Universität Bremen
FB 3 – Informatik
Prof. Dr. Rainer Koschke
TutorIn: Marcel Steinbeck

Software–Projekt 2 WiSe 2019/2020

VAK 03-BA-901.02

Architekturbeschreibung

Liam Hurwitz hurwitz@tzi.de
Aaron Rudkowski rudkowsk@uni-bremen.de

Abgabe: 22. Dezember 2019 — Version 1.0

Inhaltsverzeichnis

Version und Änderungsgeschichte

Die aktuelle Versionsnummer des Dokumentes sollte eindeutig und gut zu identifizieren sein, hier und optimalerweise auf dem Titelblatt.

Version	Datum	Änderungen
0.1	TT.MM.JJJJ	Dokumentvorlage als initiale Fassung kopiert
0.2	04.12.2019	Faktortabelle
...		

1 Einführung

1.1 Zweck

Diese Architektur Beschreibung hat als Zweck, die Ermittlung der Eingeschärften und Struktur des Software von Produktion und Logistik. Welche eine Verwaltung der Kette von Prozessen bearbeiten soll. damit die Benutzer oder Kunden als Basis für die Testen nutzen kann.

Diese Dokument beschreibt die Hauptpunkten der Parameter und Begrenzungen des Software, sowie die Parameter und die Zeit ablauft der Prozess, um diese Software Entwickeln und testest wurde.

1.2 Status

1.3 Definitionen, Akronyme und Abkürzungen

1.4 Referenzen

1.5 Übersicht über das Dokument

2 Globale Analyse

Hier werden Einflussfaktoren aufgezählt und bewertet sowie Strategien zum Umgang mit interferierenden Einflussfaktoren entwickelt.

2.1 Einflussfaktoren

Hier sind Einflussfaktoren gefragt, die sich auf die Architektur beziehen. Es sind ausschließlich architekturrelevante Einflussfaktoren, und nicht z. B. solche, die lediglich einen Einfluss auf das Projektmanagement haben. Fragt Euch also bei jedem Faktor:

Beeinflusst er wirklich die Architektur? Macht einen einfachen Test: Wie würde die Architektur aussehen, wenn ihr den Einflussfaktor E berücksichtigt? Wie würde sie aussehen, wenn Ihr E nicht berücksichtigt? Kommt in beiden Fällen dieselbe Architektur heraus, dann kann der Einflussfaktor nicht architekturrelevant sein.

Es geht hier um Einflussfaktoren, die

1. *sich über die Zeit ändern,*
2. *viele Komponenten betreffen,*
3. *schwer zu erfüllen sind oder*
4. *mit denen man wenig Erfahrung hat.*

Die Flexibilität und Veränderlichkeit müssen ebenfalls charakterisiert werden.

1. *Flexibilität: Könnt Ihr den Faktor zum jetzigen Zeitpunkt beeinflussen?*
2. *Veränderlichkeit: ändert der Faktor sich später durch äußere Einflüsse?*

Unter Auswirkungen sollte dann beschrieben werden, wie *der Faktor* was *beeinflusst. Das können sein:*

- *andere Faktoren*
- *Komponenten*
- *Operationsmodi*
- *Designentscheidungen (Strategien)*

Verwendet eine eindeutige Nummerierung der Faktoren, um sie auf den Problemkarten einfach referenzieren zu können.

2.2 Probleme und Strategien

Aus einer Menge von Faktoren ergeben sich Probleme, die nun in Form von Problemkarten beschrieben werden. Diese resultieren z. B. aus

- *Grenzen oder Einschränkungen durch Faktoren*
- *der Notwendigkeit, die Auswirkung eines Faktors zu begrenzen*
- *der Schwierigkeit, einen Produktfaktor zu erfüllen, oder*
- *der Notwendigkeit einer allgemeinen Lösung zu globalen Anforderungen.*

Dazu entwickelt Ihr Strategien, um mit den identifizierten Problemen umzugehen.

Achtet auch hier darauf, dass die Probleme und Strategien wirklich die Architektur betreffen und nicht etwa das Projektmanagement. Die Strategien stellen im Prinzip die Designentscheidungen dar. Sie sollten also die Erklärung für den konkreten Aufbau der verschiedenen Sichten liefern.

