Abschlussbericht

Team: Team 1

Mitglied 1: (Michael Hauser ,01267565)

Mitglied 2: (Angela Todhri, 11815296)

Mitglied 3: (Ismail Üner, 11721981)

Mitglied 4: (Flaminia Anselmi, 11934695)

Mitglied 5: (Sebastian Hepp, 01015083)

Mitglied 6: (Maximilian Heine, 01317323)

Proseminargruppe: Gruppe 6

Datum: 18.06.2021

1 Analyse des Projektablaufs

1.1 Projektplan

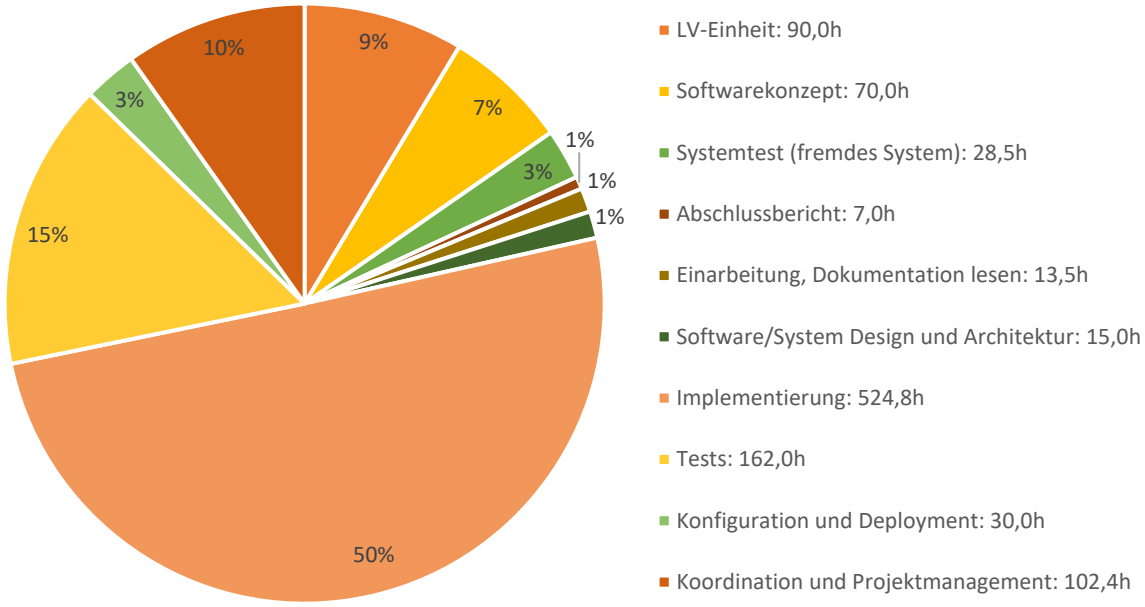
Nr.	Work Package	Responsible	Time	Deadline
1	Konzeptbeschreibung	All	50:00	18.03.
	Milestone 0: Planning			**18.03.**
2	Model	Flaminia / Angela	35:00	25.03.
3	Landing Page	Ismail	10:00	25.03.
4	Raspberry Pi Setup	Michael / Max	15:00	28.03.
5	TimeFlip Setup	Michael / Sebastian	15:00	03.04.
6	REST API	Max / Sebastian	30:00	18.04.
7	Database data	Angela	20:00	01.04.
	Milestone 1: Database, Model and Hardware			**01.04.**
8	MySQL	Flaminia	20:00	01.05.
9	Advanced Backend Functionality Controllers, Services, Repository	All	80:00	13.04.
10	Advanced Frontend	Sebastian / Max	50:00	20.04
11	Statistics & Topics	Ismail / Michael	20:00	27.04
	Milestone 2: Core Functionalities			**01.05.**
12	JSON integration	Flaminia	20:00	02.05.
13	Bugfixing I	All	50:00	09.05.
14	Testdrehbuch	Flaminia / Angela	40:00	13.05.
15	Hardware integration	Michael / Max		12.05.
16	Dockerization	Michael	25:00	06.06.
17	Stable and working system	All		13.05.
	Milestone 3: Releasable System			**13.05.**
18	Acceptance Tests	All		20.05.
19	JUnit Tests	Flaminia / Angela / Ismail	50:00	16.06.
20	Systemtest	Michael / Max	30:00	06.06.
21	Bugfixing II	All	30:00	13.06.
22	Code doku	All	10:00	13.06.
23	Softwarekonzept update	Sebastian	20:00	13.06.
24	Abschlussbericht	Sebastian	10:00	13.06.
	Milestone 4: All Software Revisions & Document Drafts			**14.06.**
25	Documents: Revision & Layout	Sebastian		16.06.
26	**Final project results**	**All**	**10:00**	**18.06.**
27	**Final presentation**	**All**	**08:00**	**21.06.**

