Abschlussbericht

Team: prodiga/4

Mitglied 1: Laura Geiger 11831841

Mitglied 2: Jamie Hochrainer 1180630

Mitglied 3: Gabriel Mitterrutzner 11832162

Mitglied 4: Maximilian Suitner 11832061

Mitglied 5: Georg Wenzel 11832025

Proseminargruppe: 4 später 1

Datum: 13.06.2020

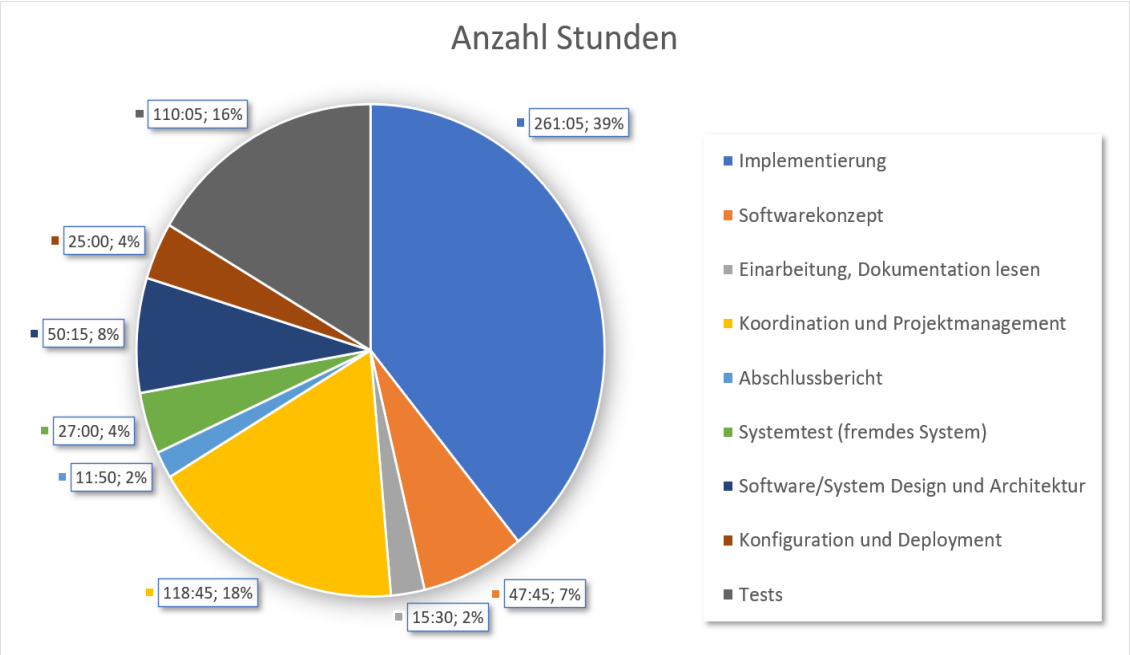
1. Analyse des Projektablaufs

Obwohl uns der große Arbeitsumfang bewusst war und wir das Auftreten von manchen “Bugs” erwartet und deren Behebung einkalkuliert haben, war der tatsächliche Arbeitsaufwand des Projektes weit über dem geschätzten.

Wir haben uns stets an die gesetzten Meilensteine gehalten und diese auch mit “Issues” fixiert, wobei das ein oder andere kleinere “Issue” manchmal auch eine Woche danach erledigt wurde, was jedoch zu keinerlei Problemen geführt hat.

Unterschätzt haben wir den Umfang von unserem Meilenstein Unterpunkt “Auf Abnahmetest vorbereiten”, weswegen wir die Tage vor der ersten Abgabe des Projekts einige Nachtschichten einlegen mussten, um das Testdrehbuch, die Konzeptbeschreibung und die Testabdeckung fertigzustellen.

Die vielen Teambesprechungen (fast zweimal die Woche) haben uns sehr geholfen, um einen besseren Überblick über unseren Projektfortschritt zu bekommen, weshalb beim Einteilen der “Issues” kaum Unklarheiten auftraten, was ein rasches und flüssiges Arbeiten ermöglichte. Zusätzlich konnte durch die laufende Kommunikation auch sichergestellt werden, dass jedes Teammitglied sich gleichermaßen am Projekt beteiligt.



Hier zusehen ist eine Übersicht aller geleisteten Stunden pro Kategorie über alle Teammitglieder summiert. Auffällig ist, dass knapp 20% des gesamten Projekts nur für Koordination verwendet wurde, darin fallen die vielen Teammeetings und das Verwalten von Issues. Außerdem sind die 50 Stunden Software Design anzumerken. Bevor einige wichtige Features implementiert wurden, haben wir uns vorher viele Gedanken über die Implementation gemacht. Dies resultiert in einer sauberen Softwarearchitektur.

2. Analyse des implementierten Systems

Die geplanten und im Konzept versprochenen Funktionalitäten konnten alle zeitgemäß umgesetzt werden. Die Ansicht der Funktionalitäten weicht jedoch etwas von den allerersten GUI-Prototypen ab.