Beschreibt möglichst mehrere Alternativen und gebt an, für welche Ihr Euch letztlich aus

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
O1 : Organisation				
O1.1 Time-To-Market				
	Die Auslieferung erfolgt am 08.03.2020.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Nicht alle Funktionen können realisiert werden.
O1.2 Architektur-Abgabe				
	Die Auslieferung erfolgt am 22.12.2020.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Durch den Zeitdruck könnte die Architektur mangelhaft werden. Wenn wir uns nicht genug Zeit lassen, könnten Aspekte, die relevant für die Architektur sind, vergessen werden.
O1.3 Entwickler				
	Die Projektgruppe besteht aus 6 Entwicklern	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Mangel an der Architektur
O1.4 Fähigkeiten Entwickler				
	Nicht alle Entwickler haben die gleiche Programmier-erfahrung und auch nicht mit den gleichen Technologien.	Hohe Veränderlichkeit durch Ausführen des Projekts und Recherche.	–/ –	Die Implementierung kann Mangel enthalten.

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
T1: Technik				
T1.1: Programmiersprache				
	Java 11 oder höher ist vorgegeben.	Ein wenig Flexibilität an der Version der Sprache aber nicht an der Sprache.	–/ –	Das Projekt muss in Java umgesetzt werden.
T1.2 Webbrowser				
	Die Anwendung muss in gängigen Browsern (Firefox, Internet Explorer, Safari, Edge) laufen.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Bei der Implementierung muss darauf geachtet werden, plattformunabhängig vorzugehen.
T1.3 Server				
	Zur Implementierung muss JavaEE 8 benutzt werden.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Das Projekt muss komplett in Java umgesetzt werden.
T1.4: Oberfläche				
	Als Framework zur Erstellung der Oberfläche muss JSF verwendet werden.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Das Projekt muss in Java umgesetzt werden.
T1.5: Persistent				
	Relationale Datenbank H2 muss für die Persistenz verwendet werden.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Bei der Implementierung muss H2 verwendet werden.
T1.6: Build System				
	Maven muss als Build-System verwendet werden.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Das Projekt muss maven-build fähig sein.

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
T1.7: Multiple Users				
	Die Anwendung muss von mehreren Benutzern gleichzeitig verwendbar sein.	Keine Flexibilität, da Vorgaben bestehen. / Keine Veränderbarkeit, da Vorgaben bestehen.	–/ –	Die Anwendung darf nicht von gleichzeitiger Verwendung von mehreren Nutzern überfordert sein; ebenfalls dürfen dadurch keine Sicherheitslücken entstehen.
P1: Produktfaktoren				
P1.1: Produktfunktionen				
P1.1.1: Upload				
	Prozessketten- und Prozessschritte Parameter sollen aus JSON-Dateien hochgeladen werden können.	Keine Veränderlichkeit, da es vom Chinese Menu ist aber Flexibilität gegeben: muss nicht unbedingt gemacht werden.	–/ –	Die Architektur muss vorsehen, dass JSON-Dateien hochgeladen werden können, aus denen automatisch für einen Prozessschritt/ eine Prozesskette die Parameter eingefügt werden.
P1.1.2: Download				
	Die Parameter von einzelnen Prozessschritten sollen nach JSON exportiert und heruntergeladen werden können.	Keine Veränderlichkeit, da es vom Chinese Menu ist aber Flexibilität gegeben: muss nicht unbedingt gemacht werden.	–/ –	Die Architektur muss vorsehen, dass für einen Prozessschritt die Parameter gelistet und in einem JSON-Format exportiert werden können.
P1.2: Protokollierung				
	Die Übergänge in den Prozessketten müssen protokolliert werden.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss vorsehen, dass für jede Prozesskette ein Protokoll gespeichert wird, das automatisch nach jedem Prozessschritt mit Nachbedingungen u.ä. (Was??) ergänzt wird.

