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
xxxx xxxxxxxx xxxx@tzi.de

Abgabe: TT. Monat JJJJ — 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	TT.MM.JJJJ	...
...		

1 Einführung

1.1 Zweck

Was ist der Zweck dieser Architekturbeschreibung? Wer sind die LeserInnen?

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.

[]@llll@ Faktor Flexibilität und Veränderlichkeit Auswirkung

[]@l@ O 1:Organisation

[]@l@ O 1.1 Time-To-Market

[]@llll@ Die Auslieferung erfolgt am 08.03.2020. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Nicht alle Funktionen können realisiert werden

[]@l@ O 1.2 Architektur-Abgabe

[]@llll@ Die Auslieferung erfolgt am 22.12.2019. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Mangel an der Architektur

[]@l@ O 1.3 Entwickler

[]@llll@ Die Projektgruppe besteht aus 6 Entwicklern Hohe Flexibilität an Arbeitsschritten, diese Flexibilität besteht aus der Verfügbarkeit der Entwicklern Implementierung mit Mangeln

[]@l@ O 1.4 Fähigkeiten Entwickler

[]@llll@ Nicht alle Entwickler haben die gleiche Programmiererfahrung und auch nicht mit den Gleichen technologien höhe Veränderlichkeit durch die selbstMotivation und durch die Ausführung des Projektes. Implementierung mit Mangeln, Effizient an der Implementierung.

[]@l@ O 1.5 Teamarbeit

[]@llll@ Kommunikationsfähigkeiten und Erfahrung an der Team arbeits die Termin können nicht Stress, Zeit kosten, wieder Implementation von Features.

[]@l@ T1: Software

[]@l@ T1.1: Programmiersprache

[]@llll@ 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.

[]@l@ T1.2: Webbrowser

[]@llll@ Die Anwendung muss in gängigen Browsern (Firefox, Internet Explorer, Safari, Edge) laufen Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Bei der Implementierung muss darauf geachtet werden, plattformunabhängig vorzugehen.

[]@l@ T1.3: Server

[]@llll@ Zur Implementierung muss JavaEE 8 benutzt werden. Keine Flexibilitat, es ist eine Hauptanforderung des Kunden Das Projekt muss komplett in Java umgesetzt werden.

[]@l@ T1.4: Oberfläche

[]@llll@ Als Framework zur Erstellung der Oberfläche muss JSF verwendet werden. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen.

[]@l@ T1.5: Persistenz

[]@llll@ Relationale Datenbank H2 muss für die Persistenz verwendet werden. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen.

[]@l@ T1.6: Build System

[]@llll@ Maven muss als Build-System verwendet werden. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Anwendung muss maven-build fähig sein.

[]@l@ T1.7: Multiple Users

[]@llll@ Die Anwendung muss von mehreren Benutzern gleichzeitig verwendbar sein. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Anwendung darf nicht von gleichzeitiger Verwendung von mehreren Nutzern überfordert sein; ebenfalls dürfen dadurch keine Sicherheitslücken entstehen.

[]@l@ P: Produkt Faktoren

[]@l@ P1: Produkt Funktionen

[]@l@ P1.1.1: Upload

[]@llll@ 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.

[]@l@ P1.1.2: Download

[]@llll@ Die Parameter von einzelnen Prozessschritten sollen nach JSON exportieren

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.

[]@l@ P1.2: Protokollierung

[]@llll@ Die Übergänge in den Prozessketten müssen protokolliert werden. Keine Veränderlichkeit/Flexibilität, 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.

[]@l@ P1.2.1: Export Protokollierung

[]@llll@ Die Protokolle müssen nach JSON oder XML exportierbar sein. Keine Veränderlichkeit/Flexibilität, 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.

[]@l@ P1.3: Benutzerverwaltung

[]@llll@ Nutzer sollen erstellt, bearbeitet und gelöscht werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Mangel an der Anforderung. Abnahme durch Kunde durch Validierung.

[]@l@ P1.4: Experimentier Stationen

[]@llll@ Stationen sollen erstellt, bearbeitet und gelöscht werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Mangel an der Anforderung. Abnahme durch Kunde durch Validierung.

[]@l@ P1.4.1: Auslastung

[]@llll@ Eine Übersicht über die Auslastung der Experimentierstationen soll möglich sein. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Architektur muss speichern, welche Experimentierstationen belegt sind.

[]@l@ P1.4.2: Kaputte Stationen

[]@llll@ Experimentierstationen sollen als kaputt gemeldet werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Anwendung muss mit kaputten Stationen umgehen können: Diese sollten nicht mehr eingeplant werden. Ebenfalls sollten Experimentierstationen für Benutzer als kaputt anzeigbar sein.

[]@l@ P1.5: Prozessschritte

[]@llll@ Prozessschritte sollen erstellt, gelöscht, bearbeitet(sofern noch nicht gestartet), und hervorgehoben(sofern im Zustand “kaputt”) werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen.

[]@l@ P1.6: Prozessketten

[]@llll@ Prozessketten sollen erstellt, gelöscht, und bearbeitet werden können. Keine

Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen.

