Lessons Learned

(TINF19C, SWE I Praxisprojekt 2020/2021)

Project: *Modelling Wizard for Device Descriptions*

Customer: *Rentschler & Holder*

Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 2:

PL: Stefan-Nemanja Banov(inf19014@lehre.dhbw-stuttgart.de)

PM: Timo Zaoral (inf19133@lehre.dhbw-stuttgart.de)

SA: Simon Jess(inf19182@lehre.dhbw-stuttgart.de)

TM: Jakob Schmidt (inf19205@lehre.dhbw-stuttgart.de)

TR: Tobias Roth (inf19202@lehre.dhbw-stuttgart.de)

LE: Thanh Vu Phillip Tran (inf19105@lehre.dhbw-stuttgart.de)

Version	Date	Author	Comment
0.1	01.04.2021	Stefan-Nemanja Banov	Dokument erstellt und erste Stichpunkte aufgeschrieben.
1.0	13.05.2021	Stefan-Nemanja Banov	Nach Interviews mit allen Teammitgliedern zusätzliche Punkte ergänzt.
2.0	15.05.2021	Stefan-Nemanja Banov	Stichpunkte ausformuliert und Dokument fertig gestellt nach Absprache mit dem Kunden

Inhalt

1. Einleitung 3
2. Positive Erfahrungen 3
3. Negative Erfahrungen 4
4. Was das Team daraus gelernt hat 6

1. Einleitung

Das Projekt „Modelling Wizzard for Device Descriptions“ mit dem Kunden Rentschler & Holder wurde über einen Zeitraum von 6 Monaten intensiv bearbeitet. Dabei gab es immer wieder Probleme, welche das Team vor verschiedene Hindernisse gestellt hat. Dieses Dokument soll sowohl Probleme als auch Dinge die gut liefen identifizieren. Anschließend kann man die gewonnen Informationen dazu nutzen, neue Lektionen zu identifizieren. Mithilfe dieser Lektionen können zukünftige Probleme mit neuen oder bestehenden Kunden vermieden oder schneller gelöst werden. Diese Lektionen helfen dem Team, effizienter und besser zu Arbeiten.

2. Positive Erfahrungen

Das Projekt war schwierig und hat von allen Teammitgliedern einiges abverlangt. Wie es üblich bei „Lessons Learned“ ist wird auch hier erst auf die positiven Erfahrungen, welche während des Projekts gesammelt wurden, eingegangen.

Da es bei einem Projekt dieser Größe ohne Dokumentation schnell in Chaos ausarten kann, hat sich das Team dazu entschlossen GitHub als Medium zu benutzen um alles was gemacht wird zentral zu speichern und ordentlich zu dokumentieren. Die Erstellung des GitHub Repositories verlief ohne Probleme. Über das ganze Projekt hinweg hat jeder seine Pushes und Merge Request sauber betitelt. Wenn Probleme erkannt wurden, sind diese auch als Issues getracked worden. Falls etwas undeutlich oder unklar für die anderen Teammitglieder war, wurde die Person angesprochen, sodass alles was gemacht wurde auch transparent für alle war.

Dieses Projekt wäre ohne dieses Team nicht in so einem guten Zustand fertig gestellt worden. Es ist wahr, dass es noch einige Anforderungen gibt, welche nicht vollends fertig gestellt worden sind. Allerdings sind alle ursprünglichen Anforderungen, welche das Team am Anfang des Projekts erhalten hat, gänzlich fertig gestellt. Im Laufe des Projektes gab es aber immer wieder zusätzliche Anforderungen von dem Kunden, von denen auch die meisten Umgesetzt werden konnten. Nur durch den Zusammenhalt und die unermüdliche Motivation des Teams waren auch die zusätzlichen Aufgaben zu bewältigen.

Wie aus dem Projekthandbuch schon hervorgeht ist das Team mithilfe des Scrum Prinzips geleitet worden. Im Gegensatz zu der ersten Hälfte des Projektes wo das Team wöchentliche Sprints absolviert hat, wurde in der zweiten Hälfte des Projektes die Länge eines Sprints auf zwei Wochen erhöht, da die Entwicklungen längere Zeit brauchten. Was besonders positiv auffällt ist das jedes Teammitglied bei allen Meetings anwesend war. Darüber hinaus waren die Teammitglieder nicht nur anwesend, sondern haben auch proaktiv an dem Meeting teilgenommen. Üblicher Weise wurde das Team in die Agenda des aktuellen Sprint Reviews eingeweiht, anschließend haben sich die Teammitglieder gegenseitig Feedback zu ihren Dokumenten oder Entwicklungen gegeben und sich gegenseitig aufgebaut falls jemand die Motivation verloren hat. Durch dieses proaktive Arbeiten der Teammitglieder und eines dynamischen und detaillierten Projektplans konnte das best- mögliche Ergebnis erzielt werden.