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
P1.2.1: Export Protokollierung				
	Die Protokolle müssen nach JSON oder XML exportierbar sein.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss vorsehen, dass es 1. eine Möglichkeit für den Benutzer gibt, sich diese Protokolle das Format seiner Wahl exportieren zu lassen, und 2. die Protokolle nach JSON oder XML exportierbar sind.
P1.3 Benutzerverwaltung				
	Nutzer sollen erstellt, bearbeitet und gelöscht werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Mangel an den Mindestanforderungen, Architektur wird möglicherweise vom Kunden nicht akzeptiert.
P1.4 Experimentierstationen verwalten				
	Stationen sollen erstellt, bearbeitet und gelöscht werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Mangel an den Mindestanforderungen, Architektur wird möglicherweise vom Kunden nicht akzeptiert.
P1.4.1 Auslastung				
	Eine Übersicht über die Auslastung der Experimentier-stationen soll möglich sein.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss speichern, welche Experimentierstationen belegt sind.
P1.4.2: Kaputte Stationen				
	Experimentier-stationen sollen als kaputt gemeldet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Anwendung muss mit kaputten Stationen umgehen können: Diese sollte nicht mehr eingeplant werden. Ebenfalls sollten Experimentierstationen für Benutzer als kaputt anzeigbar sein.

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
P1.5: Prozessschritte				
	Prozessschritte sollen erstellt, gelöscht, bearbeitet(sofern noch nicht gestartet), und hervorgehoben (sofern im Zustand kaputt) werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Mangel an den Mindestanforderungen, Architektur wird möglicherweise vom Kunden nicht akzeptiert.
P1.6: Prozessketten				
	Prozessketten sollen erstellt, gelöscht, und bearbeitet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Mangel an den Mindestanforderungen, Architektur wird möglicherweise vom Kunden nicht akzeptiert.
P1.7: Aufträge				
	Aufträge sollen erstellt, freigegeben, gestoppt, gelöscht, priorisiert, und bearbeitet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Mangel an den Mindestanforderungen, Architektur wird möglicherweise vom Kunden nicht akzeptiert.
P1.7.1: Zuordnen zu Aufträgen				
	Aufträge sollen Proben/Trägern zugeordnet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Mangel an den Mindestanforderungen, Architektur wird möglicherweise vom Kunden nicht akzeptiert.
P1.7.2: Automatisches Zuordnen				
	Alternativ sollen Aufträgen automatisch Proben/Trägern zugeordnet werden.	Keine Veränderlichkeit, da vom Chinese Menu, ist diese Anforderung optional.	–/ –	Die Anwendung muss die Parameter der Prozessschritte einer Prozesskette auswerten können, und daraus schließen, welche Proben/Träger gebraucht werden.

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
P1.7.3: Prioritäten				
	Für Aufträge soll eine Priorität errechnet werden.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Anwendung muss die Parameter der Prozessschritte einer Prozesskette auswerten können, und daraus schließen, welche Proben/Träger gebraucht werden.
P1.7.4: Zustand Auftrag				
	Der Zustand eines Prozess-schrittes (Auftrag im Kontext Technologe) soll manuell aktualisiert werden.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss vorsehen, dass von außen eine Auwahl des Zustandes möglich ist.
P1.8: Träger				
	Träger sollen erstellt und gelöscht werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss eine Möglichkeit haben, Träger zu löschen (über einen Knopf oä) und neue Träger zu erstellen.
P1.9: Dynamische Zustandsautomaten				
	Zustandsautomaten (linear, keine Verzweigungen) für die Prozessschritte sollen angelegt, gelöscht und bearbeitet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Architektur entspricht nicht den Anforderungen.
P1.10: Kaputte Proben				
	Proben sollen als kaputt gemeldet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss vorsehen, dass eine Probe unterschiedliche Zustände hat (kaputt/nicht kaputt) und dass der Übergang vom Nutzer hervorgerufen wird.

