Arbeitspakete – Übersicht

1.Projektmanagement

Arbeitspaket Nummer:	1.1
Disziplin:	Projektmanagement
Name:	Risk Management
Verantwortlicher:	Steffi

Arbeitspaket Nummer:	1.2
Disziplin:	Projektmanagement
Name:	Software Development
Verantwortlicher:	Irmi

Arbeitspaket Nummer:	1.3
Disziplin:	Projektmanagement
Name:	Projektanpassung
Verantwortlicher:	Irmi

Arbeitspaket Nummer:	1.4
Disziplin:	Projektmanagement
Name:	Website
Verantwortlicher:	Steffi

Arbeitspaket Nummer:	1.5
Disziplin:	Projektmanagement
Name:	Kommunikation
Verantwortlicher:	Kathi

2.Business Modelling

Arbeitspaket Nummer:	2.1
Disziplin:	Business Modelling
Name:	Einführung und Vision
Verantwortlicher:	Viola

Arbeitspaket Nummer:	2.2
Disziplin:	Business Modelling
Name:	UseCases
Verantwortlicher:	Zona

Arbeitspaket Nummer:	2.3
Disziplin:	Business Modelling
Name:	Glossary
Verantwortlicher:	Steffi

3.Requirements

Arbeitspaket Nummer:	3.1
Disziplin:	Requirements
Name:	Requirement Management
Verantwortlicher:	Zona, Kathi

Arbeitspaket Nummer:	3.2
Disziplin:	Requirements
Name:	Requirement Specification
Verantwortlicher:	Viola

4.Analyse & Design

Arbeitspaket Nummer:	4.1
Disziplin:	Analyse & Design
Name:	Domain Modelling
Verantwortlicher:	Irmi

Arbeitspaket Nummer:	4.2
Disziplin:	Analyse & Design
Name:	Dynamische Modellierung
Verantwortlicher:	Kathi

5. Implementierung

Arbeitspaket Nummer:	5.1
Disziplin:	Implementierung
Name:	Prototyp
Verantwortlicher:	Viola

Arbeitspaket Nummer:	5.2
Disziplin:	Implementierung
Name:	GUI-Schicht
Verantwortlicher:	Irmi

Arbeitspaket Nummer:	5.3
Disziplin:	Implementierung
Name:	Logische Schicht
Verantwortlicher:	Kathi, Zona

Arbeitspaket Nummer:	5.4
Disziplin:	Implementierung
Name:	Mitarbeiterverwaltung
Verantwortlicher:	Kathi, Zona

Arbeitspaket Nummer:	5.5
Disziplin:	Implementierung
Name:	Kundenverwaltung
Verantwortlicher:	Steffi, Kathi

Arbeitspaket Nummer:	5.6
Disziplin:	Implementierung
Name:	Kassenfunktion
Verantwortlicher:	Viola, Irmi

Arbeitspaket Nummer:	5.7
Disziplin:	Implementierung
Name:	Lagerbestände
Verantwortlicher:	Steffi

Arbeitspaket Nummer:	5.8
Disziplin:	Implementierung
Name:	Account
Verantwortlicher:	Viola, Zona

Arbeitspaket Nummer:	5.9
Disziplin:	Implementierung
Name:	Terminplanung
Verantwortlicher:	Steffi, Irmi

Arbeitspaket Nummer:	5.10
Disziplin:	Implementierung
Name:	Administratorverwaltung
Verantwortlicher:	Kathi, Zona

6. Test

Arbeitspaket Nummer:	6.1
Disziplin:	Test
Name:	Test Plan
Verantwortlicher:	Steffi, Irmi

Arbeitspaket Nummer:	6.2
Disziplin:	Test
Name:	Test Cases + Evaluation
Verantwortlicher:	Alle

Arbeitspaket Nummer:	6.3
Disziplin:	Test
Name:	Virtuelle Maschine
Verantwortlicher:	Zona

7.Configuration & Change Management

Arbeitspaket Nummer:	7.1
Disziplin:	Configuration & Change Management
Name:	Werkzeuge – GIT
Verantwortlicher:	Viola, Steffi

Arbeitspaket Nummer:	7.2
Disziplin:	Configuration & Change Management
Name:	Werkzeuge – Maven
Verantwortlicher:	Irmi

8.Development

Arbeitspaket Nummer:	8.1
Disziplin:	Development
Name:	Überführung in Anwendung
Verantwortlicher:	Alle

Arbeitspaket: Risk Management (AP 1.1)

Beschreibung:
Hier steht die Erstellung des Risk Management Plans im Vordergrund, eine Auflistung aller möglichen Risiken des Projekts (nicht der Software selbst!). Gleichzeitig wir zu jedem eventuellem Risiko ein Plan erarbeitet, der beim Eintritt des Risikos befolgt wird. Der Risk Management Plan soll nach einer Abstimmung durch das Team nicht mehr verändert werden (höchstens erweitert).