Ein weiterer wichtiger Punkt, welcher die einen oder anderen Missverständnisse verhindert hat, war die ständige Kommunikation mit dem Kunden. In der ersten Hälfte des Projektes ging es hauptsächlich um den richtigen Rahmen, in dem die Dokumente für die Dokumentation erstellt werden und ein erster Prototyp, um mit dem Kunden auf einen Einheitlichen Stand zu sein. In der zweiten Hälfte wurde dann mit der Entwicklung fortgesetzt. Nach jedem größeren Update wurde der Kunde mit eingebunden und nach Feedback gefragt, damit nicht an dem Kunden vorbei entwickelt wird.

Aus den oben erläuterten Geschehnissen kann man zusammenfassend folgende Stichpunkte herauskristallisieren:

- Erstellung des GitHubs und Pflege.
- Zusammenhalt des Teams war sehr gut. Wenn jemand nicht weiterwusste, konnte ein Teammitglied ihm meistens helfen.
- **Alle** Teammitglieder bei **JEDEM** Meeting anwesend und proaktiv beteiligt.
- Proaktive Entwicklung mit dem Kunden -> Kunde wurde in alle Schritte/Änderungen einbezogen. Unbedingt beibehalten!
- Teammitglieder haben sich gegenseitig motiviert.

3. Negative Erfahrungen

Nachdem über die positiven Erfahrungen gesprochen wurde, wird auf die Probleme und Hindernisse eingegangen, welche dem Team im Verlaufe des Projekts widerfahren sind.

Nachdem in Abschnitt 2. über die Vorteile von GitHub gesprochen wurde gibt es auch einige Probleme, die dadurch entstanden sind. Nicht jedes Teammitglied hat vorher mit GitHub gearbeitet und für viele war es das erste Mal in diesem Umfang. Anfangs hat sich der Projektmanager mit allen Teammitgliedern hingesetzt und ihnen grob erklärt wie man GitHub auf dem privaten Rechner installiert, sodass auch alle damit arbeiten konnten. Doch nicht bei allen hat es auf Anhieb funktioniert und es gab immer mal Probleme bei dem Teilen von Daten, welche sich erfahrenere Teammitglieder anschauten, um es zu lösen. Ein weiteres Problem, welches sich durch die Nutzung von GitHub ergeben hat, war die Benutzung der sogenannten Branch Funktion. Anfangs hat sich das Team darauf geeinigt, dass jeder seinen eigenen Branch bekommt, was nach genauerer Überlegung keinen Sinn gemacht hat. Anschließend wollte das Team für jedes Modul einen Branch erstellen. Danach wurden die Branches in Funktionale und Nicht-Funktionale Änderungen an dem Modelling Wizzard aufgeteilt, bis schließlich sich der Projekt Manger reichlich informiert hat. Das Endergebnis worauf sich dann auch jeder geeinigt hat ist, dass die Branches so genutzt werden wie sie auch eigentlich angedacht wurden. Ein Branch für die aktuelle funktionierende Version des Projektes und ein Branch für alle Entwicklungen, welche noch nicht eingehend getestet wurden und somit noch nicht den Produktionsstatus erlangt haben. Auf diese Weise können schließlich verschiedene Versionen des Modelling Wizzards erstellt werden.

Wie aus dem Projekthandbuch hervorgeht ist das Projekt keine Eigenentwicklung, sondern wurde von einem anderen Unternehmen so an unser Team übergeben. Es gab für das Team keine Möglichkeit, vorab den aktuellen Stand des Vorgängerprojekts zu überprüfen. Das Team wurde nicht

richtig in die Funktionalität des Programms eingeführt. Zwar gab es einen Termin, in dem ein Workflow mit dem Modelling Wizzard gezeigt wurde, aber war diese Einführung nicht informativ genug. Der Kunde hat die Anforderungen an das Projekt außerdem sehr schwammig und nicht genau genug formuliert. Es ist jedoch in der Verantwortung des Teams nicht nur während des Projekts, sondern auch vor Projektstart genaue Anforderungen und Rahmenbedingungen von dem Kunden zu ermitteln.

Der Code des Vorgängerprojekts war unstrukturiert, unleserlich, unkommentiert und zusammengefasst einfach nicht verständlich für außenstehende. Deshalb hat das Team sehr viel Zeit damit verbracht den Code zu analysieren und einen Weg zu finden die Anforderungen des Kunden umzusetzen. Zeitweise war auch die Überlegung des Teams das Vorgängerprojekt nicht zu verwenden und von vorne anzufangen. Nach langem recherchieren konnte der Projektmanager jedoch den Verantwortlichen für das Vorgängerprojekt kontaktieren und es wurde sich auf ein Treffen geeinigt, wo der Verantwortliche seinen Code erklärt hat und das Team richtig in alle Funktionalitäten und deren Codestellen eingewiesen hat. Mithilfe dieser Einweisung konnte das Team alle wichtigen Codestellen identifizieren und mit den funktionalen Änderungen beginnen.