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
P1.10.1: Verlorene Proben				
	Proben sollen als verloren gemeldet werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss mit verlorenen Proben umgehen können: Es muss möglich sein, zu sehen, wenn eine Probe verloren gegangen ist, diese darf nicht weiter verplant werden.
P1.10.2; Lagerübersicht				
	Die im Lager liegenden Proben sollen angezeigt werden können.	Keine Veränderlichkeit, da es vom Chinese Menu ist aber Flexibilität gegeben: muss nicht unbedingt gemacht werden.	–/ –	Es muss irgendwie erkennbar sein, welche Proben im Lager sind, und welche nicht. Weitergehend muss es einfach sein,alle im Lager liegenden zu finden und aufzulisten.
P1.10.4:Proben Kommentar				
	Zu Proben sollen Kommentare erstellt werden können.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Es muss eine Fläche für Kommentareingabe und Darstellung geben.
P2.1:Mehrsprachig				
	Der User soll zwischen Deutsch und Englisch entscheiden können.	Keine Veränderlichkeit, da es vom Chinese Menu ist aber Flexibilität gegeben: muss nicht unbedingt gemacht werden.	–/ –	Die Architektur muss ein Umschalten zwischen den beiden Sprachen in der Darstellung vorsehen.
P2.2:Anzeige				
	Je nach Rechten des Benutzers sollen Prozessschritte, -ketten, Stationen, Proben, Benutzer, Aufträge, dynamische Abläufe und/oder Transportaufträge angezeigt werden.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss eine Vielzahl an Informationen aus der Datenbank auslesen und anzeigen können, mit dem Hinblick darauf, dass es sich teilweise um sehr große Mengen handelt (Proben).

Abgeleitet aus	Einflussfaktor	Flexibilität und Veränderlichkeit	++/ –	Auswirkungen
P3.1: User Rechte				
	Es soll unterschiedliche Rollen mit unterschiedlichen Rechten und angezeigten Informationen geben. Jeder User soll mindestens eine Rolle haben.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss zwischen diesen Rollen unterscheiden können und sichergehen, dass bestimmte Funktionen und Informationen nur zur Verfügung stehen wenn der Benutzer die Rechte dafür hat.
P3.2: Authentifizierung				
	Benutzer sollten sich in das System einloggen müssen, um relevante Informationen angezeigt zu bekommen.	Keine Veränderlichkeit, da Teil der Mindestanforderungen.	–/ –	Die Architektur muss eine Login Seite vorsehen; ein Zugriff auf die Anwendung ohne sich einzuloggen soll nicht möglich sein.

Problem 1: Userrollen
Es soll unterschiedliche Rollen geben, die unterschiedliche Rechte/Fähigkeiten haben; jeder User kann mehrere Rollen übernehmen.
Einflussfaktoren: T 1.5: Persistenz T 1.7: Multiple Users O 1.1: Time-To-Market
Strategien
Strategie 1: Machen sichtbar unterschiedlichen Schnittstellen des Software in Bezug von der Rollen jedes User.

Tabelle 1: Caption

Problem 2: Datensicherheit
Jeder User darf nur die Informationen sehen, die ihm nach seiner Rolle/seinen Rollen zustehen. Unauthorisierten Personen sollen keine Informationen angezeigt werden. Auch die Bearbeitung von Informationen ist nur Rollen mit den entsprechenden Privilegien erlaubt.
Einflussfaktoren: T 1.5: Persistenz T 1.7: Multiple Users
Strategien
Strategie 1: Das System wird mit einer Authentifizierung versehen: Jeder muss eingeloggt sein, um die Anwendung nutzen zu können. In der Implementierung muss dafür gesorgt werden, dass es keine Benutzer ohne mindestens eine Rolle gibt.

Tabelle 2: Caption

welchem Grunde entschieden habt. Natürlich müssen die genannten Strategien in den folgenden Sichten auch tatsächlich umgesetzt werden!

Ein sehr häufiger Fehler ist es, dass SWP-Gruppen arbeitsteilig vorgehen: die eine Teilgruppe schreibt das Kapitel zur Analyse von Faktoren und zu den Strategien, die andere Teilruppe beschreibt die diversen Sichten, ohne dass diese beiden Teilgruppen sich abstimmen. Da der Zweck der Faktoren und Strategien ist, die Designentscheidungen für den Entwurf zu erarbeiten, besteht natürlich ein Zusammenhang zwischen den Faktoren, Strategien und Sichten. Dieser muss erkennbar sein, indem sich die verschiedenen Kapitel eindeutig aufeinander beziehen.

Problem 3: Nebenläufige Benutzung
Die Anwendung kann von mehreren Usern gleichzeitig benutzt werden. Dabei müssen die Daten in der Datenbank richtig angezeigt/verändert werden, und die Datensicherheit darf nicht komprimiert werden.
Einflussfaktoren: T 1.5: Persistent T 1.7: Multiple Users XXX : Datenbank
Strategien
Strategie 1: Wir verwenden eine transaktionssichere Datenbank.

