I. Risikoanalyse

Da die Entwicklung von Software naturgemäß mit Risiken verbunden ist, müssen diese im voraus identifiziert und Vermeidungsstrategien etabliert werden. Nur so können sie effektiv abgefangen werden um ein Scheitern des Projekts zu verhindern.

1. Organisation

Der Erfolg des Projekts steht und fällt mit der Planung und Verteilung der einzelnen Aufgaben. Folgende Risiken können auftreten:

a) Ungleichmäßige Aufteilung der Teilaufgaben kann einzelne Gruppenmitglieder überfordern und so Stress, Frustration und Zeitdruck erzeugen. Dies wirkt sich negativ auf Produktivität und Qualität der Aufgabenerledigung aus.

   Vermeidung: Die Aufgaben werden sorgfältig und nach Kompetenz der einzelnen Teammitglieder verteilt. Dazu bemüht sich jeder um eine realistische Selbsteinschätzung seiner Fähigkeiten. Die Verantwortlichen für die einzelnen Phasen verschaffen sich einen umfassenden Überblick über die jeweiligen Aufgaben und verteilen Teilaufgaben nach Kompetenz der einzelnen Mitglieder.
   Nach Erledigung der Aufgaben geht die Aufwandsanalyse in die zukünftige Verteilung ein.

2. Kommunikation

In arbeitsteiligen Projekten im Allgemeinen und der Softwareentwicklung im Speziellen besteht eine große Hürde in der erfolgreichen Kommunikation untereinander.

a) Mangelhafte Kommunikation der weiteren Vorgehensweise in der Aufgabenerledigung, sowie der Aufgabenverteilung führt zu unkoordiniertem Arbeiten und doppelter bzw. ungenügender Bearbeitung einzelner Teilaufgaben. Dies erzeugt Termindruck und wiederum Einbußen an Produktivität und Qualität.

   Vermeidung: Die gesamte Gruppe trifft sich wöchentlich zu einem Meeting, um das weitere Vorgehen abzusprechen. Um ausufernde Kommunikation zu vermeiden, stehen im Voraus Tagesordnungspunkte fest, die ebenfalls eine bessere Vorbereitung ermöglichen. Werden Aufgaben auf Kleingruppen verteilt, stellen diese ihre Ergebnisse bei diesen Treffen zur Diskussion. Zur späteren Einsicht und für den Fall des Fehlens einzelner Mitglieder wird ein Protokoll geführt.
   Weiterhin strukturiert ein internes Wiki die entstehenden Dokumente und die Verteilung der Aufgaben.

b) Meinungsverschiedenheiten und Streitigkeiten werden sich in der Gruppe nicht vermeiden lassen. Vor allem unter Zeitdruck sowie bei grundsätzlich unterschiedlichen Ansichten einzelner Themen entstehen interne Spannungen. Neben themenbezogenen Reibungen können auch persönliche Differenzen die Kommunikation stören.

Vermeidung: Alle Teammitglieder bemühen sich um einen konstruktiven, sachlichen und fachbezogenen Umgang miteinander. Im Kontext des Projekts, vor allem auf Meetings, ist es wichtig, Privates und Projektbezogenes zu trennen. Der Projektleiter sieht sich verantwortlich für die Disziplin auf den Meetings und das Festhalten an den Tagesordnungspunkten.
Außerdem kann ein informeller Raum in Form von Kaffeepausen oder ähnlichem weiteren Diskussionsbedarf ausräumen.
Werden Differenzen deutlich, sind unbeteiligte Mitglieder aufgerufen, zu moderieren und zu schlichten. Da klar ist, dass sich nicht alle Konflikte bis zum Konsens auflösen lassen, haben die jeweiligen Verantwortlichen das letzte Wort.

3. Zeitmanagement

Da die spezifischen Abgabetermine der einzelnen Aufgaben unbedingt einzuhalten sind und alle Gruppenmitglieder neben dem Praktikum umfängliche weitere Aufgaben im Studium zu erledigen hat, ist ein gutes Zeitmanagement unerlässlich.

a) Die Gefahr besteht in einem Verpassen von Terminen durch unerfüllte Teilaufgaben, unfertige Dokumente oder nicht getroffene Anforderungen.

