# Projektplan


Master Infrastructure Situation Display - Observing Windows and Linux

Ein System zur Überwachung von verteilten und heterogenen Netzwerken

P. Brombosch, E. Doust, D. Krauss, F. Müller, Y. Noller,

H. Schäfer, J. Scheurich, A. Schneider, S. Zillessen

Universität Stuttgart

Studenten der Fachrichtung Softwaretechnik

| | |
|---|---|
| **Autoren:** | A.Schneider, S. Zillessen, H. Schäfer |
| **Erstellt am:** | 28. Mai 2012 |
| **Freigegeben am:** | 4. Februar 2013 |
| **Version:** | Version 1.7 |

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Zweck und Abgrenzung

Dieser Projektplan legt alle projektspezifischen Daten, die den Auftragsteller (Institut *VISUS* der Universität Stuttgart) und die Entwickler betreffen fest. Dazu gehören die Zielsetzungen, Zeitplanung, Vorgehensweisen und gegebene projektspezifische Normen. Während der Durchführung des Projektes kann es zu Anforderungsänderungen von einem der Vertragspartner kommen. Diese können zu einer Änderung des Projektplanes führen. Grundsätzlich sind Änderungen der Anforderungen durch die Projektleitung zu prüfen. Anschließend wird ggf. eine neue Zeitplanung erstellt.

## 1.2 Projektüberblick und Motivation

Das Projekt *MISD* wurde durch das VISUS der Universität Stuttgart initialisiert. Die Entwicklung der Software wird im Rahmen eines Studienprojektes(*StuPros*) durchgeführt. An ihm sind neun Studenten der Softwaretechnik beteiligt.

Der Kunde benötigt für die EDV-Infrastruktur des VISUS eine Überwachungssoftware, die es Ihm ermöglicht relevante Status- und Inventarinformationen über das verteilte System abzurufen, zu überwachen und darzustellen. Die bisherige Vorgehensweise (der einzelnen Überwachung von Linux und Windows-Rechnern sowie der Cluster) ist nicht praktisch und keineswegs effizient.

Kernbestandteile des Pojektes werden dabei folgende Punkte sein:

- Die Datenbeschaffung der Inventar- und Statusinformationen
- Die Datenspeicherung und Auswertung dieser Daten
- Die Visualisierung der gewonnenen Daten auf den *Clients* ( Desktop und Powerwall)

Dabei werden die folgenden Aspekte im gesamten Entwicklungsprozess besonders beachtet:

- Modularität der Software
- Erweiterbarkeit der Datenbeschaffung und Visualisierung
- Optimierung der Interaktion und Darstellung für die Powerwall
- Wartbarkeit

Da an die Software ein hoher Anspruch an die Erweiterbarkeit, die Wartbarkeit und die Stabilität gelegt wird, ist die Rolle des Qualitätsmanagers für den Projektverlauf besonnders wichtig. Aus diesem Grund wird ein Teammitglied sich dieser Aufgabe mit 50 % seiner Arbeitszeit widmen.

## 1.3 Namensgebung

Das diesjährige StuPro wurde vom Institut VISUS der Universität Stuttgart unter dem Namen **Master Infrastructure Situation Display** ausgeschrieben. Die Kurzbezeichnung des Projekts lautet **MISD**.

Um der Anwendung eine wieder erkennbare Identität sowie einen persönlichen Bezug zu verleihen, wurde beschlossen, diese mit dem griffigen Namenszusatz **OWL** zu versehen, insgesamt heißt das Projekt nun **MISD OWL** – *Master Infrastructure Situation Display Observing Windows and Linux*. Die Abkürzung **OWL** symbolisiert das englische Wort `Eule` wieder, womit wir ausdrücken möchten, dass unsere Anwendung wie ein Eule stets alles im Blick behält.

# Kapitel 2

# Formale Grundlagen

## 2.1 Vertragliche Anforderungen an die Projektdurchführung

Das Projekt wird in einem Team bestehend aus neun Studenten durchgeführt, wobei jedes Teammitglied ein Arbeitspensum von 450h bis zum 31.03.2013 zu erfüllen hat. Als Entwicklungsprache ist `C#` für die Entwicklung der `Windows`-Software vorgeschrieben. Für die Entwicklung der Dienste auf den `Linux`-Systemen wird `Mono` verwendet. Diese Entscheidung wurde einstimmig von den zuständigen Entwicklern getroffen.

## 2.2 Vertragliche Anforderungen an das Produkt

Das Produkt muss spätestens am **31.03.2013** lauffähig und vom Kunden einsetzbar sein. Im Rahmen der Abnahme erhält der Kunde das Enprodukt in Form einer `CD` (siehe Abschnitt Lieferumfang), die alle relevanten Projektergebnisse enthält. Alle Dokumente müssen aktuell sein. Die Abnahme der entwickelten Software geschieht unter Anwesenheit der Projektleitung, des Kunden und des Qualitätsmanagers.

## 2.3 Datenkommunikation

Die Übertragung aller Informationen zwischen den *Clients*, dem *Server* und den *Workstations* erfolgt verschlüsselt, sofern es sich bei den Paketen nicht um einfache *Ping-Requests*, *Daten-Sende-Token* oder *Positionsdaten* des *Zauberstabes* handelt. Insbesondere werden alle *Statistiken*, *Inventar-* und *Statusdaten* verschlüsselt übertragen. Dabei kommt das `Windows Communication Framework` (WCF) zum Einsatz.

## 2.4 Gesetzliche Auflagen

Der Auftraggeber verpflichtet sich, die Administratoren und Benutzer über die Ausführung des Dienstes von MISD auf den *Workstations* und *Clustern* zu informieren und sie über die gesammelten Daten (insbesondere Informationen über die Anmeldung von **Usern** an den *Workstations/Clustern*) aufzuklären. Die Speicherung der Daten erfolgt auf dem vom Kunden bereitgestellten *Server*. Das Entwicklungsteam wird sich nicht um zusätzliche Sicherheitsfunktionen auf diesem kümmern. Zusätzlich ist dem Auftraggeber bekannt, dass es keine weitere Rechteverwaltung bei der Betrachtung von Daten geben wird.

## 2.5 Lizenz

Die Software steht der Universität Stuttgart zur freien Verfügung, da es sich um eine Arbeit im Rahmen des Studiums handelt.

Ansonsten wird die Software unter der GNU Lesser General Public License (LGPL) [1] veröffentlicht.

## 2.6 Namenskonventionen für dieses Dokument

- Begriffe, die Referenzen auf das Begriffslexikon darstellen, werden *kursiv* geschrieben.
- Besonders wichtige Informationen oder hervorzuhebende Teile werden **fett** geschrieben.

[1] http://www.gnu.org/licenses/lgpl.html

- Verweise auf externe Informationen werden als Fußnoten dargestellt.