Während der Entwicklung gab es auch immer wieder Probleme, die einige Mitglieder an den Rand der Verzweiflung gebracht haben. Nicht nur gab es immer wieder Bugs, sondern einfach Funktionalitäten, welche angeboten wurden, aber in Wirklichkeit nicht einmal implementiert sind. Zudem kam es auch durch einen Fehler bei der Entwicklung des Vorgänger Teams zu unwiderruflich gelöschten Inhalten der Teammitglieder. Eine fehlerhafte Speicherfunktion hat das Löschen aller Dateien in einem Verzeichnis bewirkt. Nicht nur das Lösen dieses Problems hat das Team Zeit gekostet, sondern auch die Implementierung der meisten Funktionalitäten, welche schon längst funktionieren sollten.

Zusammenfassend kann man sagen, dass das Team sehr durch dieses Projekt gefordert wurde. Die Zeit, die in dieses Projekt gesteckt wurde, hat auch andere Projekte des Teams negativ beeinflusst. Mit der Qualität des Projekts ist das Team zufrieden, jedoch ist das Projekt wirtschaftlich gesehen schlecht verlaufen, da parallellaufende Projekte in Mitleidenschaft gezogen wurden. Aus den eben genannten Erlebnissen können folgende Stichpunkte zusammengefasst werden:

- GitHub Funktion der Branches waren nicht klar.
- GitHub Probleme bei der Anbindung, durch Unwissenheit.
- Anfangsanforderungen des Kunden sehr schwammig formuliert.
- Unklar wie mit den Programmen des Kunden gearbeitet wird.
- Vorgängerprojekt unstrukturiert, unleserlich, unkommentiert und nicht verständlich, daher ein enormer Zeitaufwand für kleinste Veränderungen.
- Viele Daten verloren gegangen, da das Vorgängerprogramm einfach Ordner löscht.
- Das Vorgänger Team war schwierig zu kontaktieren, was unserem Team Zeit gekostet hat.
- Projekt war teilweise von der Entscheidung einer Person abhängig.

4. Was das Team daraus gelernt hat

Nachdem sowohl die positiven als auch die negativen Erlebnisse beschrieben wurden, kann man nun Lektionen festlegen. Diese Lektionen sollen es dem Team später ermöglichen weniger Probleme während ihrer Projekte zu begegnen. Um vollständig vorbereitet zu sein sollte das komplette Team vorher an einem GitHub Workshop teilgenommen haben. Durch diesen Workshop ist das Team optimal auf eine Zusammenarbeit mit GitHub geschult.

Die zweite und auch größere Lektion ist einen Kick-Off mit dem Kunden zu planen und alle Stakeholder an einen Tisch zu kriegen. Dieser Kick-Off kann auch über mehrere Tage gehen. Mithilfe dieses ausführlichen ersten Termins kann das gesamte Team auf den gleichen Stand des Kunden gebracht werden, hier werden unter anderem Kontaktdaten das vorherigen Entwicklerteams ermittelt, falls es einen geben sollte. Zusätzlich werden hier auch alle erforderlichen Workflows, bekannte Probleme, vorgestellte Verbesserungen, komplette Neuentwicklungen aufgezeigt. Das Team sollte nach diesem ersten Kick-Off wissen was zu tun ist und den Kunden nicht mehr löchern müssen.

Nachdem das Team sehr viel Zeit damit verbracht hat, den Code zu analysieren und nicht durch nur Eigeninitiative dieses Konstrukt vollständig verstehen konnte, ist es Sinnvoll bei Entwicklungen, die nicht durch unsere Firma durchgeführt wurden, eine vollständige Codeanalyse vor Projektstart durchzuführen. Hierbei kann ohne viel Aufwand erkannt werden in welchem Zustand sich der Code befindet. Es werden Code Richtlinien von dem Team aufgestellt, welche für das Team aber auch für die Firma gelten. Beispiele für Richtlinien sind Prinzipien wie „Dekomposition“ (Komplexe Probleme in kleine leichter zu lösende Probleme aufteilen) und auch „D.R.Y.“ (**D**on’t **R**epeat **Y**ourself).

Mithilfe dieser Richtlinien kann man bewerten wie die gegebene Code Qualität ist. Nach dieser ersten Bewertung kann man einen ersten Kostenvoranschlag an den Kunden senden. Je nach Code Qualität muss der PM mehr oder weniger Zeit für das Projekt einplanen. Mit dieser Methode kann der PM genug Zeit für das Projekt einplanen, um damit keine anderen Projekte zu gefährden.

Lektion 1	Teammitglieder, welche nicht mit dem Umgang von GitHub geschult sind, sollten vor Kundenprojekten einen Workshop besuchen.
Lektion 2	Kick-Off mit dem Kunden einplanen. Dort werden bekannte Probleme, vorgestellte Verbesserungen, komplette Neuentwicklungen und alle erforderlichen Workflows aufgezeigt.
Lektion 3	Wenn die Entwicklung an einem Vorgängerprojekt anknüpft, immer die Codequalität mithilfe der Richtlinien überprüfen und eine Zeiteinschätzung abgeben, um das Projekt aufwandstechnisch richtig einschätzen zu können.