreview: Die Spezifikation wird in einem technischen Review geprüft.
4. Korrektur der Spezifikation: Anhand der Ergebnisse aus dem Review wird eine vorläufige Endfassung der Spezifikation ausgearbeitet.

Parallel zur Spezifikation der Anforderungen werden einige Projektmitarbeiter mit der Einarbeitung in kritische Themen beauftragt, zum Beispiel die Analyse der Repositorien und die Speicherung der Daten. Dadurch soll sichergestellt werden, dass sich durch die Beschränkungen, die diese Themen der Software auferlegen, keine Risiken für den Verlauf des Projekts entstehen. Weitere Projektmitarbeiter werden sich mit Graphenvisualisierung und WPF befassen.

### Entwurf

1. Entwurf: Für die Software wird eine konkrete Architektur entwickelt und die Komponenten des Systems werden entworfen.
2. Entwurfsreview: Analog zur Prüfung der Spezifikation wird der Entwurf einem technischen Review unterzogen.
3. Korrektur des Entwurfs: Die im Review erkannten Probleme werden im Entwurf korrigiert.

### Implementierung

Die Codierung der spezifizierten Anforderungen erfolgt, wie sie in der Systemarchitektur im Entwurf festgelegt wurden. Bei der Implementierung ist auf sauber strukturierten und ausführlich kommentierten Code zu achten. Der Qualitätssicherungsingenieur wird in dieser Phase die Einhaltung der Design- und Programmierrichtlinien sowie die Kommentare überprüfen und damit die Qualität des erstellten Codes sicherstellen.

### Test

1. Aufstellen eines Testplans: Es wird ein Plan aufgestellt, der angibt, wie und in welchem zeitlichen Rahmen die Implementierung auf ihre Korrektheit getestet wird.
2. Testdurchführung: Die im Testplan aufgestellten Tests werden durchgeführt. Mängel an der Implementierung werden danach korrigiert.

## Phasen im Erweiterungsprojekt

### Spezifikation

1. Spezifikation: Die Erweiterungen werden spezifiziert.
2. Spezifikationsreview: Die Spezifikation der Erweiterungen wird in einem technischen Review geprüft.

3. Korrektur der Spezifikation: Anhand der Ergebnisse wird die Spezifikation der Erweiterungen korrigiert.

### Entwurf

1. Entwurf: Die Einbettung der Erweiterungen in das Kernsystem wird entworfen.
2. Entwurfsreview: Das Entwurfsdokument wird einem technischen Review unterzogen.
3. Korrektur des Entwurfs: Der Entwurf wird anhand der Ergebnisse des Entwurfsreviews modifiziert und korrigiert.

### Implementierung

Die Erweiterungen werden implementiert. Es gelten die gleichen Regeln wie im Kernprojekt.

### Test

1. Aufstellen eines Testplans: Es wird ein Plan aufgestellt, der angibt, wie und in welchem zeitlichen Rahmen die Implementierung auf ihre Korrektheit getestet wird.
2. Testdurchführung: Die im Testplan aufgestellten Tests werden durchgeführt. Mängel an der Implementierung werden danach korrigiert.

### Auslieferung

Nach der Abnahme des Produkts durch den Kunden wird die Software ausgeliefert.

### Dokumentationsplan

Folgende Dokumente werden im Rahmen des Projekts erstellt und gewartet:

- Fragenkatalog und Analyseergebnisse
- Projektplan
- Spezifikation des Kernsystems mit Begriffslexikon
- Entwurf und Architektur des Kernsystems
- Spezifikation des Erweiterungssystems
- Entwurf und Architektur des Erweiterungssystems
- Testpläne und Testprotokolle
- Benutzungshandbuch
- Quellcode

### Prüfungen

Beide Spezifikationen und Entwürfe werden in einem technischen Review auf ihre Qualität und Fehlerfreiheit überprüft. Die Implementierungen der Projekte werden im Rahmen von Codetests überprüft. Dabei werden sowohl einzelnen Module durch Unit-Tests als auch die komplette Software anhand aus der Spezifikation abgeleiteter Testfälle geprüft.

Die Erstellung von verständlichem und wartbarem Code wird durch einen Qualitätssicherungsingenieur unterstützt.