# Kapitel 3

# Leistungen

## 3.1 Lieferumfang

Zum Projektabschluss werden folgende Komponente an den Kunden ausgeliefert:

- Software MISD OWL
- Programmcode
- Installationsanleitung
- Benutzerhandbuch
- Architekturentwurf (inkl. Schnittstellen)
- Spezifikation
- Begriffslexikon
- Review-Protokolle
- Projektplan
- Testdaten
- Testprotokolle

## 3.2 Resultate, die nicht zum Lieferumfang gehören

Folgende Dokumente und Resultate werden nicht an den Kunden ausgeliefert:

- Besprechungsprotokolle
- Interne Produktversionen

## 3.3 Leistungen des Auftraggebers

Das VISUS stellt der Projektgruppe folgende Ressourcen zur Verfügung:

- Rechnerpool im VISUS-Gebäude
- Rechnerpool im Informatik-Gebäude
- Visualisierung-Labor im Informatik-Gebäude
- Besprechungsräume
- Zugang zum Visualisierung-Labor im VISUS-Gebäude
- Einführung in die Visualisierungs-Labore (insb. Powerwall)
- Zugriff auf einen Server für MISD
- SVN-Server
- Redmine-Server
- Kimai-Server
- Homepage zur Dokumentenveröffentlichung für die Projektgruppe
- Zugriff auf die Cluster für jedes Mitglied der Projektgruppe
- Zugriff auf das ActiveDirectory
- 1-2 Gutachter für die Reviews

## 3.4 Externe Meilensteine

| | | |
|---|---|---|
| SPEZ | 06.08.2012 | Spezifikation |
| M1 | 20.11.2012 | Kernsystem |
| M2 | 12.01.2013 | Beta-Version |
| M3 | 14.03.2013 | Release Candidate |
| M4 | 31.03.2013 | Projektabschluss |

## 3.5 Abnahmeprozedur

Zum Projektabschluss wird der Kunde das installierte Produkt prüfen. Als Grundlage für die Prüfung gelten die folgenden Dokumente in ihrer aktuellsten Form: Spezifikation, Projektplan. Anschließend findet eine öffentliche Präsentation des gesamten StuPros statt.

## 3.6 Änderungsverfahren

Änderungen der Anforderungen an MISD sind Diskussionsgegenstand zwischen Kunden und Projektleitung. Bei Bedarf werden beide Parteien Experten hinzuziehen. Dies geschieht in einem Meeting zwischen Kunden und Projektleitung, das zwei Werktage vorher vereinbart wird. Drei Monate vor Projektabschluss werden keine Änderungswünsche des Kunden mehr akzeptiert, die die Architektur des Produkts grundlegend beeinflussen.

# Kapitel 4

# Entwicklungsprozess

## 4.1 Strategie für die Qualitätssicherung der Entwicklung und Integration

Während des gesamten Entwicklungsprozesses ist eine hohe Qualität der erstellten Dokumente und Software zu gewährleisten, da es sich zum Teil um sensible Daten handelt und die Software später leicht erweiterbar und wartbar sein muss. Dazu werden die entstandenen Dokumente (insbesondere Spezifikation und Entwurf) durch Reviews geprüft und anhand der Befunde korrigiert. Bei diesen Reviews soll der Kunde als Manager anwesend sein sowie ein bis zwei Betreuer als Gutachter an den Reviews teilnehmen. Die entsprechenden Einladungen hierzu gehen immer mindestens eine Woche vor dem Termin an die Betreffenden heraus. Für die Durchführung und Organisation dieser Termine ist der jeweilige Moderator zuständig. Außerdem ist der Projektfortschritt kontinuierlich zu erfassen. Dazu wird der Kunde einen wöchentlichen Bericht der Projektleitung im Umfang von ein bis zwei Seiten erhalten.

## 4.2 Projektspezifisches Entwicklungsmodell

Als Prozessmodell kommt ein abgewandeltes *Treppenmodell* (LLSE, S. 176ff) in Kombination mit dem *Wasserfallmodell* (LLSE, S. 156ff) zum Einsatz. Die Projektzeit vom 25.05.2012 bis zum 31.03.2013 wird dabei in eine Spezifikations- und drei Entwicklungstreppen unterteilt.

Die Spezifikationstreppe zu Beginn des Projektes wird am 06.08.2012 enden. In dieser Phase wird sich eine Teilgruppe des Teams mit der Erstellung einer Grob-Spezifikation auseinandersetzen, während der Rest des Teams sich über die verwendeten Technologien und Werkzeuge informiert und die Erkenntnisse jeweils in Form eines Vortrages zur Verfügung stellt. Diese grobe Spezifikation wird den Funktionsumfang, Schnittstellenspezifizierung zwischen den einzelnen Softwaremodulen und die angenommenen Szenarien in Form von **Use-Cases** enthalten. Ebenso wird die Architektur bereits grob entworfen.

In den anschließenden Entwicklungstreppen 1-3 wird dann das Wasserfallmodell in einer abgeänderten Form angewandt. Jede einzelne Treppe wird dabei eine wichtige Komponente des Systems als abgeschlossenes Produkt für den Kunden bereitstellen. In der letzten dieser 3 Entwicklungstreppen wird Refactoring und Bug-Fixing stattfinden werden.

In jeder Stufe wird zuerst die Spezifikation im Hinblick auf die bevorstehende Treppe genau inspiziert und ggf. korrigiert und erweitert. Dadurch kann es ggf. zu kleineren Änderungen kommen, die in Absprache mit dem Kunden in das Projekt fließen. In der anschließenden Entwurfs-Phase wird dann der Entwurf für genau diese zu entwickelnde Treppe vorgenommen. Der darauf folgende Implementierungsphase schließt sich eine Testphase an. Abschließend wird eine Integration in die bereits entstandenen Ergebnisse der Stufen und die Umgebung vorgenommen.

## 4.3 Phasen der Entwicklung

Die Entwicklung der Software soll in den nachfolgenden Phasen ablaufen:

1. Grobe Spezifikation der Anwendung (inkl. Review)
2. *Stufe 1* : Entwicklung des Kernsystems (Datenbeschaffung und -speicherung)
3. *Stufe 2* : Entwicklung der Oberfläche (Powerwall und Desktop) unter der Berücksichtigung des einheitlichen Design-Konzeptes.
4. *Stufe 3* : Spezialisierte Weiterentwicklung der Oberflächen unter Berücksichtigung der Umgebungseigenschaften sowie Code-Refactoring, Bug-Fixing und intensive Integrationstests.

## 4.4 Dokumentationsplan

Die nachfolgenden Dokumente sollen während des Projektes erstellt und gepflegt werden:

- Projektplan
- Begriffslexikon
- Gesamtspezifikation
- Architektur-Entwurf
- Für jede Entwicklungs-Stufe:
  - detaillierte Spezifikation
  - Feinentwurf
  - Abnahmeprotokolle der Arbeitspakete
- Benutzerhandbuch
- Testplan inklusive der Testdaten und dem Testprotokoll
- Schnittstellendokumentation für Erweiterungen
- Prüfungsergebnisse (Reviews und Tests)

Als Prüfungsmethoden kommen bei Software Modul-Tests und bei kritischen Komponenten (insb. den Schnittstellen) Reviews zum Einsatz. Im Falle der Spezifikation und des Entwurfes wird dies durch Reviews und die anschließende Korrektur anhand der Befunde aus diesen gewährleistet.

# Kapitel 5

# Risiken

Während der Entwicklung des Tools *MISD OWL* gibt es mehrere potentielle Risiken. Auf diese muss ggf. durch Notfallmaßnahmen reagiert werden können. Außerdem ist eine Prävention dieser Risiken ein wichtiges Ziel der Projektleitung.

