Reflexion Lego II: Hafen

Freyschmidt, Henry Lewis
Hama, Zana Salih
Krasnovska, Paula
Krüger, Lucas
Prüger, Marvin Oliver
Seep, Tom-Malte
Zabel, Steven

5. November 2023

Version: 2
Verantwortliche Teammitglieder:
- Krüger, Lucas (Organisation)
- Freyschmidt, Henry Lewis (Anlegen der Dokumentenversion)
- Hama, Zana Salih (Anlegen der Dokumentenversion)
- Krasnovska, Paula (Anlegen der Dokumentenversion)
- Krüger, Lucas (Abgabe)

Anwesende während der Meetings:
- Freyschmidt, Henry Lewis
- Hama, Zana Salih
- Krasnovska, Paula
- Krüger, Lucas
- Prüger, Marvin Oliver
- Seep, Tom-Malte

(Online-) Beiträge zum Inhalt durch:
- Freyschmidt, Henry Lewis
- Hama, Zana Salih
- Krasnovska, Paula
- Krüger, Lucas
- Prüger, Marvin Oliver
- Seep, Tom-Malte
- Zabel, Steven

Korrektur gelesen durch:
- Hama, Zana Salih
- Krüger, Lucas
- Krasnovska, Paula

Inhaltsverzeichnis

1 Reflexion und Vergleich zum Workshop 2

Ähnlich zum ersten Lego-Workshop ließen sich die Herausforderungen abermals in die Themen Ressourcen, Aufgaben, Kommunikation, sowie Koordination separieren. Ins Thema **Ressourcen** gehen dabei Faktoren wie Ressourcenverteilung und Zugänglichkeit. Das Thema **Aufgaben** beinhaltet die Priorisierung von User Stories und resultierenden Anforderungen, zudem die Abschätzung von Arbeitsaufwand, das Verteilen der Anforderungen an Teams und innerhalb der Teams, sowie das Erfüllen der User Stories. **Kommunikation** umfasst den Austausch von Rahmenbedingungen zwischen dem Organisationsteam und dem Bauteam, die Einbindung des Stakeholders zur Präzisierung der Anforderungen, sowie Kontrolle der Projektvorstellung und Reaktion auf spontane Anforderungen des Stakeholders. Darüber hinaus befasst sich die **Koordination** mit der strukturellen Gliederung des gesamten Projektes und dem Einbinden agiler Praktiken.

Im Kontrast zum ersten Workshop wurde dieses Mal in Bezug auf Koordination einiges verbessert und angepasst. Da der ganze Ablauf des Projekts sich an den Ablauf des agilen Prozesses „Scrum“ halten sollte, hielt sich die anfängliche Planlosigkeit des ersten Workshops dieses Mal in Grenzen. So gab es schon nach nur wenigen Minuten klar herauskristallisierte Bau-Teams mit jeweiligem Scrum Master und Product Owner. Unmittelbar auf die Verteilung der Rollen folgend begann die strikte Einhaltung des durch Scrum vorgegebenen Zeitplans. Auch wenn so Struktur in den ganzen Prozess gebracht wurde, brachte es auch neue Komplikationen mit sich, wie beispielsweise das fehlende Wissen zu der Durchführung und Nutzung der einzelnen Meetings und agilen Praktiken.