Tabelle 3: Caption

Problem 4: Entlassen/Einstellen Mitarbeiter
Da neue Mitarbeiter hinzukommen/alte gehen können, muss es notwendig sein, neue User hinzuzufügen und alte zu Löschen, ohne dass relevante Informationen zu den Prozessketten verloren gehen (wie Nullpointer in den Protokollen).
Einflussfaktoren: T 1.5: Persistent T 1.7: Multiple Users XXX : Datenbank
Strategien
Strategie 1: Der Admin der Software kann neue Mitglieder hinzufügen und auch existierende Mitglieder entfernen.

Tabelle 4: Caption

Problem 5: Upload/Download Prozessketten- und Prozessschrittparameter
Die Software kann neue Prozessketten und Prozessschrittparameter aus JSON-Dateien hochladen, und bestehende als JSON-Dateien exportieren.
Einflussfaktoren: T 1.5: Persistent P 1.1.1: Upload P 1.1.2: Download XXX : Datenbank
Strategien
Strategie 1: XXX

Tabelle 5: Caption

Problem 6: Protokollierung der Prozessketten
Die Abarbeitung einer Prozesskette soll protokolliert werden (wann welche Zustandsübergänge stattfanden).
Einflussfaktoren: T 1.5: Persistent P 1.2: Protokollierung P 1.4: p 1.4.1
Strategien
Strategie 1: XXX

Tabelle 6: Caption

Problem 7: Stationenbearbeitung
Der Administrator soll neue Stationen hinzufügen können. Die Software muss die Fähigkeit haben neue Stationen zu übernehmen und zu nutzen. Außerdem soll der Zustand jeder Station sichtbar und bearbeitbar sein.
Einflussfaktoren: T 1.5: Persistent P 1.4: Experimentierstationen (XXX) P 1.4: Kaputte Stationen
Strategien
Strategie 1: Eingeschaften der Stationen herausfinden, damit der entsprechende Nutzer eine Station einfach hinzufügen kann.

Tabelle 7: Caption

Problem 8: Überwachung der Zustände
Wenn ein Logistiker einen Prozess gestartet hat, sollte der Prozess gestoppt, wieder gestartet und priorisiert werden können von dem Prozessketteadministrator.
Einflussfaktoren: T 1.5: Persistent P 1.7: Aufträge (XXX) P 1.7.1: Kaputte Stationen P 1.7.2: Automatisches Zuorden P 1.9: Automatisches Zuorden
Strategien
Strategie 1: Zustandautma

Tabelle 8: Caption

Problem 9: Prozessbearbeitung und Überwachung
Der Prozesskettenadministrator kann einen Prozess mit unterschiedlichen Prozessschritten bauen. Dieser Prozess soll veränderbar sein und umtauschbar, um den Bedürfnissen der Benutzer gerecht zu werden.
Einflussfaktoren: T 1.5: Persistent P 1.4: Experimentierstationen (XXX) P 1.6: Prozessketten
Strategien
Strategie 1: Eigenschaften der Stationen herausfinden, damit der entsprechende Nutzer ein Station einfach hinzufügen können.

Tabelle 9: Caption

Problem 10: Laden Informationen
Der Logistiker möchte einen Überblick über alle Proben, die im Lager liegen. Da die Probenkügelchen alle einzeln gespeichert werden, und es somit eine sehr große Menge an Proben ist, könnte das Laden aller Proben auf einmal zu Performanceproblemen führen.
Einflussfaktoren: P1.10.2: Lagerübersicht
Strategien
Strategie 1: Es wird immer nur ein kleiner Ausschnitt aus der Datenbank geladen.

Tabelle 10: Caption

Problem 11: Verlust/Schaden an Proben
Proben können im Laufe der Bearbeitung einer Prozesskette kaputt oder verloren gehen. Diese können dann nicht weiter benutzt werden, was zu Problemen im weiteren Verlauf führen könnte.
Einflussfaktoren: P1.10: Kaputte Proben P1.10.1: Verlorene Proben
Strategien
Strategie 1: Proben können als kaputt gemeldet werden. Dadurch werden sie sowohl als solche dargestellt, und können weder manuell noch automatisch irgendwelchen Prozessen zugeordnet werden.