## 5.1 Risiken und ihre Bewertung

Mögliche Risiken in der Entwicklung sind:

### 5.1.1 Ausfall eines Entwicklers

Ein Entwickler kann auf Grund von Krankheit, Abbruch des Studiums oder anderen Gründen zeitweise oder dauerhaft ausfallen.

**Dauerhafter Ausfall**

| | |
|---|---|
| **Wahrscheinlichkeit** | 10 % |
| **Kosten bei Eintritt** | 350 Stunden |
| **Risikokosten** | 0,1 * 350 h * 100 €/h = **3500 €** |

**Zeitweiser Ausfall**

| | |
|---|---|
| **Wahrscheinlichkeit** | 50 % |
| **Kosten bei Eintritt** | 30 Stunden |
| **Risikokosten** | 0,5 * 30 h * 100 €/h = **1500 €** |

### 5.1.2 Zeitverzug bei der Informationserhebung der Cluster

Die Bright-Cluster Zugriffe erwiesen sich bereits in der Vorbereitung des Seminarvortrags als sehr kompliziert.

| | |
|---|---|
| **Wahrscheinlichkeit** | 20 % |
| **Kosten bei Eintritt** | 30 Stunden |
| **Risikokosten** | 0,2 * 30 h * 100 €/h = **600 €** |

### 5.1.3 Defekte im Visualisierungs-Labor

Stehen die Powerwalls des VISUS auf Grund von Defekten, Reparaturen, oder Umbauarbeiten zur Entwicklung nicht zur Verfügung, kann die Oberfläche für die Powerwall nur unter eingeschränkten Bedingungen entwickelt werden.

| | |
|---|---|
| **Wahrscheinlichkeit** | 5 % |
| **Kosten bei Eintritt** | 20 Stunden |
| **Risikokosten** | 0,05 * 20 h * 100 €/h = **100 €** |

### 5.1.4 Ausfall der Projektleitung

Fällt die Projektleitung auf Grund von Krankheit, Abbruch des Studiums oder anderen Gründen die Entwicklung zeitweise oder ganz aus, so muss eine neue Projektleitung gefunden werden, sowie die Übergabe der Aufgaben stattfinden. Die Übergangsphase wird sich in der Projektorganisation bemerkbar machen sowie einen Entwickler komplett an diese Aufgabe binden.

| | |
|---|---|
| **Wahrscheinlichkeit** | 5 % |
| **Kosten bei Eintritt** | 100 Stunden |
| **Risikokosten** | 0,05 * 100 h * 100 €/h = **500 €** |

### 5.1.5 Änderung der Anforderungen

Ändern sich die Anforderungen des Kunden werden teilweise Neuplanungen nötig.

| | |
|---|---|
| **Wahrscheinlichkeit** | 50 % |
| **Kosten bei Eintritt** | 25 Stunden |
| **Risikokosten** | 0,5 * 25 h * 100 €/h = **1250 €** |

## 5.2 Präventivmaßnahmen

### 5.2.1 Ausfall eines Entwicklers

#### Dauerhafter Ausfall

Einen Dauerhaften Ausfall präventiv zu behandeln ist kaum möglich. Es wird jedoch auf konstante Kommunikation mit jedem Teammitglied geachtet, sodass starke Änderungen auch im Privatleben der Entwickler frühzeitig bemerkt werden. Dadurch kann ein flüssigerer Übergang geschaffen werden und Planungen bereits vor dem Ausfall angepasst werden.

#### Zeitweiser Ausfall

Durch kleine Arbeitspakete sind die Aufgaben eines Entwicklers innerhalb von kurzer Zeit auch von anderen Entwicklern zu übernehmen. Dadurch verringert sich die Wartezeit, die durch die Abhängigkeiten entstehen. Zudem werden größere Aufgabenpakete auf Teams von bis zu drei Personen angelegt, welche dann intern durch Lastverteilung die Ausfälle überbrücken können.

### 5.2.2 Zeitverzug bei der Informationserhebung der Cluster

Es wird bereits in der Zeitplanung ein extra Zeitraum für Prototyping der Bright-Cluster-Zugriffe veranschlagt. Dadurch ist der Zeitaufwand zwar in jedem Fall höher, doch das Risiko einer extremen Zusatzdauer wird verringert.

### 5.2.3 Defekte im Visualisierungs-Labor

Durch frühe und regelmäßige Einbindung der Hardware in die Entwicklung können auch während der Zeit von Defekten die Auswirkungen der Software abgeschätzt werden und Entwicklungsphasen mit verzögerter Integrationsphase möglichst problemlos gemeistert werden.

### 5.2.4 Ausfall der Projektleitung

Durch die gesplittete Projektleitung entfällt bei einem Wechsel der Projektleitung ein Großteil der Übergangszeit.

### 5.2.5 Änderung der Anforderungen

In diesem Projektplan wird bereits festgelegt in welchem Rahmen Anforderungsänderungen des Kunden erfolgen können. Durch diese Festlegung ergibt sich ein Handlungsspielraum der Projektleitung, um den Änderungswünschen zu begegnen.

## 5.3 Notfallpläne

Für den Fall, dass ein Risiko eintritt, müssen schnellstmöglich reaktive Maßnahmen ergriffen werden. Diese sind im Folgendem aufgeführt:

### 5.3.1 Ausfall eines Entwicklers

#### Dauerhafter Ausfall

Fällt ein Entwickler bereits früh im Projektverlauf dauerhaft aus, so werden Funktionalitäten aus der Anforderungenliste gestrichen, wie beispielsweise:

- Konfigurationsdateien statt einer Oberfläche zur Konfiguration
- Plugins nur in eingeschränktem Rahmen liefern

**Zeitweiser Ausfall**

Im Fall von kurzfristigen oder zeitlich begrenzten Ausfällen, wie Krankheit oder Urlaub, wird versucht die Aufgabeneinteilung anzupassen und fehlende Stunden in den Folgewochen aufzuholen.

### 5.3.2 Zeitverzug bei der Informationserhebung der Cluster

Wird mehr Entwicklungszeit für die Bright-Cluster-Zugriffe benötigt, so wird die Zusammenarbeit mit dem VISUS vertieft und auf Beispielprojekte zurückgegriffen, welche bereits mit den Clustern arbeiten.

### 5.3.3 Defekte im Visualisierungs-Labor

Beim Ausfall einer Powerwall wird in der Zwischenzeit auf die zweite ausgewichen. Sollten tatsächlich beide Powerwalls längere Zeit defekt sein, wird der Fokus der Entwicklungsarbeit vorübergehend auf die Desktopvisualisierung gelegt.

### 5.3.4 Ausfall der Projektleitung

Bei einem längerfristigen Ausfall wird die stellvertretende Projektleitung seine Entwicklungsaufgaben abgeben und zu 100 % als Projektleitung tätig sein. Bei einem kurzfristigen Ausfall der Projektleitung wird die stellvertretende Projektleitung die Entwicklungsaufgaben zurückstellen oder abgeben.

### 5.3.5 Änderung der Anforderungen

Führen die Änderungen zu ungeplanten neuen Aufwänden, die nicht in den Zeitplan eingepasst werden können, werden nach Verhandlung mit dem Kunden Funktionalitäten von MISD OWL gestrichen.

# Kapitel 6

# Richtlinien für die Entwicklung

Der gezielte Werkzeugeinsatz und Richtlinien sollen die Ziele des Projekts wie hohe Qualität und einheitlicher Code unterstützen sowie die Zusammenarbeit der Teammitglieder vereinfachen.