### Projektorganisation

#### Schnittstelle zum Auftraggeber

Der Kunde steht den Entwicklern per Telefon-, E-Mail- oder persönlicher Kommunikation zur Verfügung.

## Schnittstellen zu anderen Projekten

Zur Datengewinnung wird die abgeschlossene Schnittstelle der Bibliothek SharpSVN und eventuell das Phoenix SDK benutzt.

## Schlüsselpersonen

Im Folgenden sind die am Projekt beteiligten Personen und ihre Rollen im Projekt, soweit diese bereits feststehen, aufgelistet.

**Kunde** Dipl.-Inf.  
Telefon:  
E-Mail:

**Prüfer** Prof. Dr.  
Telefon:  
E-Mail:

**Betreuer** Dr.  
Telefon:  
E-Mail:

Dipl.-Inf.  
Telefon:  
E-Mail:

Dipl.-Inf.  
Telefon:  
E-Mail:

**Projektleiterin**

**Projektmitarbeiter**

Folgende Rollen sind dabei von den Projektmitarbeitern zu besetzen:

Rolle	Beschreibung
-------	--------------

Projektleiter	<p>Der Projektleiter organisiert die Arbeiten im Projekt, überwacht den Fortschritt und die Einhaltung des Zeitplans, achtet auf eventuell auftretende Risiken und vertritt das Projektteam gegenüber dem Kunden.</p> <p>In der ersten Sitzung wurde Miriam Greis zur Projektleiterin gewählt; Stefan Gerzmann wird ihre die Stellvertretung übernehmen.</p>
QS-Ingenieur	<p>Der Qualitätssicherungs-Ingenieur überwacht während der Kodierung die Einhaltung der Design- und Programmierrichtlinien.</p> <p>Christian Buchgraber wurde in der ersten Sitzung zum QS-Ingenieur auserkoren.</p>
Entwickler	Der Entwickler erstellt Spezifikation und Entwurf. Er ist für die Umsetzung der Entwürfe in sauberen, gut strukturierten und dokumentierten Code verantwortlich.
Tester	Der Tester prüft anhand definierter Testfälle, ob die entwickelte Software den spezifizierten Vorgaben entspricht. Tester dürfen nicht Entwickler des Codes sein, den sie testen.
Review-Moderator	Der Review-Moderator ist für die Organisation des Reviews verantwortlich. Er darf sonst keine Rolle im Review einnehmen und auch nicht Autor des zu prüfenden Dokuments sein.
Review-Gutachter	Der Review-Gutachter prüft das Dokument anhand der ihm durch den ReviewModerator zugeteilten Kriterien. Er darf sonst keine Rolle im Review einnehmen und auch nicht Autor des zu prüfenden Dokumentes sein.
Review-Notar	Der Review-Notar ist für die Dokumentation der Befunde während dem Review verantwortlich. Er darf sonst keine Rolle im Review einnehmen und auch nicht Autor des zu prüfenden Dokumentes sein.

## Entwicklungsplan

### Kostenplan

Von den 16 für das Studienprojekt angesetzten Semesterwochenstunden Zeitaufwand tragen 10 SWS zur tatsächlichen Entwicklung der Software bei. Mit einer Projektdauer von einem Jahr entspricht das einer Gesamtarbeitszeit von etwa 400 Stunden pro Mitarbeiter, also insgesamt 4.800 Stunden.

Bei einem angenommenen Lohn von 100,– €/Stunde ergibt das Projektkosten von insgesamt 480.000,– €.

### Risiken und ihre Bewertung

Im Folgenden werden Risiken vorgestellt, die aus Entwicklersicht für das Projekt relevant und im Eintrittsfall problematisch werden können. Dabei werden die Risiken anhand ihrer Kosten für das Projekt und der Eintrittswahrscheinlichkeit bewertet und Gegenmaßnahmen formuliert.