Die Meilensteine wurden im Großen und Ganzen eingehalten. Manche Problemstellen wurden teilweise nach hinten verschoben (z.B. Signup). Erstellung, Initiierung und eigentliches Gameplay waren sehr aufwändig, besonders auch durch zusätzlich notwendiges einlesen in HTML und JSF, konnten aber (nach ca. 4 Wochen freizeitlosem Durcharbeiten) dennoch termingerecht abgeschlossen werden. Relativ großer Spielraum vor Abgabe für den Abnahmetest war wichtig um zahlreiche Spezialsituationen behandeln und Fehler beheben zu können.

1.2 Geplanter und tatsächlicher Zeitaufwand

Arbeitspaket	Stunden veranschlagt	Stunden benötigt	Mehr-aufwand
LV-Einheit	108,0	90,0	-17%
Projektmanagement		87,4	
Softwarekonzept	70,0	70,0	+0%
Model	35,0	25,0	-29%
Database Data	20,0	20,0	+0%
MySQL	20,0	20,0	+0%
Hardware setup	40,0	30,0	-25%
REST API	30,0	15,0	-50%
Software core functionalities	172,0	409,3	+138%
Bugfixing	80,0	60,0	-25%
Docker	25,0	15,0	-40%
Testdrehbuch & Abnahmetest	80,0	28,5	-64%
JUnit Tests	50,0	162,0	+224%
Documenting	20,0	11,0	-45%
Summe	750,0	1043,2	+39%

Nach Tätigkeit



2 Analyse des implementierten Systems

Die Use Cases wurden größtenteils kaum verändert. Besonders beim Profil kamen Funktionalitäten hinzu (z.B. Authorisierungsabfragen). Nur die Spielerstellung und der Spielbeitritt wurden stark umkonzeptioniert.

Das Klassendiagramm wurde zu Beginn zu umfangreich und kleinteilig entworfen. Es wurde im Verlauf stark reduziert und vereinfacht. Manche Grundannahmen mussten geändert werden (z.B. den Namen von *Topics* als Primärschlüssel zu verwenden, verhinderte das nachträgliche Umbenennen).

Die Grundlegende Komponentenstruktur hat sich im Projekt bewährt und wurde versucht strikt einzuhalten.

Alle vorgesehenen Funktionalitäten wurden umgesetzt, bis auf ein Aktivitätslogging mit Rücksetzfunktion. Es gibt allerdings nur zwei systemkritische Stellen bei denen dies sinnvoll gewesen wäre:

- Löschen von *Topics*:
  Topics können sehr viele Einträge enthalten. Unvorsichtiges Löschen kann daher zu teurem Verlust führen.
- Löschen von Benutzerkonten

Dies Fälle wurden anderweitig abgesichert:

- *Topics* lassen sich nur löschen, wenn sie keine Einträge enthalten. Die Einträge können einzeln gelöscht werden.
- Das endgültige Löschen eines Benutzerkontos erfordert insgesamt 5 Einzelschritte. Solange Nutzer eingeloggt sind, können sie nicht gelöscht werden.