## 6.1 Konfigurationsmanagement

Um die vielfältigen Änderungen sowie das parallele Arbeiten am Projekt zu vereinfachen, kommt als Konfigurationsmanagement die freie Software SVN (Apache Subversion)[1] zum Einsatz.

Der für SVN nötige Server wird vom Auftraggeber zur Verfügung gestellt und ermöglicht eine zentrale Verwaltung sämtlicher dem Projekt zugehöriger Codeteile sowie der Dokumentation inklusive Versionierung aller Dateien. Hierdurch ist es möglich im Problemfall einfach auf alte Versionen zuzugreifen. Des Weiteren ermöglicht SVN das Arbeiten in "branches" welche die Entwicklung einzelner Bestandteile unabhängig vom Hauptprojekt ermöglichen, wodurch das Hauptprojekt lauffähig bleibt und die zusätzlichen Bestandteile nach der Fertigstellung eingebunden werden.

## 6.2 Design- und Programmierrichtlinien

Die Codierung des Projekts erfolgt bis auf die Dienste für die Linux-Systeme in `C#`.

Als Design- und Programmierrichtlinien kommen bei diesem Projekt die Microsoft MSDN

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

C# Coding Convention[2] sowie die Microsoft MSDN Design Guidelines for Developing Class Libraries[3] zum Einsatz. Hierdurch kann gewährleistet werden, dass auch über mehrere Entwickler hinweg ein einheitlicher, sauberer, gut lesbarer und verständlicher Code entsteht.

Zur Kommentierung wird die built-in-XML-Dokumentation genutzt. Als Kommentierungssprache für den Code kommt ausschließlich Englisch zum Einsatz.

Für die Dienste welche auf den Linux-Systemen ist festgelegt, dass die Implementierung in mit Hilfe von Mono durchgeführt wird. Auch in diesem Bestandteil wird hoher Wert auf eine einheitliche und gut kommentierte Codierung gelegt. Auch für diese Bestandteile sind die oben genannten Richtlinien als Grundlage zu verstehen, auch wenn mögliche Unterschiede in der Syntax und Semantik keine blinde Übertragung der Richtlinien zulassen.

## 6.3 Einsatz von Werkzeugen

Als Entwicklungsumgebung für die C# Entwicklung kommt Microsoft Visual Studio 2010 Ultimate[4] zum Einsatz. Diese Entwicklungsumgebung bietet reichhaltige Unterstützung bei der Entwicklung und ermöglicht über Plugins wie beispielsweise AnkhSVN[5], eine einfache Einbindung des SVN Servers. Des Weiteren bietet Visual Studio auch Unterstützung beim Formatieren und Strukturieren des Codes, beim Erstellen von Tests und in vielen weiteren Bereichen.

Auch für die Entwicklung des Linux-Dienstes wird von den Entwicklern erwartet, dass eine Entwicklungsumgebung zur Unterstützung der Arbeit eingesetzt und genutzt wird.

Zur Aufwandserfassung und -auswertung kommt die Zeiterfassungssoftware Kimai[6] zum Einsatz. Diese ermöglicht eine einfache, zentrale Zeiterfassung für jeden Entwickler und eine Auswertung und Analyse sämtlicher Aufwände. Der benötigte Server wird vom Auftraggeber zur Verfügung gestellt.

Als Bugtracking- und Wikiverwaltungssoftware kommt Redmine[7] zum Einsatz. Hierdurch ist es einfach möglich Fehler zu erfassen und gezielt zu beseitigen. Des Weiteren ist ein Wiki

[2] http://msdn.microsoft.com/en-us/library/ff926074.aspx

[3] http://msdn.microsoft.com/en-us/library/ms229042.aspx

[4] http://www.microsoft.com/germany/visualstudio/products/team/visual-studio-ultimate.aspx

[5] http://visualstudiogallery.msdn.microsoft.com/E721D830-7664-4E02-8D03-933C3F1477F2

[6] http://www.kimai.org

[7] http://www.redmine.org

enthält welches die Möglichkeit bietet, wissenswerte Erfahrungen zentral und durchsuchbar zu erfassen und zu verwalten. Die Integration des Versionsverwaltungssystems SVN ermöglicht es, Änderungen direkt auf einen Fehler zu beziehen und somit Problemlösungen einfach nachvollziehbar zu machen. Auch diese Software läuft auf einem vom Auftraggeber zur Verfügung gestellten Server.

Zur geeigneten Visualisierung des Entwicklungsphasen im Projekt kommt der GTD-Manager[8] zum Einsatz. Hierdurch können Verzögerungen sowie andere Vorkommnisse im Projekt unter anderem als Termindrift-Diagramm dargestellt werden. Für das Gantt-Diagramm wird Redmine als Grundlage dienen, da hier ab der ersten Stufe bereits alle Arbeitsschritte als Tickets hinterlegt werden.

Als Unterstützung für die Planung und Durchführung von Reviews verschiedener Dokumente kommt RevAger[9] zum Einsatz.

[8] `http://www.iste.uni-stuttgart.de/se/werkzeuge/gtd-manager.html`

[9] `http://www.iste.uni-stuttgart.de/se/werkzeuge/revager.html`

# Kapitel 7

# Anforderung an die Umgebung

## 7.1 Infrastruktur

### 7.1.1 Räume und Rechner

Das Team wird vom Kunden mit der nötigen Infrastruktur an Räumen und Rechnern versorgt. Dazu gehört:

- Rechnerpool im VISUS-Gebäude
- Rechnerpool im Informatik-Gebäude
- Visualisierung-Labor im Informatik-Gebäude
- Zugang zum Visualisierung-Labor im VISUS-Gebäude
- Besprechungsräume (evtl. ein freies Doktorandenzimmer)

### 7.1.2 Tools und Software

Zusätzlich wird das Team mit folgenden Tools, Servern und Software ausgestattet:

- Homepage für die Projektgruppe
- SVN-Server
- Redmine-Server

- Kimai-Server
- Visual Studio 10 Ultimate

### 7.1.3 Zugriffe

Um schon frühzeitig mit der realen Umgebung des Produktes zu arbeiten, wird das Team außerdem mit folgenden Rechten ausgestattet:

- Zugriff auf einen Server für MISD
- Zugriff auf eine *Teststrecke* im VIS(US)
- Zugriff auf die Cluster
- Zugriff auf das ActiveDirectory

### 7.1.4 Direkte Beteiligung

Zum reibungslosen Ablauf des Projektes ist eine enge Zusammenarbeit mit dem Auftraggeber notwendig. Dazu wurden bereits jetzt folgende Zusammenarbeiten festgelegt:

- Einführung in die Visualisierungs-Labore (insb. Powerwall)
- Review der Spezifikation (Zwei Betreuer als Gutachter)
- Review des Entwurfs (Zwei Betreuer als Gutachter)

## 7.2 Leistungen Dritter im Projekt

### 7.2.1 Schulung in `C#`

Zur Einarbeitung in die Programmiersprache `C#` wird am Montag den 04.06.2012 von 13.30 Uhr bis 17.00 Uhr Schulung stattfinden. Es sollen in den 3,5 Stunden projektrelevante Themen und Umsetzungen in `C#` erklärt und praktisch angewandt werden. Die Schulung wird von Patrick Mutter durchgeführt.