Ziel:
Das Ziel ist es, Risiken zu erkennen und wenn möglich zu umgehen. Durch eine Festlegung des Vorgehens vor dem Auftritt eines bestimmten Risikos soll die Sicherheit und Stabilität des Projekts unterstützt werden.

Risiken:
Risiken bestehen hier höchstens in einem zu knapp gefassten oder lückenhaftem Risk Management Plan. Beim Auftreten von nicht geplanten Risiken muss der Risk Management Plan um die entsprechenden Schritte erweitert werden.

Eingangsdokumente: Vision Dokument
Ausgangsdokumente: Risk Management Plan

Beginn: Inception
Dauer: 3h
Ende: Inception

Arbeitspaket: Software Development (AP 1.2)

Beschreibung:
Unverbindliche, ungenaue Abschätzung der Ressourcen und des Aufwands. Der Software Development Plan sammelt alle Informationen, die notwendig sind, um das Projekt zu verwalten. Er wird in der Anfangsphase entwickelt und über das gesamte Projekt hin beibehalten.

Ziele:
Das Ziel des Software Developments ist die Überwachung der Prozesse für die Softwareentwicklung. Die einzelnen Verfahren und jede Aktivität soll verfolgt werden können. (Projektplan, Organisation, Ressourcen).

Risiken:
Eine zu pessimistische bzw. zu optimistische Schätzung, kann vor allem am Ende des Projekts für Probleme sorgen. Deshalb sollte die Abschätzung nach gemeinsamer Besprechung erfolgen und muss im Ernstfall angepasst werden.

Eingangsdokumente: Vision Dokument
Ausgangsdokumente: Software Development Plan

Beginn: Inception
Dauer: 3h
Ende: Inception

Arbeitspaket: Projektanpassung (AP 1.3)

Beschreibung:
Der Ablauf des Projekts wird laut dem Rational Unified Process vorgenommen. Zu diesem Zweck müssen Phasen und Iterationen genau an das Projekt „Mosti“ angepasst werden. Des weiteren werden Arbeitspakete erstellt und Aufgaben an die verschiedenen Teammitglieder verteilt.
Der resultierende Development Case wird früh in der Anfangsphase erstellt und während des gesamten Projekts bei Bedarf aktualisiert.

Ziele:
Das Ziel des Development Cases ist es, das Projekt genau an die bestehenden Gegebenheiten anzupassen, um eine optimale Projetdurchführung zu unterstützen.

Risiken:
Eine zu ungenaue Anpassung an das Projekt kann vor allem zum Ende hin zu Komplikationen führen. Deshalb ist es auch während dem Projekt wichtig, den Verlauf an die neuen Gegebenheiten und Erkenntnisse anzupassen.

Eingangsdokumente: Vision Dokument, Software Development Plan
Ausgangsdokumente: Development Case

Beginn: Inception
Dauer: 35h
Ende: Transition

Arbeitspaket: Website (AP 1.4)

Beschreibung:
Wir erstellen eine Website, in der unser Projekt und das Team kurz beschrieben werden und die wichtigsten Meilensteine zum Projektverlauf erscheinen. Zum Ende des Projekts soll auf der Website der Download der Software ermöglicht werden.

Ziele:
Mit dieser Website wird es Außenstehenden ermöglicht, den Ablauf unseres Projekts nachzulesen. Des Weiteren dient die Website als Zusammenfassung der Arbeitsergebnisse für alle Teammitglieder und eventuellen Neueinsteigern.

Risiken:
Durch viele bereitgestellte Online-Vorlagen sollte es kein Schwierigkeit sein, die Website zu erstellen. Nur die Bereitstellung der Downloadfunktion könnte durch die geringe Erfahrung der Teammitglieder scheitern.

Eingangsdokumente: Auswahl an erstellten Artefakten
Ausgangsdokumente: Bilder und Dokumentationen für Außenstehende

Beginn: Transition
Dauer: 12h

Ende: Transition

Arbeitspaket: Kommunikation (1.5)

Beschreibung:
Es werden innerhalb des Projekts mehrere Sitzungen gehalten, um den aktuellen Stand des Projekts zu besprechen und zu diskutieren. Diese erfolgen entweder durch persönliche Treffen oder via Skype, wenn einzelne Teammitglieder verhindert sind. Zu jeder Sitzung wird ein Protokoll erstellt.

Ziel:
Ziel der Sitzungen ist es, alle Mitglieder auf den aktuellen Stand zu bringen. Es werden die Fortschritte, Probleme und Änderungsvorschläge der Teammitglieder besprochen und diskutiert. Des Weiteren können hier auch ggf. Fragen oder Anträge geklärt werden. Für die Sitzungen werden ca. 2 Stunden pro Woche veranschlagt.