Sowohl das Komponentendiagramm, als auch das Klassendiagramm weichen in der finalen Version der Software vom ursprünglichen Design ab.
Auf dem im Komponentendiagramm beschriebenen Hardware-Service wurde komplett verzichtet. Die geplante Logik darin (Auslesen von Hardware Daten und Speichern von Aktivitäten, die vom REST-Client erfüllt werden müssen) wurde in mehrere Services/REST-Controller aufgeteilt. Des Weiteren wurden mehrere, kleinere Services hinzugefügt, welche die Verwaltung von Entitäten übernehmen. Die restlichen Komponenten wurden wie beschrieben implementiert und übernehmen auch die beschriebenen Aufgaben.
Das fachliche Klassendiagramm hat sich sehr stark zur Ursprungsversion verändert. Dies hat vor allem mit dem TimeFlip Würfel zu tun, da dieser vor der Umsetzung eine Blackbox für sämtliche Entwickler war. Vor allem wurde auf das direkte Mapping zwischen Aktivitätstyp (im Klassendiagramm BookingCategory) und Würfel sowie die Zuweisung zwischen Bookings und dem TimeFlip Würfel verzichtet.

Die Verwendung von Continuous Integration hat die Qualität der Software stets verbessert. Wurde eine Komponente so stark verändert, dass die Funktionalität der Software nicht mehr gegeben war, konnten wir sofort auf den Fehler eingehen und entsprechend beheben.
Die Verwendung von SonarQube war hingegen nicht hilfreich, die vorgeschlagene Verbesserung bringen meist keine höhere Qualität. Für den Gewinn, den SonarQube gebracht hat, war der Zeitaufwand das System zu installieren nicht proportional.

3. Ursachenanalyse

Da die GUI oft sehr stark von dem darunterliegenden Model abhängt, ist es schwer vorauszusagen, wie die finale Benutzeroberfläche aussieht. Für eine konsistente Oberfläche müsste auch ein konsistentes Datenmodell gegeben sein. Für ein gründliches Datenmodell müsste vor Projektbeginn die involvierte Hardware (in unserem Fall der TimeFlip Würfel und der Raspberry Pi) genauer studiert werden. Da einige Teammitglieder bereits Erfahrung mit Server-Client Architektur bzw. Dem Raspberry Pi haben, waren diese Voraussetzungen erfüllt. Der TimeFlip Würfel war jedoch für alle Teammitglieder neu, daher mussten wir unser Datenmodell stark ändern, um die Datenstruktur, welche den Würfel repräsentiert, leichter in unser System integrieren zu können. Hier wäre eine gründlichere Arbeitsteilung notwendiger gewesen, bspw. könnten drei Teammitglieder die Business Logik, die unabhängig von Würfel und Client ist, implementieren, während das restliche Team sich ausschließlich mit Würfel und RaspberryPi beschäftigen.

Grundsätzlich ist es jedoch immer schwer, vor Projektbeginn zu wissen, wie die finale Software aussieht bzw. aufgebaut ist. Unsere Software hatte trotz teilweiser großer Änderungen jedoch zu jedem Zeitpunkt ein hohes Maß an Qualität.

4. Erfahrungen mit den eingesetzten Werkzeugen

4.1. Spring und JSF/Primefaces

Hier half uns die Erfahrung, die wir bereits letztes Semester gesammelt haben, sehr. Viele Fehler konnten leicht behoben werden, da wir diesen bereits begegnet sind. Auch einige Features, die dem letzten Projekt ähneln, konnten wiederverwendet werden.
Jedoch gab es auch bei diesem Projekt wieder Probleme, die nicht “schön” gelöst werden konnten. Bspw. konnte nicht mit Javas Futures gearbeitet werden, da die Unit-Tests keine ThreadExecutor unterstützen. Die Dokumentation beider Frameworks fehlt teils komplett, bzw. muss sich aufwendig aus mehreren Stackoverflow Posts selbst angeeignet werden.

4.2. MySQL

Die Verwendung von MySQL war eine deutliche Verbesserung zur In-Memory H2 Datenbank. Man konnte SQL-Abfragen einfacher testen bzw. optimieren, da eine bessere Auswahl an Client-Tools gegeben ist.

4.3. Docker

Die Verwendung von Docker war vor allem für die gegenseitigen Abnahmetests sehr hilfreich, da es theoretisch ein vom Betriebssystem unabhängiges Ausführen erlaubt. Jedoch gibt es bei der Ausführung derselben docker-compose Datei auf verschiedenen Betriebssystemen oft Inkonsistenzen, was das Schreiben einer qualitativ hochwertigen docker-compose Datei stark erschwert.