Wie im ersten Workshop wurden in Bezug auf Aufgaben wieder alle erfüllt, doch aufgrund des fehlenden Grundlagenwissens über den Scrum-Prozess wurde die erste Sprintplanung nicht effektiv bis falsch angewendet. Die Product Owners haben die Verantwortung übernommen die Anforderungen kategorisiert an die jeweiligen Bau-Teams weiterzuleiten, jedoch ist die Kategorisierung zu grob ausgefallen. Sie wurden lediglich in Oberthemen sortiert, z. B. Kran und Wachturm. Es wäre notwendig gewesen die Anforderungen, die schon in Prioritäten sortiert waren, in sinnvolle Teilaufgaben aufzuteilen, diese dann in benötigte Aufwände zu gliedern – indem die agile Praktik Planning Poker implementiert wird – und die notwendige Menge an Arbeitskräfte zu berücksichtigen. Mit diesen Verbesserungen hätte es initial den Beginn des Realisierens vorschoben, jedoch würden spätere Komplikationen und somit Zeitverschwendung vorgebeugt werden. Hierbei lässt sich erkennen, dass sich die Aufgabenverteilung im zweiten Workshop kaum vom ersten Workshop unterscheidet, da im ersten die Aufgaben ebenfalls in große Gebiete eingeteilt wurden, aber keine Teilaufgaben entstanden. Weiterhin wurden die Aufgaben erneut wie im ersten Workshop nicht ausreichend nach ihrem Aufwand eingeteilt, sodass die Aufgabenverteilung im ersten als auch im zweiten Workshop ineffizient war.

Eine effiziente Aufgabeneinteilung hätte hierbei einige Konflikte verhindern können, wie beispielsweise eine uneindeutige Formulierung der Aufgabenstellung, welche ungewollte Freiheiten ergeben hat, wodurch nicht wenige Rückfragen an die Product Owner entstanden sind. Dadurch wurden nicht nur die Product Owner aus ihren Prozess/Gedanken herausgezogen, sondern die Bau-Teams mussten ihre Arbeit stoppen, um die gewünschte Umsetzung zu realisieren. Das Unterteilen der Anforderungen in verarbeitbare Teilaufgaben wird in die Bau-Teams verlagert, weshalb die Zeit der Arbeitskräfte ineffektiv genutzt wurde. Hier lassen sich Parallelen zum ersten Workshop erkennen, bei welchen die Aufgaben ebenfalls nicht konkret gestellt wurden und somit ungewünschte Resultate erzeugt wurden.

Sich der Vorbeugung von Problemen anschließend ist die korrekte Implementierung von Scrum, wie die sich den Sprints anschließende Retrospektive. Durch eine tiefgründige Unterteilung der Aufgaben und Einschätzungen des Arbeitsaufwandes wäre ein Vergleich zwischen den Erwartungen und der Realität ermöglicht und somit unerwartete Probleme erkennbar geworden. Im Workshop wurde diese Phase vollkommen verdrängt, indem triviale Probleme angesprochen werden konnten, weshalb eine nicht zufriedenstellende Evaluation stattfand. Ebendeshalb konnte die nächste Sprintplanung keine Verbesserungen integrieren.

In Bezug auf Kommunikation hat sich im zweiten Workshop einiges geändert, wie die Einbindung des Stakeholders, welche durch Scrum zwar anspruchsvoll umzusetzen war, da dieser lediglich einmal während eines Meetings zur Verfügung stand, bei welchem die aktuelle Entwicklung jedes Teams vorgestellt wurde. Aber hierbei war die Einbindung deutlich besser verglichen zum ersten Workshop, da innerhalb dessen der Stakeholder wenig einbezogen wurde. Dadurch konnte der Stakeholder den Product Ownern kenntlich machen, was überarbeitet und ergänzt werden musste. Somit war das Endprodukts im Allgemeinen zufriedenstellender und es konnten Fehler vermieden werden, welche sonst zu spät bemerkt worden wären. Dies verbesserte auch die ästhetische Funktionalität der einzelnen Baukörper, welche im ersten Workshop teilweise nicht erkannt werden konnte. Ein weiterer Fortschritt innerhalb der Kommunikation wäre die Regelung des Maßstabs. Diesbezüglich wurde im ersten Workshop der Maßstab geringfügig besprochen, was zu einem uneinheitlichen Baustil innerhalb der Teams führte. Im Kontrast dazu wurde der Maßstab im zweiten Workshop am Anfang anhand eines Objektes festgelegt, weswegen das Endprodukt weitaus normierter war.