Tabelle 11: Caption

Problem 12: Ambitionierter Zeitplan
Die Auslieferung der Anwendung muss am 08.03.2020 erfolgen. Dadurch könnten die Entwickler unter Zeitdruck geraten und eventuell nicht alle Anforderungen erfüllen.
Einflussfaktoren: O1.1: Time-To-Market
Strategien
Strategie 1: Einteilung Anforderungen: Es gibt grundliegende Anforderungen, die umbedingt erfüllt werden müssen, und andere Forderungen, die schön wären, aber nicht notwendig sind. Aus letzteren können die Entwickler, wenn Zeit über ist, weitere zur Implementierung aussuchen.

Tabelle 12: Caption

Problem 13: Vorgaben von Technologien
Die Vorgaben, welche Technologien bei der Implementierung verwendet werden sollen/dürfen, sind relativ streng.
Einflussfaktoren: T 1.1: Programmiersprache T1.2: Webbrowser T1.3: Server T1.4: Oberfläche T1.5: Persistenz T1.6: Build System
Strategien
Strategie 1: Alle Entwickler müssen sich die Zeit nehmen, sich in diese Technologien anzueignen. Ebenfalls sollte die Gruppe sich untereinander unterstützen; hilfreich wäre zum Beispiel eine Sammlung an Problemen, auf die Entwickler gestoßen sind, und wie sie diese gelöst haben, um eventuell anderen Gruppenmitgliedern die Recherche zu erleichtern.

Tabelle 13: Caption

Problem 14: Probenzuordnung zu Aufträgen
Proben sollen ohne Einfluss des Benutzers Aufträgen passend zugeordnet werden. Das bedeutet, dass das System die Eigenschaften von Proben kennen und den Anforderungen von Aufträgen passend zuordnen soll.
Einflussfaktoren: P1.7.2: Automatisches Zuordnen
Strategien
Strategie 1: Diese Anforderung ist optional, das heißt zur Not kann man ihn einfach weglassen. **Strategie 2:** Die Eigenschaften/Anforderungen von Proben/Aufträgen sollten nicht einfach nur String-Felder sein, da diese miteinander vergleichen schwierig ist (unterschiedliche Formulierungen etc). Stattdessen sollten Eigenschaften/Anforderungen vordefiniert sein (vom Prozesskettenadministrator?) und gespeichert werden, welche Eigenschaften zu welchen Anforderungen passen.

Tabelle 14: Caption

Problem 15: Kompetenz der Entwickler
Die unterschiedlichen Entwickler haben unterschiedlich (weitreichende) Kenntnisse der Technologien; das führt dazu, dass die Implementierung eventuell länger dauert/fehlerbehaftet ist. Aneignen dieser Informationen ist vor allem wegen dem Ambitionierten Zeitplan stressig.
Einflussfaktoren: O1.1: Time-To-Market O1.3: Entwickler O1.4: Fähigkeiten Entwickler T1.1: Programmiersprache T1.2: Webbrowser T1.3: Server T1.4: Oberfläche T1.5: Persistenz T1.6: Build System
Strategien
Strategie 1: Die Implementierung wird bestmöglich nach Können der Entwickler aufgeteilt; die Entwickler bemühen sich frühzeitig darum, sich weiteres Wissen anzueignen. Ebenfalls unterstützen sich alle Gruppenmitglieder gegenseitig.

Tabelle 15: Caption

3 Konzeptionelle Sicht

Diese Sicht beschreibt das System auf einer hohen Abstraktionsebene, d. h. mit sehr starkem Bezug zur Anwendungsdomäne und den geforderten Produktfunktionen und -attributen. Sie legt die Grobstruktur fest, ohne gleich in die Details von spezifischen Technologien abzugleiten. Sie wird in den nachfolgenden Sichten konkretisiert und verfeinert. Die konzeptionelle Sicht wird mit UML-Komponentendiagrammen visualisiert.

4 Modulsicht

Diese Sicht beschreibt den statischen Aufbau des Systems mit Hilfe von Modulen, Subsystemen, Schichten und Schnittstellen. Diese Sicht ist hierarchisch, d. h. Module werden in Teilmodule zerlegt. Die Zerlegung endet bei Modulen, die ein klar umrissenes Arbeitspaket für eine Person darstellen und in einer Kalenderwoche implementiert werden können. Die Modulbeschreibung der Blätter dieser Hierarchie muss genau genug und ausreichend sein, um das Modul implementieren zu können.