# Kapitel 8

# Projektorganisation

## 8.1 Schlüsselpersonen des Entwicklerteams

### 8.1.1 Projektleitung

Die Projektgruppe wird von zwei Personen geleitet, die jeweils auch in der Entwicklung tätig sind.

1. Projektleiterin 67 % Leitung - 33 % Entwicklung (Hanna Schäfer, schaeferhannaj@googlemail.com)
2. Stv. Projektleiter 33 % Leitung - 67 % Entwicklung (Sebastian Zillessen, sebastianzillessen@googlemail.com)

### 8.1.2 Qualitätssicherung

Zur Sicherung der Qualität ist ein Mitglied der Entwicklergruppe zu 50 % mit den Aufgaben des Qualitätsmanagement beschäftigt. Dies geschieht in enger Zusammenarbeit mit der Projektleitung.

- Arno Schneider (schneiao@studi.informatik.uni-stuttgart.de)

### 8.1.3 Team

Insgesamt besteht das Entwicklungsteam aus folgenden Studenten:

- Paul Brombosch (etk64978@stud.uni-stuttgart.de)
- Ehssan Doust (swt79282@stud.uni-stuttgart.de)
- David Krauss (swt79771@stud.uni-stuttgart.de)
- Fabian Müller (muellefn@studi.informatik.uni-stuttgart.de)
- Yannic Noller (nolleryc@studi.informatik.uni-stuttgart.de)
- Hanna Schäfer (swt79917@stud.uni-stuttgart.de)
- Jonas Scheurich (scheurjs@studi.informatik.uni-stuttgart.de)
- Arno Schneider (schneiao@studi.informatik.uni-stuttgart.de)
- Sebastian Zillessen (zillessn@studi.informatik.uni-stuttgart.de)

## 8.2 Schlüsselpersonen des Auftraggebers

### 8.2.1 Prüfer

Prof. Dr. Thomas Ertl
+49 (0)711 685-88331
thomas.Ertl@vis.uni-stuttgart.de

### 8.2.2 Kunde des Projektes MISD

Dipl.-Inf. Christoph Müller
+49 (0)711 685-88626
christoph.mueller@vis.uni-stuttgart.de

### 8.2.3 Betreuer

Dr. Guido Reina
+49 (0)711 685-88627
guido.reina@vis.uni-stuttgart.de

Dipl.-Inf. Bernhard Schmitz
+49 (0)711 685-88302
bernhard.Schmitz@vis.uni-stuttgart.de

Dipl.-Inf. Michael Wörner
+49 (0)711 685-88214
michael.woerner@vis.uni-stuttgart.de

Dipl.-Inf. Daniel Kauker
+49 (0)711 685-88630
daniel.kauker@vis.uni-stuttgart.de

## 8.3 Schnittstelle zum Auftraggeber

Die Schnittstelle zum Auftraggeber bezieht sich auf verschiedene Aufgabenbereiche. Um diese getrennt zu behandeln, werden die Verantwortlichkeiten unter den Beteiligten der Projektleitung aufgeteilt.

### 8.3.1 Kommunikation zum Kunden

Die Kommunikation mit dem Kunden wird durch die Projektleiterin (Hanna Schäfer) durchgeführt. Hierbei sind die folgenden Aufgabenbereiche abgedeckt:

- Wöchentlicher Projektbericht über Planung, Fortschritt und Probleme
- Entgegennahme von Änderungswünschen
- Vereinbarung von Terminen zur Besprechung von Änderungen
- Gespräche über Verschiebung von Meilensteinen und Zeitplanung
- Eventuelle Gespräche über Einschränkung des Funktionsumfangs

### 8.3.2 Kommunikation zu den Betreuern

Die Kommunikation zu den Betreuern wird durch den Projektleiter (Sebastian Zillessen) durchgeführt. Hierbei sind die folgenden Aufgabenbereiche abgedeckt:

- Technische Probleme bei der Entwicklung
- Probleme mit der Rechtevergabe und den Zugriffen
- Probleme mit den zur Verfügung gestellten Servern und Tools
- Probleme mit der zur Verfügung stehenden *Teststrecke*

## 8.4 Schnittstellen zur eigenen Organisation

Innerhalb des Teams werden folgende Schnittstellen definiert:

### 8.4.1 Projektleiterin

Die Projektleiterin (Hanna Schäfer) ist projektübergreifende Schnittstelle für alle Probleme, Anliegen und Fragen. Dabei wird sie über folgende Wege angesprochen:

- Tagesordnungspunkte und Berichte des wöchentlichen Meetings
- Information durch (cc) im E-mail Verkehr
- Information durch täglichen Fortschrittsbericht
- Persönliche Informationen durch monatliches Einzelgespräch
- Direkten Kontakt bei Fragen und Problemen

Das Ziel dieser ausführlichen Kommunikation ist die leichtere Abschätzung der Projektsituation, welche sich aus dem Zustand der einzelnen Teammitglieder, der Verfassung von Untergruppen, dem zeitlichen Fortschritt und der Qualität des Entwicklungsstandes zusammensetzt. Aus diesen Informationen folgen Entscheidungen über eventuellen Eingriffsmaßnahmen, Umstrukturierungen, aber auch alltägliche organisatorischen Entscheidungen.

### 8.4.2 Zweiter Projektleiter

Der Projektleiter (Sebastian Zillessen) ist in Abwesenheit der Projektleiterin in allen Punkten dieser Schnittstelle verantwortlich. Im Normalfall ist seine Kommunikation jedoch spezialisiert auf folgende Fragestellungen:

- Zentrale Anlaufstelle für technische Probleme
- Schnittstelle bei fehlenden Zugriffsrechten
- Ansprechpartner bei Problemen mit der Projektleiterin

### 8.4.3 Qualitätsmanager

Der Qualitätsmanager (Arno Schneider) ist als Schnittstelle für alle Fragen der Qualitätssicherung verantwortlich. Seine Aufgaben sind:

- Erstellen von Richtlinien für die Programmierung
- Einführen in die Richtlinien und Beantwortung von auftauchenden Fragen
- Aufsetzen der Qualitätsbedingungen einzelner Abgaben
- Entgegennahme der Abgaben
- Prüfung und Annahme (bzw. Ablehnung) aller Abgaben

## 8.5 Berichtswesen

Neben den Dokumenten des Lieferumfangs, die bereits vor Projektabschluss ausgeliefert werden, werden folgende Dokumente an den Kunden/Betreuer ausgeliefert:

- Wöchentlicher Projektbericht der Projektleitung
- Monatlicher Bericht der Zeiterfassung
- Projektbericht jedes Projektmitglieds zum Projektabschluss

Intern wird ebenfalls der Fortschritt der Entwicklung erfasst. Dazu werden folgende Schritte durchgeführt.