Die Entwicklungsumgebungen lieferten hilfreiche Werkzeuge. Darunter die Bearbeitungsansicht bei Git-Merge-Konflikten, die es erlaubt übersichtlich sich unterscheidende Versionen einer Klassendatei zusammenzuführen. Außerdem wurden die Werkzeuge zur Refakturierung oft benötigt, um zur besseren Verständlichkeit Methoden umzubenennen oder aufzuteilen, Teile von Methoden auszugliedern, da sie mehrfach benötigt wurden, oder um zu groß gewordene Klassen aufzuteilen.

3 Ursachenanalyse

Längere Konzeptionsphase

Das Softwarekonzept sollte iterativ entstehen: Sofort mit einer Vielzahl viel zu konkreter Use-Cases zu beginnen war nicht sinnvoll. Sinnvoller wären zum Beispiel Flussdiagramme zur Erfassung der benötigten Funktionalitäten gewesen. Diese sollten dann zusammen mit einem Implementierungskonzept der Funktionalitäten schrittweise verbessert, angepasst und konkretisiert werden. Erst dann kann ein sinnvoller konkreter Projektplan hinsichtlich aller Implementierungsaufgaben erstellt werden. Daraus kann dann ein Softwarekonzept abgeleitet werden, das die Projektarbeit tatsächlich koordinieren kann.

Methoden zur effizienten Team- und Kollaborationsarbeit finden:

- Wie teilt man ein Projekt in sinnvolle Arbeitspakete?
- Wie ermöglicht man effiziente Zusammenarbeit und verhindert Konflikte und Überschneidungen?

Regelmäßige Aktivitätsplanung:

Beispielsweise nennen alle Beteiligten für die nächsten 2 Wochen ihre geplanten Zeiteinsätze, welche Aufgaben übernommen werden und wann ein Ergebnis geliefert werden kann. Der 2-Wochen-Plan wird dann für alle zur Einsicht dargestellt, damit jeder den Überblick behält.

Wir hatten zwar regelmäßige Koordinationsbesprechungen, aber diese waren zu wenig konkret und wurden nicht dokumentiert.

JUnit-Tests parallel zur Implementierung

- Ermöglicht besseren Überblick über Teststatus aller Funktionen
- Zeitnahes Implementieren und Testen ist effizienter
- Nach kleineren Änderungen kann die Funktionalität sofort überprüft werden
- in der knappen Zeit unmöglich, Funktionalität musste priorisiert werden
- Nachteil: Wenn sich Methoden komplett ändern, war das Schreiben der Tests vergeudete Zeit.

Gemeinsame Arbeitsphasen:

- ermöglicht leichteren Wissensaustausch
- in Corona-Zeiten schwierig
- verursacht auch zeitlichen Overhead (bei Zeitdruck wäre das weniger möglich)

4 Erfahrungen mit den eingesetzten Werkzeugen

Das Spring-Framework ist sehr umfangreich und nimmt von Webkonfiguration bis Datenbankmanagement viel Arbeit ab. Da allerdings nicht bekannt ist, was tatsächlich im Hintergrund passiert, ist das Finden und Beheben von Fehlerquellen oft mühsam und zeitaufwendig. Dasselbe gilt für HTML mit Primefaces-Integration, wobei Primefaces nach kurzer Einarbeitung relativ einfach zu verwenden war und die Dokumentation mit vielen Fallbeispielen sehr hilfreich ist.

Besonders das Löschen aus der Datenbank ist über das Framework aufgrund der Abhängigkeiten zwischen den Entitäten oft sehr kompliziert; teilweise hat es nicht funktioniert und es mussten Workarounds gefunden werden.

Das Arbeiten mit REST ging nach kurzem Einarbeiten und Einlesen in das Thema leicht von der Hand. Auch der Workshop zu diesem Thema war eine gute Hilfe. Das Programmieren der Endpoints im Spring Projekt geht mittels der Annotationen sehr gut. Innerhalb des Raspberry Projekts haben wir Unirest-Java verwendet, was wiederum eine angenehme Lösung bietet Requests abzusenden. Die Dokumentation war hier auch eine große Hilfe. Am meisten Zeit in Anspruch nahm die Authentifizierung. Hier schien die von Spring angebotene Lösung zu kompliziert, weshalb wir selbst eine Authentifizierungs-Methode schrieben.