Risiken:
Die Meinungen der Teammitglieder bezüglich gewissen Themen oder Aufgabe können stark auseinandergehen. Doch genau wegen diesem Risiko werden die Sitzungen jede Woche veranstaltet, um bei Problemen schnell eine gemeinsame Lösung zu finden.

Eingangsdokumente: Gespräche, Dokumente
Ausgangsdokumente: Sitzungsprotokoll

Beginn: ab Projektplanung
Dauer: ca. 2h pro Woche
Ende: bei Projektende

Arbeitspaket: Einführung und Vision (AP 2.1)

Beschreibung:
Als erster Schritt stehen das Zusammenfinden zu einer Gruppe und die Einigung auf ein Projektthema an. Jeder kann Vorschläge einbringen, die dann diskutiert und in ihrem Umfang und Aufwand abgeschätzt werden. Die Abstimmung auf ein Thema erfolgt nach demokratischen Grundlagen.

Ziele:
Durch die Diskussion eines Projektthemas und die Erstellung des Vision Dokuments kann sichergestellt werden, dass alle Teammitglieder denselben Grobentwurf vor Augen haben.

Risiken:
Es muss sichergestellt werden, dass alle Teammitglieder dieselbe Vorstellung vom Projektthema haben. Des Weiteren muss der Umfang des Projekts an Personen- und Zeitressourcen angepasst werden, um das Projekt in den vorgegebenen Maßen umsetzen zu können.

Eingangsdokumente: ---
Ausgangsdokumente: Vision Dokument

Beginn: Inception
Dauer: 10h
Ende: Inception

Arbeitspaket: UseCase (2.2)

Beschreibung:
Während der Inception werden die Namen der meisten UseCases identifiziert. Sie beschreiben, wie ein System den Benutzer in seinen Aufgaben unterstützt. Die UseCases werden in späteren Iterationen erweitert, ergänzt und in „fully-dressed“-Format ausgearbeitet. Dieser Schritt erfolgt in enger Zusammenarbeit mit Mitarbeitern eines Mostereibetriebs.

Ziele:
Durch das Schreiben der UseCases kann der Kunde (hier: die Mitarbeiter der Mosterei) seine spezifischen Wünsche und Anforderungen an die Software mit einfachen Mitteln an die Programmierer weitergeben.

Risiken:
Einige wichtige Anforderungen an das Projekt könnten übersehen werden (lückenhafte UseCases) oder werden von den Mostereimitarbeitern nicht genannt, da es sich für diese um Selbstverständlichkeiten handelt. Deshalb ist ein ständiger Austausch von Programmierern und Mostereimitarbeitern nötig, um ggf. fehlende oder geänderte Anforderungen an das Projekt anzupassen.

Eingangsdokumente: Vision Dokument, Berichte der Mitarbeiter
Ausgangsdokumente: UseCase Model

Beginn: Inception
Dauer: 40 Stunden
Ende: Elaboration

Arbeitspaket: Glossary (2.3)

Beschreibung:
Das Glossary besteht aus wichtigen Termini und dessen Definitionen, die im Laufe des Projekts immer wieder verwendet werden. Das Glossary wird von Anfang des Projekts bis zum Ende laufend erweitert.

Ziele:
Durch das Glossary wird sichergestellt, dass als Teammitglieder vom Selben sprechen. Ebenso dient das Glossary als Grundlage für Außenstehende oder Neueinsteiger, sich in der Terminologie des Projektes schnell zurechtzufinden.

Risiken:
Ein lückenhaftes oder unregelmäßig aktualisiertes Glossary kann zu Verwirrungen und Missverständnissen innerhalb der Gruppe führen.

Eingangsdokumente: Mitarbeiterberichte
Ausgangsdokumente: Glossary

Beginn: Inception
Dauer: 4h
Ende: Transition

Arbeitspaket: Requirement Management (AP 3.1)

Beschreibung:
Mit der Erstellung der Requirement Management Plans wird eine Anleitung verfasst, die auf die Änderung von Anforderungen zugeschnitten ist. Es wird geklärt, wie Anforderungsänderungen umgesetzt werden und nach welchen Kriterien Anforderungen geändert werden sollen.

Ziele:
Ziel des Requirement Management ist es, eine einheitliche Richtlinie bei der Änderung oder Erweiterung von Anforderungen während des gesamten Projekts beizubehalten. So können Änderungen schnell realisiert und kontrolliert durchgeführt werden.

Risiken:
Nicht genau spezifizierte Anforderungsänderungen oder später hinzugefügte Anforderungen werden eventuell nicht getestet und können so zu Risiken für die Software führen. Deshalb muss genau dokumentiert werden, welche Anforderungen sich geändert haben und die spezifischen Tests müssen angepasst werden.