4.4. REST-API und REST-Client

Da einige Teammitglieder hier bereits sehr viel Erfahrung Gesammelten haben, fiel die Entwicklung der REST-Schnittstelle sehr leicht. Positiv zu erwähnen war die Verwendung von Swagger, was ein sehr einfaches Testen und eine einfache Dokumentation der API ermöglicht hat. Außerdem konnte der Client-Code vollständig generiert werden, was die Entwicklung sehr beschleunigt hat. Schwierigkeiten gab es jedoch bei der Implementierung der Authentifizierung. Da sowohl REST-API und die Website selber den gleichen Server-Code teilen, war die Verwendung von Basic-Auth in Kombination mit JWT-Auth eher schwierig. Die Dokumentation von Spring war hier eher mangelhaft, mit anderen Technologien wäre diese duale Authentifizierung sehr einfach gewesen. Um die Authentifizierung in Spring zu verwirklichen, mussten einige “Hacks” eingebaut werden, die in einem Produktivsystem unmöglich zu verwenden sind.

4.5. TimeFlip

Die Einbindung eines IoT Systems war an sich eine gute Idee, um das Projekt spannender zu machen und mehrere Technologien zu verwenden. Da aber die Dokumentation des TimeFlip Würfels sehr oberflächlich war und man sich zusätzlich in die Schnittstelle Bluetooth LE einlesen musste, war das System sehr mühsam zu testen. Ein großes Problem war zusätzlich die Instabilität des Würfels, da die Verbindung zufällig unterbrochen werden konnte, die BLE Services zum Teil nach einem schnellen Neustart nicht mehr angeboten wurden, die Seiten ID's sich nach einem Batteriewechsel ändern, usw.. Die Entwicklung eines stabilen Clients umfasste daher zu einem großen Teil lediglich das "Stabilisieren" des IoT Systems.

4.6. Zusammenfassung

Auch für dieses Projekt wäre eine freie Technologiewahl wünschenswert. Das Ziel jedes Softwareprojektes ist es, eine innovative Lösung für ein Problem zu finden. Innovative Produkte können jedoch nur mit innovativen Technologien verwirklicht werden, was für die vorgeschriebenen Systeme und Frameworks jedoch nur teilweise zutrifft.

5. Feedback zur Proseminar-Organisation

Die Durchführung des Projektes war sehr interessant und lehrreich. Wir finden jedoch, dass der tatsächliche Arbeitsaufwand nicht in Relation mit den ECTS steht. Verbessern könnte man dies, indem entweder weniger umfangreiche externe Komponenten gewählt werden (z.B. kein TimeFlip Würfel und kein Docker) oder weniger Leistung innerhalb des bestehenden Projektes gefordert wird (z.B. geringere Testabdeckung, geringere Featuredichte).

Die Durchführung von Workshops war zwar eine sehr gute Idee, jedoch waren einige Workshops für die Entwicklung wenig hilfreich. Beispielsweise wurden einige wichtige Dokumente beim Testing-Workshop nicht erwähnt, und manchmal hatte man das Gefühl, dass die Workshopleiter sich gegenseitig widersprechen.

Positiv sind auch die regelmäßigen Projektpräsentationen zu erwähnen, sie fördern einen kontinuierlichen Verbesserungsprozess und zeigen den Fortschritt im Vergleich zu anderen Gruppen.

Die besondere Abwicklung des Proseminars wurde gut gelöst, jedoch gab es teilweise Probleme mit Adobe Connect, welche in anderen Proseminaren nicht gegeben aufgetreten sind. Die Qualität des Streams ist eher niedrig im Vergleich zu anderen möglichen Streaming-Plattformen, und es ist auch nicht auf allen Plattformen funktional. (z.B. Linux)

Außerdem ist uns aufgefallen, dass die Proseminar-Leitung häufig Mails über eine ungleichmäßig verteilte Aufgabenverteilung versendet, obwohl dies nicht der Fall ist. Die Zuständigen für den Timeflip-Würfel müssen sich viel mehr Einlesen und haben deshalb eine geringere Anzahl an "commits". Dies sollte aufgrund der aktuellen Situation (nur eine Person hat den TimeFlip-Würfel und Raspberry Pi) von

der Proseminar-Leitung berücksichtigt werden. Hier sollte mehr Vertrauen in die Studierenden gesetzt werden, bei Problemen selbst die Proseminar-Leitung anzusprechen.

Ebenso erhielten wir in der vorherigen Proseminar-Gruppe wenig positives Feedback. Unser Proseminar-Leiter hat von allen Gruppen zu viel gefordert und großteils nur negative Punkte angesprochen. Dies erachten wir als wenig motivierend und als enormen Druck.

Nach dem Wechsel der Gruppe erhielten wir deutlich mehr positives Feedback und wurden sehr gut durch das Projekt geleitet. Unser Arbeitsfluss verbesserte sich dadurch auch und man hatte nicht ständig das Gefühl im Verzug zu sein. Generell war die Kommunikation zwischen Proseminar-Leiter und Studenten durchaus kollegialer und freundlicher. Auch das Stellen von Fragen und Anliegen war dadurch erleichtert.