Die Modulsicht wird durch UML-Paket- und Klassendiagramme visualisiert.

Die Module werden durch ihre Schnittstellen beschrieben. Die Schnittstelle eines Moduls M ist die Menge aller Annahmen, die andere Module über M machen dürfen, bzw. jene Annahmen, die M über seine verwendeten Module macht (bzw. seine Umgebung, wozu auch Speicher, Laufzeit etc. gehören). Konkrete Implementierungen dieser Schnittstellen sind das Geheimnis des Moduls und können vom Programmierer festgelegt werden. Sie sollen hier dementsprechend nicht beschrieben werden.

Die Diagramme der Modulsicht sollten die zur Schnittstelle gehörenden Methoden enthalten. Die Beschreibung der einzelnen Methoden (im Sinne der Schnittstellenbeschreibung) geschieht allerdings per Javadoc im zugehörigen Quelltext. Das bedeutet, dass Ihr für alle Eure Module Klassen, Interfaces und Pakete erstellt und sie mit den Methoden der Schnittstellen verseht. Natürlich noch ohne Methodenrümpfe bzw. mit minimalen Rümpfen. Dieses Vorgehen vereinfacht den Schnittstellenentwurf und stellt Konsistenz sicher.

Jeder Schnittstelle liegt ein Protokoll zugrunde. Das Protokoll beschreibt die Vor- und Nachbedingungen der Schnittstellenelemente. Dazu gehören die erlaubten Reihenfolgen, in denen Methoden der Schnittstelle aufgerufen werden dürfen, sowie Annahmen über Eingabeparameter und Zusicherungen über Ausgabeparameter. Das Protokoll von Modulen wird in der Modulsicht beschrieben. Dort, wo es sinnvoll ist, sollte es mit Hilfe von Zustands- oder Sequenzdiagrammen spezifiziert werden. Diese sind dann einzusetzen, wenn der Text allein kein ausreichendes Verständnis vermittelt (insbesondere bei komplexen oder nicht offensichtlichen Zusammenhängen).

Der Bezug zur konzeptionellen Sicht muss klar ersichtlich sein. Im Zweifel sollte er explizit erklärt werden. Auch für diese Sicht muss die Entstehung anhand der Strategien erläutert werden.

5 Datensicht

Hier wird das der Anwendung zugrundeliegende Datenmodell beschrieben. Hierzu werden neben einem erläuternden Text auch ein oder mehrere UML-Klassendiagramme verwendet. Das hier beschriebene Datenmodell wird u. a. jenes der Anforderungsspezifikation enthalten, allerdings mit implementierungsspezifischen Änderungen und Erweiterungen. Siehe die gesonderten Hinweise.

6 Ausführungssicht

Die Ausführungssicht beschreibt das Laufzeitverhalten. Hier werden die Laufzeitelemente aufgeführt und beschrieben, welche Module sie zur Ausführung bringen. Ein Modul kann von mehreren Laufzeitelementen zur Laufzeit verwendet werden. Die Ausführungssicht beschreibt darüber hinaus, welche Laufzeitelemente spezifisch miteinander kommunizieren. Zudem wird bei verteilten Systemen (z. B. Client-Server-Systeme) dargestellt, welche Module von welchen Prozessen auf welchen Rechnern ausgeführt werden.

7 Zusammenhänge zwischen Anwendungsfällen und Architektur

In diesem Abschnitt sollen Sequenzdiagramme mit Beschreibung(!) für zwei bis drei von Euch ausgewählte Anwendungsfälle erstellt werden. Ein Sequenzdiagramm beschreibt den Nachrichtenverkehr zwischen allen Modulen, die an der Realisierung des Anwendungsfalles beteiligt sind. Wählt die Anwendungsfälle so, dass nach Möglichkeit alle Module Eures entworfenen Systems in mindestens einem Sequenzdiagramm vorkommen. Falls Euch das nicht gelingt, versucht möglichst viele und die wichtigsten Module abzudecken.

8 Evolution

Beschreibt in diesem Abschnitt, welche Änderungen Ihr vornehmen müsst, wenn sich Anforderungen oder Rahmenbedingungen ändern. Insbesondere würden hierbei die in der Anforderungsspezifikation unter „Ausblick“ genannten Punkte behandelt werden.

. . .