   Vermeidung: Die Strategie zur Vermeidung dieses weit verbreiteten Risikos in der Softwareentwicklung besteht durch regelmäßige Absprache mit den Verantwortlichen der einzelnen Phasen und der Projektleitung. Die zuvor erwähnten Kommunikationswege, insbesondere das interne Wiki, erlauben einen allen einen guten Überblick über den Stand der Aufgabenbearbeitung. Es ist von allen aktuell und übersichtlich zu halten, verantwortlich ist hier ebenso die Projektleitung. Ebenso ist auf die zuvor erwähnten Richtlinien zur Kommunikation zu achten, um sich nicht in Debatten zu verlieren.
   Verbindliche Deadlines werden frühzeitig gesetzt um eventuelle Lücken noch rechtzeitig schließen zu können, es wird eher zu viel als zu wenig Zeit eingeplant.

b) Das Zeitmanagement kann nur dann einwandfrei funktionieren, wenn das ganze Team mitarbeitet. Sollten einzelne ihren Teil nicht beitragen leidet die Motivation der ganzen Gruppe, vor allem derer, die fehlende Mitarbeit kompensieren müssen.

   Vermeidung: Die Absprachen und Termine erfolgen verbindlich, jedes Teammitglied ist sich seiner Verantwortlichkeit bewusst. Sollten verteilte Aufgaben die Kompetenz einzelner überschreiten, wird der Projektleiter beziehungsweise der Verantwortliche der jeweiligen Phase rechtzeitig darüber informiert. Jeder bemüht sich hier um unbedingte Ehrlichkeit mit dem Team und sich selbst. Es ist schließlich wahrscheinlich, dass nicht immer absehbar sein wird, wie die Aufgaben zu verteilen sind. Darum erhöht rechtzeitiges Nachfragen im Problemfall die Transparenz im Stand der Bearbeitung, schafft die Möglichkeit zum Support durch andere und hilft Versagensängsten vorzubeugen.

4. Ausfall von Teammitgliedern

Auch ein gut geplantes Zeitmanagement kann vom Ausfall von Mitarbeitern beeinträchtigt werden. Der Zeitraum eines Semester ist hierbei groß genug, um eventuelle Krankheit oder den Leistungsabfall aus persönlichen oder psychischen Gründen einbeziehen zu müssen.

a) Dem Team sind durch die Termine enge Grenzen gesetzt. Fallen einzelne Mitglieder aus, sind die Deadlines eventuell nicht zu halten. Vor allem in kleinen Teams wie unseren könnte sich der Verlust eines wichtigen Mitarbeiters stark negativ auswirken.

   Vermeidung: Für den Erfolg des Projekts, müssen persönliche Probleme, die sich auf die Leistungsfähigkeit auswirken, sowie Krankheit, rechtzeitig von dem Betroffenen angesprochen werden. Die Gruppe wird sich dann bemühen, dem Einzelnen bei seiner Aufgabe unter die Arme zu greifen und gegebenenfalls hierfür außerordentliche Treffen in Kauf nehmen. Die Sicherheit des Rückhalts im Team soll die Motivation stärken und Stress mildern. Für den Fall einer kurzen Krankheit im Bereich von 1-2 Tagen ist es sinnvoll, Puffer im Zeitmanagement vorzusehen. Sollte ein Teammitglied für längere Zeit ausfallen, wird die Gruppe sich schnellstmöglich über das weitere Vorgehen einig.