Eingangsdokumente: ---
Ausgangsdokumente: Requirement Management Plan

Beginn: Inception
Dauer: 3h
Ende: Inception

Arbeitspaket: Requirement Specification (AP 3.2)

Beschreibung:
Auf Grundlage der Anforderungen aus dem UseCase Model werden nun die tatsächlich umsetzbaren Anforderungen an das Projekt spezifiziert. Vor allem sollen hier die nicht-funktionale Anforderungen durch Gespräche mit den Mostereimitarbeitern herausgefiltert werden.

Ziele:
Durch die Spezifizierung der Anforderungen wird versucht, die Vision von Programmierern und späteren Nutzern, die Mitarbeiter des Mostereibetriebs, auf einen gemeinsamen Nenner zu bringen. Die Software Requirement Specification dient als Grundlage für die zu absolvierenden Tests.

Risiken:
Alle Anforderungen an die Software müssen zurück verfolgbar sein (Tracability). Nicht spezifizierte Anforderungen bedeuten nicht getestete Anforderungen, was ein hohes Risiko für die Software darstellen kann. Des Weiteren wird es im späteren Verlauf des Projekts immer schwerer möglich, nichtfunktionale Anforderungen zu ändern.

Eingangsdokumente: UseCase Model
Ausgangsdokumente: Software Requirement Specification

Beginn: Inception
Dauer: 30h
Ende: Elaboration

Arbeitspaket: Domain Analyse (AP 4.1)

Beschreibung:
Auf Grundlage der UseCases und der Besprechungen mit den Mitarbeitern soll eine erste Abbildung des Problembereichs erfolgen. Beginnend mit den grundlegenden Funktionen soll das Domänenmodell Schritt für Schritt um weitere Funktionen ergänzt werden. Dabei entwirft zunächst jedes Teammitglied sein eigenes Model, welche später in der Gruppe diskutiert werden.

Ziele:
Durch die Modellierung mehrerer, teils völlig verschiedener Varianten des Problembereichs soll aus den erstellten Möglichkeiten die beste herausgearbeitet und zusammengestellt werden. Das Domain Model dient als grobe Vorlage des Softwareentwurfs.

Risiken:
Da es bei der Modellierung des Problembereichs kein „Richtig“ und „Falsch“ gibt, kann die Einigung auf ein gemeinsames Modells erschwert werden. Ebenso können fehlerhafte Domainmodelle vor allem in späteren Phasen des Projektes zu Komplikationen und Problemen führen.

Eingangsdokumente: Vision Dokument, UseCase Model
Ausgangsdokumente: Domain Model

Beginn: Elaboration
Dauer: 26h
Ende: Construction

Arbeitspaket: Dynamische Modellierung (AP 4.2)

Beschreibung:
Nach dem Erstellen des Domainmodells ist das Zusammenwirken der einzelnen konzeptionellen Klassen bestimmt. Dieses kann nun in der dynamischen Analyse spezifiziert werden. Auch hier sollen die Diagramme erst wieder von einzelnen Mitglieder erstellt werden und dann in gemeinsamer Diskussion auf ein Modell geeinigt werden.

Ziele:
Durch die Erstellung von Systemsequenzdiagrammen, Contracts und eventuell Aktivitäts- und Zustandsdiagrammen soll das Verständnis des Wirkens und des Zusammenwirkens der einzelnen Packages untereinander gebessert werden.

Risiken:
Vor allem ungenau definierte Schnittstellen zwischen mehreren Packages können im Laufe des Projekts zu Problemen und Missverständnissen führen.

Eingangsdokumente: Domain Model, Use Cases
Ausgangsdokumente: Aktivitätsdiagrame, Zustandsdiagrame, Systemsequenzdiagram, Contracts für Systemoperationen

Beginn: Elaboration
Dauer: 25h
Ende: Construction

Arbeitspaket: Prototype (AP 5.1)

Beschreibung:
Laut RUP-Prozess soll auch die Software Mosti in einem inkrementellen, iterativen Prozess wachsen. Dennoch soll jede 1-2 Wochen ein kompilierbares Programm erstellt werden, dass der Kunde – der Mostereimitarbeiter – testet und so seine gewünschten Anforderungen zeitnah überprüfen kann.

Ziele:
Durch den regelmäßigen Kontakt mit den Mostereimitarbeitern und ihr Feedback zur Software können unterschiedliche Ansichten oder fehlende Anforderungen schnell erkannt werden und so die entsprechenden Schritte unternommen werden.

Risiken:
Auch hier gewonnene Erkenntnisse über Anforderungen und Anforderungsänderungen müssen unbedingt in die Anforderungsliste mitaufgenommen werden – nach den Maßstäben des Requirement Management Plans.