[]@l@ P1.7: Aufträge

[]@llll@ Aufträge sollen erstellt, freigegeben, gestoppt, gelöscht, priorisiert, und bearbeitet werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen.

[]@l@ P1.7.1: Zuordnen zu Aufträgen

[]@llll@ Aufträgen sollen Proben/Träger zugeordnet werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Architektur muss vorsehen, dass

[]@l@ P1.7.2: Automatisches Zuordnen

[]@llll@ Alternativ sollen Aufträgen automatisch Proben/Träger zugeordnet werden. Keine Veränderlichkeit, aber Flexibilität: 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.

[]@l@ P1.7.3: Alte Aufträge//raus

[]@llll@ Alte Arbeitsaufträge sollen besonders hervorgehoben werden. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Architektur muss die Speicherung vom letzten Änderungsdatum der Aufträge vorgesehen.

[]@l@ P1.7.4: Prioritäten

[]@llll@ Für Aufträge soll eine Priorität errechnet werden. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. //wirklich errechnen?

[]@l@ P1.7.5: Zustand Auftrag

[]@llll@ Der Zustand eines Prozessschrittes (Auftrag im Kontext Technologe) soll manuell aktualisiert werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Architektur muss vorsehen, dass von außen eine Auswahl des Zustandes möglich ist.

[]@l@ P1.8: Träger

[]@llll@ Träger sollen erstellt und gelöscht werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen.

[]@l@ P1.9: Dynamische Zustandsautomaten

[]@llll@ Zustandsautomaten (linear, keine Verzweigungen) für die Prozessschritte sollen vom Prozesskette Administrator angelegt, gelöscht, und bearbeitet werden können. Keine flexibilität, weil es eine Anforderung für das Software ist, in Bezug auf die Fähigkeiten der Rollen. Mangel an der Anforderung. Abnahme durch Kunde durch Validierung.

[]@l@ P1.10: Kaputte Proben

[]@llll@ Proben sollen als kaputt gemeldet werden können. Keine Veränderlichkeit/Flexibilität, 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.

[]@l@ P1.10.1: Verlorene Proben

[]@llll@ Proben sollen als verloren gemeldet werden können. Keine Veränderlichkeit/Flexibilität, 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.

[]@l@ P1.10.2: Lagerübersicht

[]@llll@ Die im Lager liegenden Proben sollen angezeigt werden können. Keine Veränderlichkeit, aber Flexibilität: da vom Chinese Menü, nicht zwingend notwendig 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.

[]@l@ P1.10.3: Gruppen Hervorhebung/raus?

[]@llll@ Es soll einen konfigurierbaren Schwellenwert geben; Proben Gruppen, die unter diesen Wert fallen, sollen hervorgehoben werden. Keine Veränderlichkeit, aber Flexibilität: da vom Chinese Menü, nicht zwingend notwendig

[]@l@ P1.10.4: Proben Kommentar

[]@llll@ Zu Proben sollen Kommentare erstellt werden können. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Es muss eine Fläche für Kommentareingabe und Darstellung geben.

[]@l@ P2: Benutzerschnittstelle

[]@l@ P2.1: Mehrsprachig

[]@llll@ Der User soll zwischen Deutsch und Englisch entscheiden können. Keine Veränderlichkeit, da es vom Chinese Menu ist aber Flexibilität gegeben: muss nicht gemacht werden, um Mindestanforderungen zu erfüllen. Die Architektur muss ein umschalten zwischen den beiden Sprachen in der Darstellung vorsehen.

[]@l@ P2.2: Anzeige

[]@llll@ Je nach Rechten des Benutzers sollen Prozessschritte, -ketten, Stationen, Proben, Benutzer, Aufträge, dynamische Abläufe und/oder Transportaufträge angezeigt werden. Keine Veränderlichkeit/Flexibilität, 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 hierbei teilweise um sehr große Mengen handelt, wie im Falle der Proben.

[]@l@ P3: Verlässlichkeit

[]@l@ P3.1: User Rechte

[]@llll@ Es soll unterschiedliche Rollen mit unterschiedlichen Rechten und angezeigten Informationen geben. Jeder User soll mindestens eine Rolle haben. Keine Veränderlichkeit/Flexibilität, 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.

[]@l@ P3.2: Authentifizierung

[]@llll@ Benutzer sollten sich in das System einloggen müssen, um relevante Informationen angezeigt zu bekommen. Keine Veränderlichkeit/Flexibilität, da Teil der Mindestanforderungen. Die Architektur muss eine login Seite vorgesehen; ein Zugriff auf die Anwendung ohne sich einzuloggen soll nicht möglich sein.

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 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 Gruppe schreibt das Kapitel zur Analyse von Faktoren und zu den Strategien, die andere Gruppe beschreibt die diversen Sichten, ohne dass diese beiden Gruppen sich abstimmen. Natürlich besteht aber ein Zusammenhang zwischen den Faktoren, Strategien und Sichten. Dieser muss erkennbar sein, indem sich die verschiedenen Kapitel eindeutig aufeinander beziehen.

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 versieht. 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.

. . .