<b>Risikobeschreibung</b>	<b>Ausfall eines Teammitglieds</b>	
<b>Einstufung</b>	mittel	
<b>Wahrscheinlichkeit</b>	mittel	

<b>Vermeidungsmaßnahmen</b>	Auf gute Kommunikation im Team achten Arbeiten an einem Bereich werden nicht alleine durchgeführt Auf gute Dokumentation wird Wert gelegt Prüfungszeiträume werden in der Arbeitsverteilung eingeplant
<b>Gegenmaßnahmen</b>	Neuverteilung der ausstehenden Arbeit

<b>Risikobeschreibung</b>	<b>Mangelhafte Schnittstellenspezifikation</b>	
<b>Einstufung</b>	kritisch	
<b>Wahrscheinlichkeit</b>	mittel	
<b>Vermeidungsmaßnahmen</b>	Genaue Definition der Schnittstellen im Entwurf, ausreichend Zeit für die Schnittstellendefinition einräumen	
<b>Gegenmaßnahmen</b>	Probleme der Spezifikation frühzeitig ansprechen und beheben	

<b>Risikobeschreibung</b>	<b>Schlechte Stimmung im Team</b>	
<b>Einstufung</b>	mittel	
<b>Wahrscheinlichkeit</b>	mittel	
<b>Vermeidungsmaßnahmen</b>	Viel Kommunikation, Treffen außerhalb des Projekts	
<b>Gegenmaßnahmen</b>	Regeln für den gemeinsamen Umgang festlegen	

<b>Risikobeschreibung</b>	<b>Mangelnder Einsatzwille</b>	
<b>Einstufung</b>	gering	
<b>Wahrscheinlichkeit</b>	mittel	
<b>Vermeidungsmaßnahmen</b>	Zeiterfassung beobachten, Projektleiter verteilt Aufgaben und überprüft deren Durchführung	
<b>Gegenmaßnahmen</b>	Persönliches Gespräch mit dem Projektleiter, eventuell auch mit den Projektbetreuern	

<b>Risikobeschreibung</b>	<b>Kommunikationsprobleme im Team</b>	
<b>Einstufung</b>	mittel	
<b>Wahrscheinlichkeit</b>	mittel	
<b>Vermeidungsmaßnahmen</b>	Kommunikationsmittel am Anfang des Projekts festlegen, regelmäßige Besprechungen	
<b>Gegenmaßnahmen</b>	Probleme beim Teamleiter ansprechen	

<b>Risikobeschreibung</b>	<b>Fehlendes Wissen</b>	
<b>Einstufung</b>	mittel	
<b>Wahrscheinlichkeit</b>	hoch	
<b>Vermeidungsmaßnahmen</b>	Einarbeitung in die Themen, Seminarvorträge, Workshops veranstalten, Spezialisierung von Teammitgliedern	
<b>Gegenmaßnahmen</b>	Rückmeldung beim Teamleiter für zusätzliche Einarbeitungszeit, Kommunikation mit anderen Teammitgliedern	

Risikobeschreibung	Unterschätzte Einarbeitungszeit		
Einstufung	mittel		
Wahrscheinlichkeit	mittel		
Vermeidungsmaßnahmen	Gruppenorganisation in der großzügig eingeplanten Einarbeitungszeit		
Gegenmaßnahmen	Anpassung der Zeitplanung		

Risikobeschreibung	Zeitliche Verzögerung		
Einstufung	kritisch		
Wahrscheinlichkeit	hoch		
Vermeidungsmaßnahmen	Regelmäßige Prüfung des Terminplans, Zeitpuffer bei den geplanten Meilensteinen, Kommunikation und Berichterstattung des Projektstandes an den Kunden		
Gegenmaßnahmen	Zeitplanung und Anforderungen in Absprache mit dem Kunden anpassen		



Risikobeschreibung	Hohe Fehlerrate		
Einstufung	mittel		
Wahrscheinlichkeit	mittel		
Vermeidungsmaßnahmen	Tests, Qualitätssicherungs-Ingenieur, gut dokumentierter und kommentierter Quellcode		
Gegenmaßnahmen	Behebung der gefundenen Fehler		



Risikobeschreibung	Kommunikationsprobleme zwischen Kunde und Entwicklern	
Einstufung	kritisch	
Wahrscheinlichkeit	niedrig	
Vermeidungsmaßnahmen	Regelmäßige Rücksprachen mit dem Kunden, Besprechung der Zwischenergebnisse	
Gegenmaßnahmen	Besprechung der Lage mit dem Kunden und entsprechende Anpassung	