Eingangsdokumente: Requirement Management Plan, Code
Ausgangsdokumente: Funktionierender Code, Anforderungsänderungen

Beginn: Elaboration
Dauer: 14h
Ende: Construction

Arbeitspaket: GUI-Schicht (AP 5.2)

Beschreibung:
Es soll Schritt für Schritt eine Benutzeroberfläche für die Mosti Software erstellt werden. Hierbei stützen wir uns auf vorgefertigte Java-Klassen (java.awt). Zuerst soll ein Frame für das Hauptmenü erstellt werden, welches durch Menu-Items mit weiteren Unter-Frames verbunden ist. Diese Unter - Frames werden in späteren Iterationen entwickelt und ausgebaut.

Ziele:
Gemäß dem GRASP-Pattern Controller, soll auch hier die GUI-Schicht selbst keine Logik enthalten, sondern lediglich die Befehle an geeigneten Controller delegieren. Das Wechseln zwischen den einzelnen Unterframes (z.B Abrechnung, Terminplan) soll wiederum mit Swing - Komponenten möglich sein. Die einzelnen Elemente sollen sichtbar und leicht zu finden dargestellt werden.

Risiken:
Ein mögliches Risiko wäre die Verbindung der GUI-Schicht mit der Logik des Programms. Die GUI-Schicht soll allerdings nur Benutzeranfragen an die Controller der Logikschicht weiteleiten. Deshalb wäre eine Auslagerung der GUI-Elemente in eine eigenes Package und einer dünnen Schnittstelle zu den Logikpackages eine wichtige Präventionsmaßnahme.

Eingangsdokumente: System-Sequenzdiagramme, Domainmodell
Ausgangsdokumente: Architektur-Plan

Beginn: Elaboration
Dauer: 13h
Ende: Construction

Arbeitspaket: Logische Schicht (AP 5.3)

Beschreibung:
In diesem Arbeitspaket wir die logische Schicht für das System entwickelt. Da einzelne Aufgabenbereiche der Software in eigene Packages zusammengefasst werden, fungiert die logische Sicht als Vermittler zwischen den Klassenpackages. Die logische Sicht soll in den einzelnen Iterationen inkrementell um die Packages erweitert und Schnittstellen angepasst werden

Ziele:
Die logische Sicht soll als Verbindung zwischen den – von verschiedenen Teammitgliedern – erstellten Klassen und Arbeitspaketen dienen. Sie unterstützt eine geringe Kopplung und dünne Schnittstellen. Durch den Gebrauch der logischen Schicht als Vermittler sollen die anderen Packages voneinander unabhängiger werden.

Risiken:
Ein Risiko wäre einerseits ein hohe Kopplung, dass die logische Sicht z.B. auf Informationen einer Klasse zugreift, die nicht zwingend benötigt werden oder auch anderweitig zur Verfügung gestellt werden könnten. Dieses Problem kann man durch eine saubere Ausarbeitung des Designmodells verhindern. Ein anderes Risiko wäre die Inkompatibilität der Schnittstellen zu den verschiedenen Klassen. Im schlimmsten Fall müssen ganze Klassenstrukturen neu ausgerichtet werden. Deshalb ist ein Austausch über die Art der Schnittstellen (z.B. mittels System-Sequenzdiagrammen) dringend erforderlich.

Eingangsdokumente: Designmodell, SSD, Interaktionsdiagramme
Ausgangsdokumente: erweitertes/ angepasstes Designmodell

Beginn: Elaboration
Dauer: 13h
Ende: Construction

Arbeitspaket: Mitarbeiterverwaltung (AP 5.4)

Beschreibung:
Zum Arbeitspaket der Mitarbeiterverwaltung gehört die Erstellung von Mitarbeiterklassen (Name, Adresse, ID etc.) sowie die Verwaltung und Speicherung derselben in einem persistenten Dokument, das auch die Laufzeit des Programms überdauert. Zudem sollen
Schnittstellen entwickelt werden, um von anderen Klasen aus auf wichtige Dateien der Mitarbeiter zugreifen zu können.

Ziele:
Mit dem Package der Mitarbeiter steht für das Programm eine geschlossene Klasse von Daten zur Verfügung. Sie ermöglicht eine kompakte Eingabe und Übersicht über die Angestellten und dient als Grundlage für weitere Features, wie zum Beispiel der Schichtplanerstellung.

Risiken:

Der Zugriff auf die gespeicherten Mitarbeiterdaten und das Abspeichern der Daten im richtigen Format könnte aufgrund mangelnder Erfahrung zu Problem werden. Wichtig ist hierbei die Einigung auf ein Speichermedium (Datenbank oder Textdatei) und die Tests zur richtigen Abspeicherung und Auslesung der Daten.