- Zeiterfassung ist tagesaktuell (ab 8 Uhr für den Vortag)
- Täglicher Fortschrittsbericht über Tätigkeiten, Probleme und Fragen

# Kapitel 9

# Entwicklungsplan

## 9.1 Projektstrukturplan (Arbeitsgliederung)

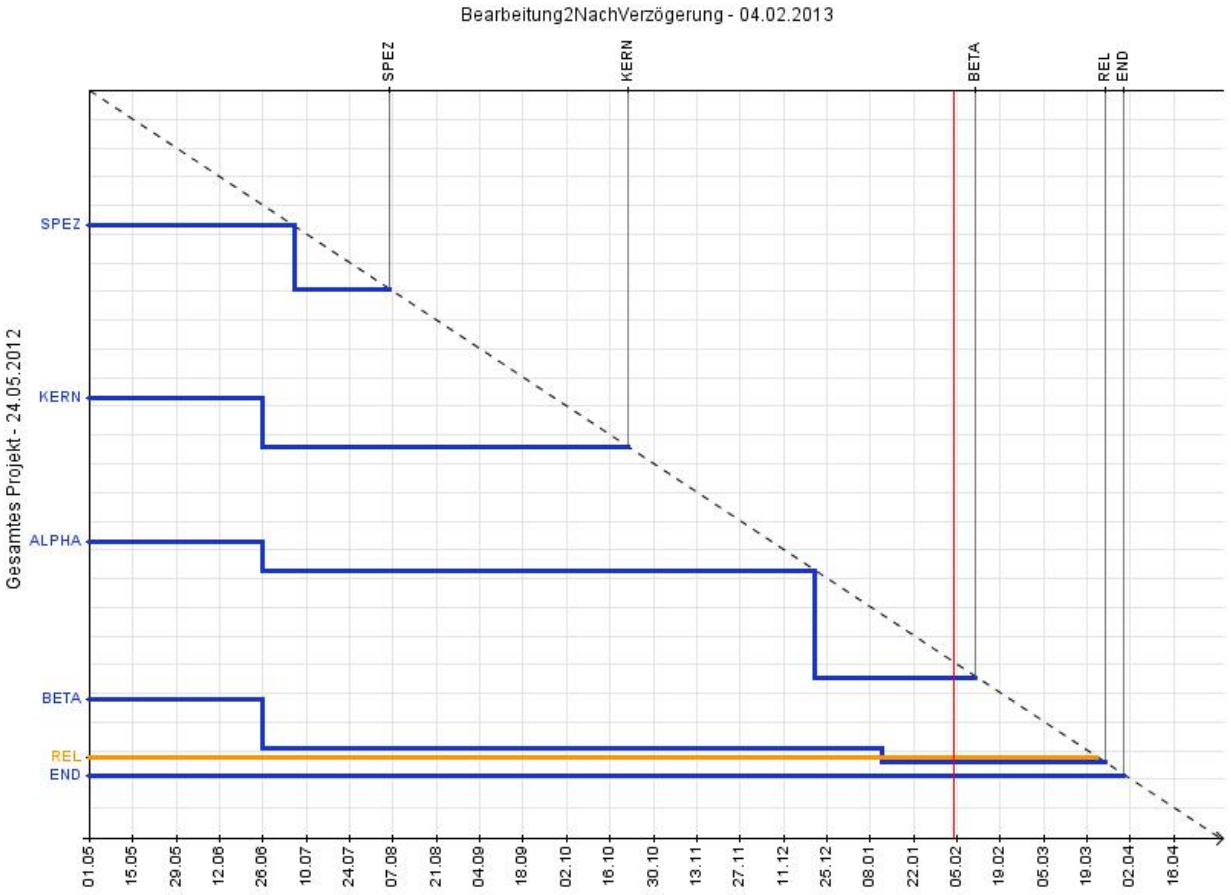
Jedem Arbeitspaket wird eine beliebige Anzahl an Teammitgliedern zugewiesen. Es ist möglich, dass in eine Kalenderwoche mehr als neun Entwickler eingeteilt sind, da die Arbeitszeit eines Entwicklers auch auf Arbeitspakete verteilt werden kann. Die folgenden Aufgaben eines Arbeitspaketes liegen in der Verantwortlichkeit von verschiedenen Personen:

| | |
|---|---|
| **Aufteilung der Arbeitspakete** | Projektleitung |
| **Sicherung der Termineinhaltung** | Hanna Schäfer |
| **Zentraler Ansprechpartner** | Hanna Schäfer |
| **Durchführung des Arbeitspaketes** | Jeweils verantwortliches Unterteam |
| **Abgabe der Dokumente** | Jeweils verantwortliches Unterteam |
| **Überprüfung der Umsetzung des Arbeitspaketes** | Arno Schneider |
| **Überprüfung des Dokumentationsprozesses** | Arno Schneider |

## 9.2 Termindrift-Diagramm

Um die Veränderungen an den Meilensteinen und Phasen des Projektes jederzeit aktuell im Projektplan verzeichnet zu haben, wurde ein Termindrift-Diagramm angelegt und im Folgenden dargestellt:

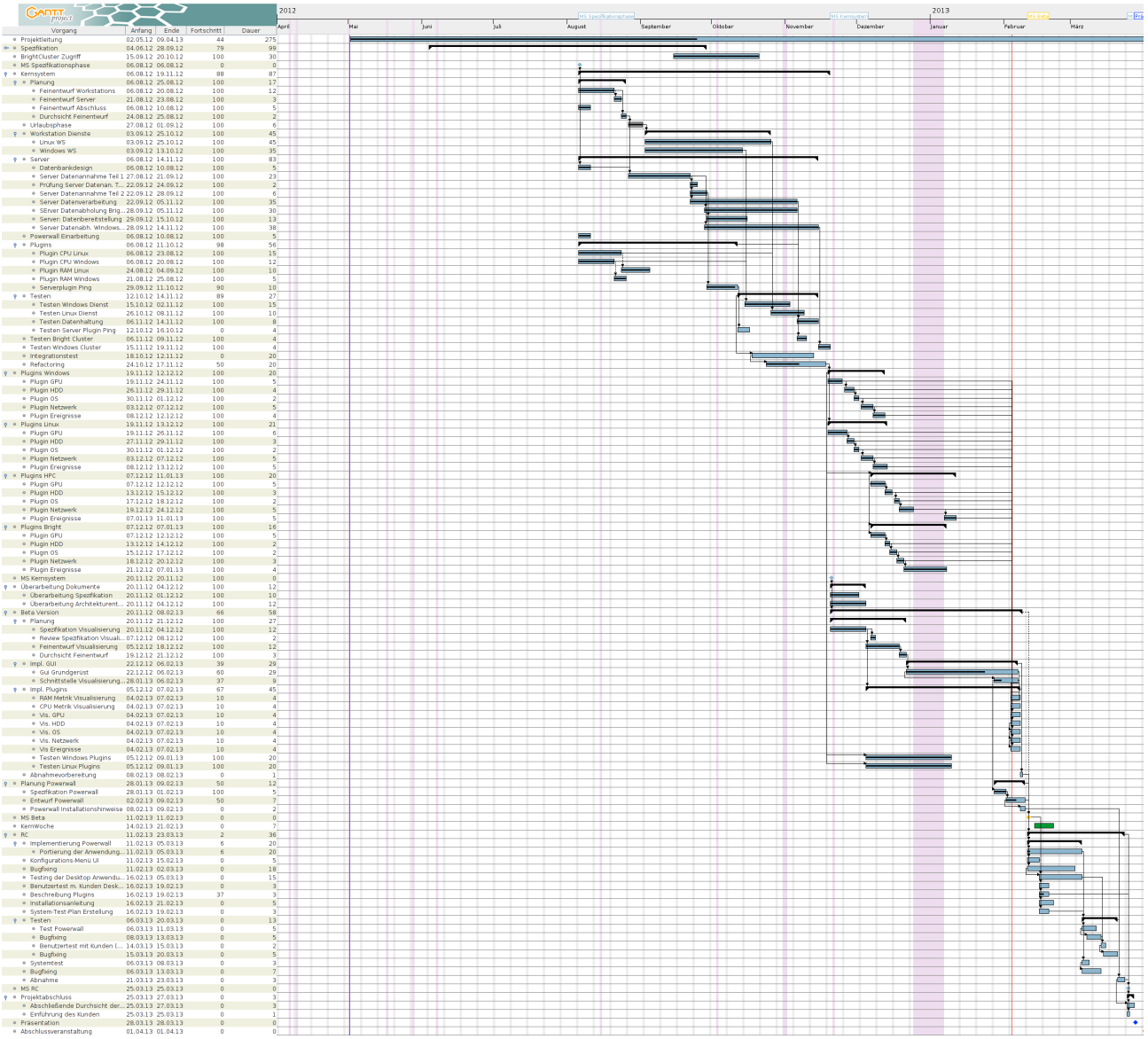
Die Phasen gliedern sich in der aktuellen Version des Projektplanes wie folgt:




- SPEZ 04.06.2012-28.09.2012 Spezifikation
- KERN 06.08.2012-17.11.2012 Kernsystem
- BETA 20.11.2012-11.01.2013 Beta-Version
- REL 12.01.2013-13.03.2013 Release Candidate
- END 13.03.2013- 31.03.2013 Projektabschluss

## 9.3 Gantt-Diagramm

Gemäß der Projektbeschreibung stehen dem neun köpfigen Entwicklungsteam **9x450 Stunden** für die Durchführung des Projektes zur Verfügung. Die gegebene Projektzeit wird in kleine Arbeitspakete eingeteilt, um die Aufwandsschätzung möglichst einfach zu gestalten. Die Entwicklung wird nach folgendem Gantt-Plan ablaufen.


GANTT project
2012
2013
April
Mai
Juni
Juli
August
September
Oktober
November
Dezember
Januar
Februar
März
MS Spezifikationsphase
MS Kernsystem
MS Beta
Vorgang
Anfang
Ende
Fortschritt
Dauer
Projektleitung 02.05.12 09.04.13 44 275
Spezifikation 04.06.12 28.09.12 79 99
BrightCluster Zugriff 15.09.12 20.10.12 100 30
MS Spezifikationsphase 06.08.12 06.08.12 0 0
Kernsystem 06.08.12 19.11.12 88 87
Planung 06.08.12 25.08.12 100 17
Feinentwurf Workstations 06.08.12 20.08.12 100 12
Feinentwurf Server 21.08.12 23.08.12 100 3
Feinentwurf Abschluss 06.08.12 10.08.12 100 5
Durchsicht Feinentwurf 24.08.12 25.08.12 100 2
Urlaubsphase 27.08.12 01.09.12 100 6
Workstation Dienste 03.09.12 25.10.12 100 45
Linux WS 03.09.12 25.10.12 100 45
Windows WS 03.09.12 13.10.12 100 35
Server 06.08.12 14.11.12 100 83
Datenbankdesign 06.08.12 10.08.12 100 5
Server Datenannahme Teil 1 27.08.12 21.09.12 100 23
Prüfung Server Datenan. T... 22.09.12 24.09.12 100 2
Server Datenannahme Teil 2 22.09.12 28.09.12 100 6
Server Datenverarbeitung 22.09.12 05.11.12 100 35
SErver Datenabholung Brig... 28.09.12 05.11.12 100 30
Server: Datenbereitstellung 29.09.12 15.10.12 100 13
Server Datenabh. Windows... 28.09.12 14.11.12 100 38
Powerwall Einarbeitung 06.08.12 10.08.12 100 5
Plugins 06.08.12 11.10.12 98 56
Plugin CPU Linux 06.08.12 23.08.12 100 15
Plugin CPU Windows 06.08.12 20.08.12 100 12
Plugin RAM Linux 24.08.12 04.09.12 100 10
Plugin RAM Windows 21.08.12 25.08.12 100 5
Serverplugin Ping 29.09.12 11.10.12 90 10
Testen 12.10.12 14.11.12 89 27
Testen Windows Dienst 15.10.12 02.11.12 100 15
Testen Linux Dienst 26.10.12 08.11.12 100 10
Testen Datenhaltung 06.11.12 14.11.12 100 8
Testen Server Plugin Ping 12.10.12 16.10.12 0 4
Testen Bright Cluster 06.11.12 09.11.12 100 4
Testen Windows Cluster 15.11.12 19.11.12 100 4
Integrationstest 18.10.12 12.11.12 0 20
Refactoring 24.10.12 17.11.12 50 20
Plugins Windows 19.11.12 12.12.12 100 20
Plugin GPU 19.11.12 24.11.12 100 5
Plugin HDD 26.11.12 29.11.12 100 4
Plugin OS 30.11.12 01.12.12 100 2
Plugin Netzwerk 03.12.12 07.12.12 100 5
Plugin Ereignisse 08.12.12 12.12.12 100 4
Plugins Linux 19.11.12 13.12.12 100 21
Plugin GPU 19.11.12 26.11.12 100 6
Plugin HDD 27.11.12 29.11.12 100 3
Plugin OS 30.11.12 01.12.12 100 2
Plugin Netzwerk 03.12.12 07.12.12 100 5
Plugin Ereignisse 08.12.12 13.12.12 100 5
Plugins HPC 07.12.12 11.01.13 100 20
Plugin GPU 07.12.12 12.12.12 100 5
Plugin HDD 13.12.12 15.12.12 100 3
Plugin OS 17.12.12 18.12.12 100 2
Plugin Netzwerk 19.12.12 24.12.12 100 5
Plugin Ereignisse 07.01.13 11.01.13 100 5
Plugins Bright 07.12.12 07.01.13 100 16
Plugin GPU 07.12.12 12.12.12 100 5
Plugin HDD 13.12.12 14.12.12 100 2
Plugin OS 15.12.12 17.12.12 100 2
Plugin Netzwerk 18.12.12 20.12.12 100 3
Plugin Ereignisse 21.12.12 07.01.13 100 4
MS Kernsystem 20.11.12 20.11.12 100 0
Überarbeitung Dokumente 20.11.12 04.12.12 100 12
Überarbeitung Spezifikation 20.11.12 01.12.12 100 10
Überarbeitung Architekturent... 20.11.12 04.12.12 100 12
Beta Version 20.11.12 08.02.13 66 58
Planung 20.11.12 21.12.12 100 27
Spezifikation Visualisierung 20.11.12 04.12.12 100 12
Review Spezifikation Visuali... 07.12.12 08.12.12 100 2
Feinentwurf Visualisierung 05.12.12 18.12.12 100 12
Durchsicht Feinentwurf 19.12.12 21.12.12 100 3
Impl. GUI 22.12.12 06.02.13 39 29
Gui Grundgerüst 22.12.12 06.02.13 60 29
Schnittstelle Visualisierung... 28.01.13 06.02.13 37 9
Impl. Plugins 05.12.12 07.02.13 67 45
RAM Metrik Visualisierung 04.02.13 07.02.13 10 4
CPU Metrik Visualisierung 04.02.13 07.02.13 10 4
Vis. GPU 04.02.13 07.02.13 10 4
Vis. HDD 04.02.13 07.02.13 10 4
Vis. OS 04.02.13 07.02.13 10 4
Vis. Netzwerk 04.02.13 07.02.13 10 4
Vis Ereignisse 04.02.13 07.02.13 10 4
Testen Windows Plugins 05.12.12 09.01.13 100 20
Testen Linux Plugins 05.12.12 09.01.13 100 20
Abnahmevorbereitung 08.02.13 08.02.13 0 1
Planung Powerwall 28.01.13 09.02.13 50 12
Spezifikation Powerwall 28.01.13 01.02.13 100 5
Entwurf Powerwall 02.02.13 09.02.13 50 7
Powerwall Installationshinweise 08.02.13 09.02.13 0 2
MS Beta 11.02.13 11.02.13 0 0
KernWoche 14.02.13 21.02.13 0 7
RC 11.02.13 23.03.13 2 36
Implementierung Powerwall 11.02.13 05.03.13 6 20
Portierung der Anwendung... 11.02.13 05.03.13 6 20
Konfigurations-Menü UI 11.02.13 15.02.13 0 5
Bugfixing 11.02.13 02.03.13 0 18
Testing der Desktop Anwendu... 16.02.13 05.03.13 0 15
Benutzertest m. Kunden Desk... 16.02.13 19.02.13 0 3
Beschreibung Plugins 16.02.13 19.02.13 37 3
Installationsanleitung 16.02.13 21.02.13 0 5
System-Test-Plan Erstellung 16.02.13 19.02.13 0 3
Testen 06.03.13 20.03.13 0 13
Test Powerwall 06.03.13 11.03.13 0 5
Bugfixing 08.03.13 13.03.13 0 5
Benutzertest mit Kunden (... 14.03.13 15.03.13 0 2
Bugfixing 15.03.13 20.03.13 0 5
Systemtest 06.03.13 08.03.13 0 3
Bugfixing 06.03.13 13.03.13 0 7
Abnahme 21.03.13 23.03.13 0 3
MS RC 25.03.13 25.03.13 0 0
Projektabschluss 25.03.13 27.03.13 0 3
Abschließende Durchsicht der... 25.03.13 27.03.13 0 3
Einführung des Kunden 25.03.13 25.03.13 0 1
Präsentation 28.03.13 28.03.13 0 0
Abschlussveranstaltung 01.04.13 01.04.13 0 0