Risikobeschreibung	Konzentration auf unwichtige Features	
Einstufung	kritisch	
Wahrscheinlichkeit	niedrig	
Vermeidungsmaßnahmen	Prioritätenliste für Features verwenden, unwichtige Features aufschieben, klare Trennung in Features für die Iterationsschritte	
Gegenmaßnahmen	Durchsetzung der Prioritätenlisten durch die Team- und Projektleitung	



Risikobeschreibung	Unvollständige Rollenverteilung	
Einstufung	kritisch	
Wahrscheinlichkeit	niedrig	
Vermeidungsmaßnahmen	Gemeinsam die Rollenverteilung gemäß der Selbsteinschätzung und Motivation der Mitglieder vornehmen	

<b>Gegenmaßnahmen</b>	Überforderte Mitglieder unterstützen, eventuell Rollentausche vornehmen
-----------------------	---

<b>Risikobeschreibung</b>	<b>Ungünstige Werkzeuge</b>	
<b>Einstufung</b>	mittel	
<b>Wahrscheinlichkeit</b>	niedrig	
<b>Vermeidungsmaßnahmen</b>	Vorab über die Funktionalitäten der verschiedenen Werkzeuge informieren und mit den Anforderungen für die Projektarbeit vergleichen	
<b>Gegenmaßnahmen</b>	Sofern es durch das Kosten-/Nutzen-Verhältnis möglich ist, Werkzeuge wechseln	

<b>Risikobeschreibung</b>	<b>Datenverlust, Ausfall eines Werkzeugs</b>	
<b>Einstufung</b>	kritisch	
<b>Wahrscheinlichkeit</b>	niedrig	
<b>Vermeidungsmaßnahmen</b>	Regelmäßige Backups, Problemlösung vor der Benutzung	
<b>Gegenmaßnahmen</b>	Backup wiederherstellen, Rekonstruktion des aktuellen Stands durch lokale Daten bei den Teammitgliedern	

<b>Risikobeschreibung</b>	<b>Schlechte Architektur</b>	
<b>Einstufung</b>	kritisch	
<b>Wahrscheinlichkeit</b>	niedrig	
<b>Vermeidungsmaßnahmen</b>	Ausreichend Zeit für den Entwurf einräumen, Diskussion der Ergebnisse mit den Betreuern	
<b>Gegenmaßnahmen</b>	Überarbeitung des Entwurfs	

<b>Risikobeschreibung</b>	<b>Teammitglieder sind an Terminen abwesend</b>	
<b>Einstufung</b>	niedrig	
<b>Wahrscheinlichkeit</b>	hoch	
<b>Vermeidungsmaßnahmen</b>	Termine schriftlich verteilen und zentral organisieren, Sanktionen bei Nichterscheinen auferlegen	
<b>Gegenmaßnahmen</b>	Sitzungsabläufe werden protokolliert, fehlende Mitarbeiter werden über die Treffen informiert	

Risikobeschreibung	Performanceprobleme		
Einstufung	kritisch		
Wahrscheinlichkeit	hoch		
Vermeidungsmaßnahmen	Vorwegnahme von Performanceproblemen durch Einarbeitung in kritische Themen in Expertengruppen		
Gegenmaßnahmen	Anforderungen in Absprache mit dem Kunden und den Betreuern anpassen		



<b>Risikobeschreibung</b>	<b>Schlechte Verfügbarkeit des Kunden</b>	
<b>Einstufung</b>	kritisch	
<b>Wahrscheinlichkeit</b>	mittel	
<b>Vermeidungsmaßnahmen</b>	Termine werden frühzeitig mit dem Kunden geplant	
<b>Gegenmaßnahmen</b>	Fragen per Mail oder mit den Betreuern klären, sofern sie zum Anlass den Kunden vertreten können	

## Versionshistorie

Beschreibung	Version	Datum
Erstellung des Dokuments	0.1	17.12.2010
Deckblatt, Überarbeitung des Risikokatalogs	0.2	19.12.2010
Risiken, generelle Überarbeitung, Rollen, Gantt-Chart, Funktionsumfang	0.3	10.01.2011
Überarbeitung nach Gespräch mit Betreuern und dem gesamten Team	0.4	25.01.2011
Finale Fassung des Projektangebots	1.0	27.01.2011

**Das Projekt soll nach diesen Vorgaben durchgeführt werden.**

---

Unterschrift Kunde

---

Unterschrift Projektleitung