Eingangsdokumente: Domainmodell, SSD
Ausgangsdokumente: Code

Beginn: Elaboration
Dauer: 23h
Ende: Construction

Arbeitspaket: Kundenverwaltung (AP 5.5)

Beschreibung:
In diesem Arbeitspaket sollen Klassen zur Speicherung von Kunden und deren Daten erstellt werden. Auch diese müssen in persistenten Datenformaten gespeichert werden. Es soll die Möglichkeit geben nach Kunden zu suchen und ihre Daten abzurufen.

Ziele:
Die Klasse Kunde soll später für andere Packages zur Verfügung stehen, da sich wichtige Informationen zum Kunden enthält und z.B. Abrechnungen und Termine mit dem Kunden verknüpft werden sollen.

Risiken:
siehe Speicherung der Mitarbeiter. Durch die unterschiedlichen Ansichten von Mitarbeiter bzw. Chef auf die Kunden kann es zu Komplikationen kommen. Hier muss vor allem die Klasse Account in Zusammenhang mit der Kundenverwaltung genau getestet und untersucht werden.

Eingangsdokumente: Domainmodell, SSD
Ausgangsdokumente: Code

Beginn: Elaboration
Dauer: 24h
Ende: Construction

Arbeitspaket: Kassenfunktion (AP 5.6)

Beschreibung:
Dieses Arbeitspaket dreht dich um die Erstellung der Abrechnungs- und Kassenfunktionsklassen. Das System soll anhand von spezifischen, zum Kunden gehörigen Daten den Preis für den aktuellen Pressvorgang berechnen. Zusätzlich soll eine Schnittstelle entwickelt werden, um den Kunden einen Beleg ihres Einkaufes drucken zu können.

Ziele:

Die Kassenfunktion vereint alle Aspekte der Abrechnung der Kunden und ist variable bezüglich Abfüllmenge, Abfüllart, Maschine etc. Durch diese kompakte Zusammenfassung des Abrechnungsvorgangs kann der Arbeitsablauf erleichtert werden.

Risiken:
Ein mögliches Risiko birgt die Schnittstelle vom System zum Drucker, der die Abrechnungs-daten als Übersicht zur Verfügung stellt. Alternativ könnte man auf eine Textdatei ausweichen, die der Kassier dann händisch ausdruckt.

Eingangsdokumente: Domainmodell, SSD
Ausgangsdokumente: Code, Schnittstellen zum Drucker

Beginn: Elaboration
Dauer: 28h
Ende: Construction

Arbeitspaket: Lagerbestände (AP 5.7)

Beschreibung:
In diesem Arbeitspaket steht die Umsetzung der Lagerverwaltung im Fokus. Es sollen Klassen und Attribute zur Überwachung des Bestands implementiert werden, die selbstständig den Chef informieren, wenn ein Produkt ausläuft. Zusätzlich soll die Möglichkeit einer Lagerübersicht erstellt werden.

Ziele:
Mit dem Package des Lagerbestands wird eine weitere Gruppe von Klassen zur Verfügung gestellt, die der Chef für verschiedene Aktionen benutzen kann (Lagerstand einsehen, Bestellungen tätigen etc.) und die eigenständig den Ein- und Ausgang von Produkten überprüft.

Risiken:
Ein mögliches Risiko wäre der große Umfang, den die Lagerhaltung und dessen Bestand erfordern. Schon in der Anforderungsanalyse und im Domainmodell muss genau darauf geachtet werden, was für diese Package Lagerbestand nötig ist und was nur als „synthetic sugar“ zählt.

Eingangsdokumente: Domainmodell, SSD
Ausgangsdokumente: Code

Beginn: Elaboration
Dauer: 13h
Ende: Construction

Arbeitspaket: Account (AP 5.8)

Beschreibung:

Für die verschiedenen Rechte im Programm selbst sollen parallel zueinander zwei verschiedene Account-Möglichkeiten entwickelt werden. Hierbei soll aber der Mitarbeiteraccount im Chefaccount enthalten sein. Dafür muss ein Überprüfungsmodus entwickelt werden, der den Login-Vorgang bewertet und entsprechende Masken anzeigt. Des Weiteren müssen die Accountklassen auf die Mitarbeiter und Kunden zugreifen können.

Ziel:
Mit den verschiedenen Accounts wird eine Sicherung der Daten erreicht (z.B. Terminplan nur vom Chef editierbar). Ein weiteres Ziel ist die Nutzung der Software durch mehrere Mitarbeiter zur gleichen Zeit, was durch verschiedene Account und Anmeldungen ermöglicht wird.

Risiken:
Ein mögliches Risiko liegt in der Entwicklung der beiden Accounts selbst. Sie sollen voneinander unabhängig sein, gehören aber doch zusammen (Chefaccount ist nur eine Erweiterung des Mitarbeiteraccounts) Das Programm muss auch mit mehreren angemeldeten Nutzern gleichzeitig umgehen können, was nur durch geringe Kopplung erreichbar ist.