Workshop und eigene Versuche lieferten einen guten ersten Einstieg in die Verwendung der Hardware und die Kommunikation über BLE. Eine Schwierigkeit war zunächst die Ausgabe des TimeFlips in Form von Bit-Streams korrekt interpretieren zu können. Die Library TinyB hat diese Aufgabe sehr erleichtert. Da sie in Java geschrieben ist, war sie für uns gut verständlich und aufgrund zahlreicher zugehöriger Beispiele einfach nachvollziehbar.

Docker ist ein sehr angenehmes Werkzeug für das Deployment. Das Erstellen des Containers erfordert moderaten, aber einmaligen Aufwand. Von den Endnutzern dagegen wird kein projektspezifisches Wissen verlangt. So war es zum Beispiel für den Abnahmetest für uns sehr einfach, auch ohne irgendwelches Wissen über das andere Projekt, alles einzurichten und auszuführen.

5 Feedback zur Proseminar-Organisation

Verbesserungswürdig

- Das Testdrehbuch-Dokument sollte sich entweder als ausfüllbares Formular in PDF umwandeln lassen oder als Word dem anderen Team zur Verfügung gestellt werden.

Zeitaufwand vs. Umfang laut Curriculum

Die Lehrveranstaltung verlangte uns insgesamt ca. 1030 Arbeitsstunden ab, wogegen im Curriculum insgesamt 750 Stunden vorgesehen sind. Sie ist somit für ein Semester viel zu umfangreich.

Durch den stark überzogenen Zeitaufwand werden nicht nur einfach die Studenten etwas strapaziert, sondern es ergeben sich konkrete Nachteile, die die effektive Qualität der LV auf ein enttäuschendes Maß verringern.

Die LV lässt sich sicherlich als eines der zentralen Elemente des Informatikstudium sehen. Von der LV erhofft wurden sich

1. ein Softwareprojekt ungefähr von Anfang bis Ende durchzuspielen
2. Projektmanagement erlernen
3. lernen qualitativ gute Software zu produzieren

Nur den ersten Punkt lernt man in dieser LV effektiv, was grundsätzlich positiv ist, aufgrund der viel zu knappen Zeit aber unmöglich bei den obigen positiven Punkten aufgenommen werden kann (siehe dazu auch Lösungsvorschlag).

Die beiden letzten werden in dieser LV nur unzureichend vermittelt.