5. Fehlentscheidungen von Verantwortlichen

Um Konflikte effektiv aufzulösen und Endlosdiskussionen zu vermeiden, wird den Verantwortlichen für die verschiedenen Phasen im Streitfall das letzte Wort gewährt. Das bedeutet aber auch, dass Fehlentscheidungen ihrerseits besonders kritisch sind.

a) Der Projektleiter hat im ganzen Team die größte Verantwortung. Sein Organisationstalent und seine Fähigkeit, den Überblick zu behalten, bestimmen maßgeblich den Projekterfolg. Trifft er eine Fehlentscheidung, oder kommt seiner Aufgabe nicht ernsthaft nach, ist der Erfolg des Projekts gefährdet. Dies gilt mit Einschränkungen auch für die Verantwortlichen der einzelnen Phasen der Entwicklung.

   Vermeidung: Fehler lassen sich nie ganz vermeiden. Deshalb sollte sichergestellt sein, dass in Streitfragen das "letzten Wort" nicht ohne jegliche Diskussion gesprochen wird. Konstruktive Kritik, auch an sich selbst, ist hier notwendig. Das Team achtet im Kleinen wie im Großen auf gute Vorbereitung und gewissenhafte Pflichterfüllung. Der Projektleiter ist sich seiner besonderen Pflichten bewusst. Er sorgt für einen guten Überblick über die Einzelheiten des Projekts, sowie die kompetenzgemäße Aufgabenverteilung und geringes Konfliktpotential.

6. Modellierung

Die Modellierung erscheint uns als einer der essentiellen Punkte in der Entwicklung unserer Software. Trotz absolvierten Java-Praktikums, haben wir großen Respekt vor den Anforderungen einer effektiven und verbindlichen Modellierung.

a) Die Modellierung erfolgt nach der Anforderungsanalyse und vor der Implementierung. Erfolgt die Anforderungsanalyse nicht sauber und möglichst vollständig, sind entstehende Fehler in der Modellierung später eventuell nicht mehr zu korrigieren. Unkontrolliertes „Hacken“ kann zu Fehlverhalten wie nicht passenden Schnittstellen zwischen den einzelnen Klassen führen und die Programmfunktion bis hin zum Scheitern des Projekts beeinträchtigen.

   Vermeidung: Vor der Anforderungsanalyse hat eine gründliche Recherche zu erfolgen. Erst dann ist ein erfolgreiches Kundengespräch möglich. Prägnante Fragestellungen

werden von der Verantwortlichen für Recherche herausgearbeitet und mit dem Team abgesprochen. In der Modellierung wird ein besonderes Augenmerk auf die Schnittstellen gelegt. Sie sind absolut verbindlich. Sollten aus technischen Gründen Änderungen am Modell notwendig werden, bedürfen diese der Rücksprache mit dem gesamten Team. Dies ist jedoch ein absoluter Ausnahmefall.
Fragen zu Modell und Programmarchitektur werden sollten frühzeitig gestellt und Unklarheiten geklärt werden. In der Implementierung hilft sauber dokumentierter Code, die Funktion fremden Programmcodes zu überblicken. Da wir eine Erweiterung für eine schon bestehende Software entwickeln, wählen wir den Ansatz der outside-in Modellierung, das heißt alle Programmfunktionen werden ausgehend von den bestehenden Schnittstellen entwickelt, um spätere Imkompatibilitäten zu vermeiden.

7. Nachträgliche Änderungen

Es ist unwahrscheinlich, dass alle Anforderungen im Voraus analysiert werden können. Im Gegenteil sind später auftretende Detailfragen und nachträgliche Veränderungen in der Softwareentwicklung die Regel und oft Ursache von Verzögerungen.

a) Je weiter der Entwicklungsprozess fortgeschritten ist, desto schwerer wiegen grundsätzliche Änderungen. Problemen, die aus falsch verstandenen Aufgabenstellungen und ungenügendem Kundenkontakt ergeben, muss deshalb frühzeitig begegnet werden.

   Vermeidung: In den frühen Phasen des Projekts werden grundsätzliche Designfragen gemeinsam diskutiert. Offen bleibende Fragen über weitere Anforderungen werden frühzeitig mit dem Kunden abgesprochen, jedoch möglichst schon in der Anforderungsanalyse geklärt. Bei schon stehender Modellierung wird diese zuerst nachgezogen, bevor Veränderungen am Code stattfinden. Weiterhin helfen Prototypen der frühen Einschätzung der Leistungsfähigkeit und Korrektheit des Programms.