Eingangsdokumente: Domainmodell, SSD
Ausgangsdokumente: Code

Beginn: Elaboration
Dauer: 23h
Ende: Construction

Arbeitspaket: Terminplanung (AP 5.9)

Beschreibung:
Für die Terminplanung müssen Klassen zur Darstellung von Kalender- und Uhrzeitdaten implementiert werden. Des Weiteren benötigt man Funktionen, die Termine besetzen und diese in der Ansicht kennzeichnen. Zur Terminerstellung benötigt dieses Package Zugriff auf die Kundendaten, was durch eine geeignete Schnittstelle umgesetzt werden kann.

Ziel:
Mit diesem Arbeitspaket soll für die Anwendung eine übersichtliche Möglichkeit der Terminvergabe erstellt werden. Diese soll nur für den Chef editierbar sein, jedoch für alle Mitarbeiter einsehbar. Analog soll auch der Mitarbeiterschichtplan erstellt werden.

Risiken:
Ein risikoreiches Feature ist das Aktualisieren des Terminplans auf allen anderen Computer des Betriebes. Wichtig ist die Abstimmung, wie die Verbindung umgesetzt werden soll, ob mit Kabeln (weniger übersichtlich, muss zusätzlich zum Programm eingerichtet werden) oder über Internet (was passiert bei langsamen Internet/Internetausfall?).

Eingangsdokumente: Domainmodell, SSD, Architektur Plan
Ausgangsdokumente: Code, evtl. Benutzeranleitung

Beginn: Elaboration
Dauer: 25h
Ende: Construction

Arbeitspaket: Administratorverwaltung (AP 5.10)

Beschreibung:
Das Arbeitspaket Administratorverwaltung beschäftigt sich mit der Erstellung des Chefaccounts und seiner erlaubten Zugriffe. Dieses Package beinhaltet die Verwaltung von Mitarbeitern (Schichtplan), der Kunden (Terminvergabe) und der Lagerbestände. Es muss sichergestellt werden, dass nur berechtigte Nutzer auf den Administratoraccount zugreifen können.

Ziel:
Der Account des Administrators soll eine Erweiterung der Mitarbeiteraccounts sein und mindestens dieselben Funktionalitäten umfassen. Geänderte Daten (z.B. im Terminplan) sollen auch den Mitarbeitern angezeigt werden.

Risiken:
Ein Risiko besteht in der Anmeldung zum Administratoraccount. Es muss darauf geachtet werden, dass nur berechtigte Nutzer (mit gültigem Passwort) auf die Daten zugreifen können. Zu diesem Zweck müssen auch Möglichkeiten zur Accounterstellung oder Passwortänderung verfügbar sein.

Eingangsdokumente: Domainmodel, SSD, UseCase Model
Ausgangsdokumente: Code

Beginn: Elaboration
Dauer: 22h
Ende: Construction

Arbeitspaket: Test Plan (AP 6.1)

Beschreibung:
In diesem Arbeitspaket steht die Erarbeitung eines Test Management Plan im Vordergrund. Dieser enthält die nach Absprache vereinbarte Vorgehensweise beim Testen der Software. So soll vor der eigentlichen Implementierung des Codestücks bereits ein passender Test geschrieben werden, der sich an den Anforderungsspezifikationen orientiert.

Ziele:
Durch eine einheitliche Vorgehensweise im Testen von Code können Fehler vermindert werde. Des Weiteren erlauben angeglichene Verfahren das schnellere Verständnis für fremde Tests und können so auch wechselseitig durchgeführt werden.

Risiken:
Ein ungenügend ausgereifter oder fehlerhafter Test Plan, der sich durch das ganze Projekt zieht und an dem sich alle Teammitglieder orientieren, kann Testfälle übersehen und stellt somit ein Risiko für die Software dar.

Eingangsdokumente: Use Cases, Requirement Specification
Ausgangsdokumente: Test Plan

Beginn: Elaboration
Dauer: 4h
Ende: Elaboration

Arbeitspaket: Test Case (AP 6.2)

Beschreibung:
Jeder spezielle Test Case orientiert sich genau an den vorgegebenen Anforderungen, die der spezielle UseCase bzw. das Codefragment aufweisen sollen. Es wird jeweils mitdokumentiert, welche Teste erfolgreich bzw. nicht erfolgreich waren und welche weiteren Schritte notwendig sind.

Ziele:
Durch eine spezielle Anpassung des Tests auf ein Codestück soll die Fehlerfindung und -behebung vereinfacht werden. Die Testprotokolle dienen als Grundlage für eventuelle Anforderungsänderungen und als Hilfestellungen für die anderen Teammitglieder (breitere Fehlerabdeckung)

Risiken:
Um eine genaue Analyse zu bekommen, braucht man viele Test Cases. Dies kann schnell unübersichtlich werden, weshalb eine gute Strukturierung nötig ist.