# Anhang A

# Anhang

## A.1 Quellen

### A.1.1 Literaturverweise

**LLSE**:

Ludewig, J., H. Lichter: *Software Engineering – Grundlagen, Menschen, Prozesse, Techniken.* 2. Aufl., dpunkt.verlag Heidelberg, 2010. ISBN 978-3-89864-662-8

## A.2 Versionshistorie

### Version 0.1

| | |
|---|---|
| **Datum** | 28. Mai 2012 |
| **Änderungen** | Einleitung, Formale Grundlagen und Entwicklungsprozess verfasst |
| **Bearbeiter** | Sebastian Zillessen |

### Version 0.2

| | |
|---|---|
| **Datum** | 29. Mai 2012 |
| **Änderungen** | Deckblatt überarbeitet |
| **Bearbeiter** | Sebastian Zillessen |

## Version 0.4

| | |
|---|---|
| **Datum** | 29. Mai 2012 |
| **Änderungen** | Deckblatt überarbeitet |
| **Bearbeiter** | Sebastian Zillessen |

## Version 0.5

| | |
|---|---|
| **Datum** | 30. Mai 2012 |
| **Änderungen** | Namensgebung integriert |
| **Bearbeiter** | Sebastian Zillessen |

## Version 0.6

| | |
|---|---|
| **Datum** | 01. Juni 2012 |
| **Änderungen** | Richtlinien überarbeitet |
| **Bearbeiter** | Arno Schneider |

## Version 0.7

| | |
|---|---|
| **Datum** | 02. Juni 2012 |
| **Änderungen** | Änderungen in Kapitel 1, 2 und 4 |
| **Bearbeiter** | Sebastian Zillessen |

## Version 0.8

| | |
|---|---|
| **Datum** | 03. Juni 2012 |
| **Änderungen** | Zeilenabstand auf den 1,5 fachen Wert gesetzt |
| **Bearbeiter** | Sebastian Zillessen |

## Version 0.9

| | |
|---|---|
| **Datum** | 04. Juni 2012 |
| **Änderungen** | Kapitel 8 Projektorganisation, 5 Risiken und 7 Umgebung erstellt |
| **Bearbeiter** | Hanna Schäfer |

## Version 0.10

| | |
|---|---|
| **Datum** | 07. Juni 2012 |
| **Änderungen** | Richtlinien überarbeitet |
| **Bearbeiter** | Arno Schneider |

## Version 0.11

| | |
|---|---|
| **Datum** | 10. Juni 2012 |
| **Änderungen** | Richtlinien überarbeitet |
| **Bearbeiter** | Arno Schneider |

## Version 1.0

| | |
|---|---|
| **Datum** | 10. Juni 2012 |
| **Änderungen** | Kapitel 3 Leistungen und 9 Entwicklungsplan hinzugefügt |
| **Bearbeiter** | Hanna Schäfer |

## Version 1.1

| | |
|---|---|
| **Datum** | 11. Juni 2012 |
| **Änderungen** | Änderungen im Projektplan nach der Durchsicht |
| **Bearbeiter** | Hanna Schäfer |

## Version 1.2

| | |
|---|---|
| **Datum** | 11. Juni 2012 |
| **Änderungen** | Kompilierung |
| **Bearbeiter** | Sebastian Zillessen |

## Version 1.3

| | |
|---|---|
| **Datum** | 19. Juli 2012 |
| **Änderungen** | Änderungen nach Spezifikation, Termindrift |
| **Bearbeiter** | Hanna Schäfer |

## Version 1.4

| | |
|---|---|
| **Datum** | 24. Juli 2012 |
| **Änderungen** | Gantt-Diagramm |
| **Bearbeiter** | Hanna Schäfer |

## Version 1.5

| | |
|---|---|
| **Datum** | 30. September 2012 |
| **Änderungen** | Änderungen am Gantt-Diagramm wegen Projektverzögerungen.<br>Bisher keine Kürzungen notwendig.<br>Termin-Drift musste nicht geändert werden. |
| **Bearbeiter** | Hanna Schäfer, Sebastian Zillessen |

## Version 1.6

| | |
|---|---|
| **Datum** | 05. November 2012 |
| **Änderungen** | Änderungen am Gantt-Diagramm wegen Projektverzögerungen.<br>Verschiebung der Meilensteine.<br>Termin-Drift geändert.<br>Streichung der Interaktion mit der Powerwall mit dem Flight-Stick aus der Planung |
| **Bearbeiter** | Hanna Schäfer, Sebastian Zillessen |

## Version 1.7

| | |
|---|---|
| **Datum** | 04. Februar 2013 |
| **Änderungen** | Änderungen am Gantt-Diagramm wegen Projektverzögerungen.<br>Verschiebung der Meilensteine.<br>Termin-Drift geändert. |
| **Bearbeiter** | Hanna Schäfer, Sebastian Zillessen |