8. Implementierung

Da die Kenntnisse und Fähigkeiten im Implementieren in unserer Gruppe durchaus ungleichmäßig verteilt sind, muss auch daraus entstehenden Problemen und Risiken begegnet werden.

a) Trotz vorhandenen Java-Praktikums sind nicht alle Teammitglieder kompetente Programmierer. Darunter kann die Qualität der Software sowie die Einhaltung der Termine leiden.

   Vermeidung: Die sich aus dem Modell ergebenden Klassen und Methoden werden nach Kompetenz der Teammitglieder verteilt. Die Strategie des Pair-Programmings soll außerdem die Richtigkeit und Funktion des Codes absichern. Paare werden dabei nach Kompetenzen besetzt, „schwächere“ und „stärkere“ Implementierer zusammen. Der Verantwortliche für Tests erstellt ein konsistentes Testkonzept und kontrolliert dessen Einhaltung sowie die ordnungs- und modellgemäße Funktion des Codes.
   In Hinsicht auf die spätere Wartbarkeit der Software strebt das Team einen sauberen und gut strukturierten Code an, der unnötige und schwer zu verstehende Hacks so gut es geht vermeidet. Es wird sich hierbei nach den Java Styling Guides richten.

9. Dokumentation

Da Software immateriell ist, und das Fortschreiten des Entwicklungsprozesses daher nicht immer offensichtlich, ist eine gute Dokumentation unerlässlich. Auch die Korrekt und die spätere Wartbarkeit des Produkts kann nur so garantiert werden.

a) Die Gliederung des Praktikums und die Grundregeln der Softwaretechnik fordern eine saubere und lückenlose Dokumentation des Projektfortschritts.

   Für die Vermeidung von Lücken in der Dokumenterstellung überwacht der Verantwortliche für Qualitätsmanagement und Dokumentation. Er prüft alle erstellten Dokumente und moniert eventuelle Korrekturen und Leerstellen. Neben den projektrelevanten Dokumenten, werden auch die auf den Meetings getroffenen Entscheidungen und die zu ihnen führenden Argumente dokumentiert, um Transparenz zu gewährleisten.

b) In der Implementationsphase kommt der Dokumentation große Bedeutung zu. Sie beeinflusst hier die Wartbarkeit, Pflege und Verständlichkeit des Softwareprodukts in besonderem Maße. Falls andere Teammitglieder nicht funktionierenden Code debuggen müssen, bedeutet schlecht dokumentierter Code erhöhten Zeitaufwand und Frust.

   Vermeidung: Alle bemühen sich um klar dokumentierten, strukturierten Code, der auch im Krankheitsfall von anderen überblickt werden kann. Die Kommentierung von Klassen und Methoden, sowie die Nutzung von Javadoc sind obligatorisch. Schwer verständliche Codefragmente werden zusätzlich Inline kommentiert.

10. Technische Ausfälle

Elektronisch gespeicherte Daten sind immer vom Verlust durch technische Ausfälle bedroht.

a) Projektrelevante, wichtige Dokumente werden redundant durch die bearbeitenden Mitglieder und im svn gespeichert. Die technische Assistentin stellt die Funktion von svn und Wiki als Orte gemeinsamer Datenablage sicher und nimmt, vor allem in der Implementationsphase, regelmäßige Projektbackups vor.

II. Verteilung der Rollen

Projektleiter	Robert Heinecke
Technische Assistentin	Katharina Hößel
Verantwortliche für Recherche	Katrin Karlisch
Verantwortlicher für Modellierung	Sascha Vinz
Verantwortlicher für Tests	Sascha Vinz
Verantwortlicher für Implementierung	Sebastian Lippert
Verantwortlicher für QM & Dokumentation	Martin Brümmer