Eingangsdokumente: spezieller Use Case, Test Plan, Code
Ausgangsdokumente: gesäuberter Code, Test Dokumentation

Beginn: Elaboration
Dauer: 75h
Ende: Construction

Arbeitspaket: Virtuelle Maschine (AP 6.3)

Beschreibung:
Ein Medium, auf dem wir unser Projekt Mosti testen und unsere Codestücke vereinen. Auf diese VM kann jeder zugreifen und gemeinsame Arbeitsergebnisse können gesehen und besprochen werden.

Ziel:
Die Software kann auf verschiedenen Betriebssystemen getestet werden, ohne das eigene Betriebssystem zu schädigen. Es können auch Codestücke, die auf verschiedenen Betriebssystemen laufen, ohne Komplikationen miteinander verbunden werden.

Risiken:
Virtuelle Maschinen sind weniger effizient als reale Maschinen, gleichzeitig betriebene VM beeinflussen sich gegenseitig, Schutzmechanismen gegen Viren und Maleware wird nicht vom eigenen Betriebssystem übernommen.

Eingangsdokumente: Codestücke
Ausgangsdokumente: Prototypen, kompilierbarer Code

Beginn: Construction
Dauer: 14h
Ende: Transition

Arbeitspaket: Werkzeuge - GIT (AP 7.1)

Beschreibung:
GIT ist eine freie Quellcode-Management-Software. Durch die Bereitstellung eines Repositorys, zu dem jedes Teammitglied Zugang hat, können hier gemeinsame Daten abgelegt, eingesehen und editiert werden. Alle von den Mitgliedern erstellten Artefakte stehen hier dem kompletten Team zur Verfügung.

Ziele:
Durch die Versionsverwaltung kann auch auf überschriebene Dateien und frühere Versionen zugegriffen werden ohne regelmäßige Backups erstellen zu müssen. Des Weiteren kann man sehen, welches Mitglied welche Dateien bearbeitet hat und so den Arbeitsprozess nachverfolgen.

Risiken:
Durch falsche oder Nichtnutzung der GIT-Software kann es zu Unstimmigkeiten in den Dateien kommen und zu Verständnisproblemen, da die Mitglieder nicht auf dem neusten Stand sind.

Eingangsdokumente: sämtliche Artefakte
Ausgangsdokumente: Artefakte und ihre früheren Versionen

Beginn: Mit Beginn des Projekts
Dauer: 12h
Ende: Mit Ende des Projekts

Arbeitspaket: Werkzeug - Maven (AP 7.2)

Beschreibung:
Maven ist ein Build-Management-Tool, welches auf Java basiert. Hier kann man Java-Programme erfassen und verwalten. Mit diesem werden die einzelnen Quellcodes der Mosti-Software zusammengefügt unabhängig von den vorhandenen oder nichtvorhandenen Importen in den Einzeldateien.

Ziele:
Zu einem sollen möglichst viele Schritte (über Kompilieren bis hin zum Testen) automatisiert werden können. Es soll auch ermöglichen, dass schneller analysiert und gearbeitet wird. Zuletzt soll die Entwicklungsumgebung noch unterstützt werden.
Bei Mosti soll schneller Kompiliert werden können und Verfahrensschritte automatisiert werden können.

Risiken:

Die geringe Erfahrung der Teammitglieder bezüglich der Nutzung der Maven-Software könnte zu Problemen führen.

Eingangsdokumente: Quellcode
Ausgangsdokumente: Fehlermeldungen

Beginn: Mit Beginn der ersten Implementierung
Dauer: 31h
Ende: Mit Ende der letzten Implementierung

Arbeitspaket: Überführung in Anwendung (AP 8.1)

Beschreibung:
Zuerst wird unsere Anwendung mit Hilfe der Virtuellen Maschine und ohne die Entwicklungsumgebung auf verschiedenen Betriebssystemen getestet.

Ziele:
Mosti soll auf vielen verschiedenen Betriebssystemen lauffähig sein. Ebenso soll es möglich sein, dass mehrere Accounts gleichzeitig auf die Anwendung Zugriff haben können und diese auch bearbeiten können. Es soll also ein Anwendungsfall funktionieren. Außerdem soll die Anwendung mit Hilfe einer Website heruntergeladen werden können und so zur Verfügung stehen.

Risiken:
Zu einem kann es passieren, dass die Anwendung nicht auf allen Betriebssystemen lauffähig ist. Zum anderen kann aus sein, dass der Anwendungsfall nicht funktioniert und somit auch nicht heruntergeladen werden kann.

Eingangsdokumente: Quellcode
Ausgangsdokumente: lauffähige Anwendung bzw. Software

Beginn: Construction
Dauer: 30h
Ende: Bei Projektende