- Projektmanagement:
  Wie wiederholt im Studium erklärt, ist das Projektmanagement äußerst wichtig im Softwareengineering. Dennoch wurden viel zu wenig Zeit veranschlagt um das Projekt ausreichend im Vorhinein zu planen.
  Eine ausreichende Analysephase und ein schrittweises Entwickeln eines Konzepts und Projektplanes vom groben zum Ausführlichen wären wichtig gewesen (siehe dazu auch Punkt 3 *Längere Konzeptionsphase*).
  Mehr Hilfestellung und Anregungen bezüglich Projektplanung, Werkzeugen für Projektmanagement, verwenden von agilen Methoden wären sehr hilfreich gewesen.
  Die entsprechenden Themen in der VO sind grundsätzlich interessant und bestimmt hilfreich, kamen aber zu spät im Semester, um noch eingebunden werden zu können. Es war auch keine Zeit dafür vorhanden.
  Wenn zum ersten Mal ein solche Softwareprojekt angegangen wird, ist es dadurch kaum möglich eine sinnvolle, für den weiteren Verlauf stabile und wirklich hilfreiche Projektkonzeption zu formulieren. Somit ist das Projekt unkoordinierter verlaufen als es hätte müssen.
- Quantität vor Qualität:
  Es wurde versucht alles von Konzeption über Java, Datenbankanbindung und Webapplikation bis hin zu Testen und Deployment in den Kurs zu packen, ohne dafür ausreichend Zeit vorzusehen. Darunter leidet schlussendlich die Qualität. Ziel einer Universitätsbildung sollte nicht sein möglichst viel in möglichst kurzer Zeit zu *produzieren*, sondern möglichst qualitativ hochwertiges Arbeiten zu erlernen.
- Um Workshops zu besuchen, die man für seine übernommenen Aufgaben nicht unbedingt benötigte, blieb oftmals keine Zeit, auch wenn sie interessant gewesen wären.
- Sich ausreichend einzulesen und einzuarbeiten ist nur schwer möglich, wenn die Umsetzung so zeitaufwendig ist, dass dann Gefahr besteht die Funktionalitäten nicht zeitgerecht implementieren zu können (siehe auch Punkte *Qualität* und *Lösungsvorschlag*).
- Damit die Arbeit schaffbar ist, muss sie konsequent aufgesplittet werden:
  So können sich nur wenige einzelne Mitglieder mit Hardware, REST und Docker beschäftigen, obwohl es für alle interessant wäre. Aufgaben müssen an diejenigen verteilt werden, die sich bereits am besten mit einem Thema auskennen, was aber nicht sinnvoll für das Lernen ist.
- Keine Zeit für graphisches Aufbereiten:
  Wohl kein besonders wichtiger Teil, aber einer der Spaß macht. Es enttäuscht, wenn man ein Semester an einem Projekt arbeitet und dann keine Zeit hat es wirklich auch graphisch ansprechend aufzubereiten.

- Mit mehr Zeit hätte man sich besser einlesen, mit Themen auseinandersetzen und die Arbeit verbessern können. Dadurch hätte man nicht nur mehr lernen können, sondern auch die Qualität des gelernten hätte sich stark verbessert.

Lösungsvorschlag:

a. (schlecht:) den Umfang reduzieren
b. (besser und wie in anderen Studiengängen seit langem normal, in der Informatik aber aus unnachvollziehbaren gründen scheinbar undenkbar:)
   2-semestrige LV
   - Möglicherweise ließe sich die LV mit dem PS Softwarearchitektur zusammenlegen. Besonders das Projekt in Softwarearchitektur überschneidet sich unnötigerweise völlig mit dem PS Softwareengineering und verschwendet dadurch nur Zeit, die sich besser in Planung oder andere Vorarbeit investieren ließe.

Es ist zudem leider notwendig anzumerken, dass sich die Enttäuschung über die LV zusätzlich verstärkt, wenn man bedenkt, dass dieses Problem schon vom letzten Jahr bekannt ist und auf die damalige Kritik nicht nur nicht eingegangen wurde, sondern sich die Aufgabenkomplexität sogar noch erhöht hat.

Positives

- Workshops waren hilfreich und informativ.
  Projektmanagement-Teil könnte mehr konkrete Werkzeuge (hinsichtlich agilem Projektmanagement) vorstellen.
- Die StatusQuo-Gespräche waren gut um einen Überblick zu erhalten, wie die anderen Teams vorgehen und um die eigene Arbeitsqualität einschätzen zu können.
- Abnahmetests waren sehr hilfreich:
  - Viele Probleme findet man nicht.
  - Man wird schnell blind für eventuelle Verwendungs-/Nachvollziehbarkeitsprobleme des eigenen Projekts.
  - Manche Probleme erscheinen erst, wenn das Programm in anderer Umgebung ausgeführt wird.
- Der Zeitraum zwischen Abnahmetests und Endabgabe ist sinnvoll und angemessen. Hier kann die Software gut überarbeitet und verbessert werden.
- Gute Kommunikation mit der LV-Leitung: Feedback, Beantwortung von Fragen
- LV-Leiter war positiv, verständnisvoll und hilfsbereit, was die Projektarbeit trotz den Umständen (wie weiter oben aufgeführt) angenehm gemacht hat.