Zuallerletzt war die Ressourcenverteilung vergleichsweise effizienter. Es entstanden keine Engpässe bei den Ressourcen und es wurde darauf, geachtet Ressourcen zu beanspruchen, welche das eigene Team wirklich benötigt, statt jede mögliche Ressource zu nehmen. Durch diese Veränderung kam es nicht mehr zu einer Ressourcenknappheit wie im ersten Workshop, und der Zugriff auf Ressourcen verlief effizienter.

2 Eigener Prozessentwurf

2.1 Agile Praktiken

In unserem eigenen Prozess ist es uns wichtig **testgetriebene Entwicklung** zu betreiben. Zum einen wird dabei redundanter Code vermieden, da man sich auf das konzeptuell Wesentliche konzentriert und andererseits werden auch Fehler vermieden, da man nur so viel programmiert, bis alle Tests fehlerfrei durchlaufen wurden und es somit keine ungetesteten Komponenten beinhaltet. Dies erleichtert unter anderem das darauffolgende Refactoring.

Ebenso wichtig ist die Praktik des **Pair Programmings**, die wir auf ausgewählte Situationen anwenden werden. Auf elementarer Ebene fördert es das Arbeitsklima, da man gemeinsam über Ansätze nachdenken kann und einander fördert, was auch positive Einflüsse innerhalb der Gruppenkommunikation induzieren kann. Andererseits werden auch durch das Konzept Fehler früher erkannt und es steigert grundsätzlich die Qualität des produzierten Codes, da der Begutachter die Korrektheit des Codes im Blick behält und simultan auch über nachhaltige Verbesserungen am Design reflektiert.

So wichtig wie die bisher erwähnten Praktiken sind, ohne **Coding Standards** wäre die Anschaulichkeit des Codes nicht garantiert und würde beim Nachvollziehen durch das Team redundante Zeitkosten verursachen. Um dies zu vermeiden und die Einheitlichkeit des Codes zu bewahren, führen wir folgende Regeln ein, die womöglich erweitert werden im Verlaufe des Projekts:

---

1. Schlüssige Kommentare:
   - falls Abkürzungen ihre Verwendung finden, sollen diese eindeutig gekennzeichnet werden, beispielsweise durch Ausformulierung innerhalb der ersten Verwendung
   - das Charakteristikum einer Funktion sollte grob umrissen werden
   - Namenskürzel zur Kennzeichnung des Programmierers einer Funktion zuzüglich einer klaren Kennzeichnung einer Revision (falls gemacht)
2. Verwendung intuitiver Parameternamen ohne Raum für Ambiguität
3. kompakte Code-Formatierung
   - bei Kontrollstrukturen mit einem Befehl lassen wir die geschweiften Klammern weg:

```
1 public static something(){
2     if(true) doSomething();
3 }
```

   - sonst:

```
1 public static something(){
2     if(true){
3         doSomething();
4         doSomethingDifferent();
5     }
6 }
```

4. Funktionen so kompakt wie möglich halten, ansonsten: Unterteilung in Unterfunktionen zur Bewahrung der Übersichtlichkeit

---

Zuletzt nutzen wir das Konzept des **Backlogs**. Hierdurch können wir eine strukturierte Arbeitsweise ermöglichen und haben zu jedem Zeitpunkt die wichtigsten Aufgaben im Überblick. Inwiefern dieses angelegt wird, erübrigt sich im weiteren Verlauf des Projekts.

2.2 Rollenverteilung

- Product Owner
  - ➢ Zabel, Steven
- Scrum Master
  - ➢ Hama, Zana Salih
- weiteres Team
  - ➢ Freyschmidt, Henry Lewis
  - ➢ Krasnovska, Paula
  - ➢ Krüger, Lucas
  - ➢ Prüger, Marvin Oliver
  - ➢ Seep, Tom-Malte

2.3 Zeiteinteilung

Unsere Sprints werden wir mit einem Planning Game/Poker umsetzen. Hierdurch können wir die Zeitkostenaufwände der User Stories durch Einbezug der Entwickler ermitteln und bestimmen, welche dieser im aktuellen Sprint bearbeitet werden sollen.

Des Weiteren planen wir wöchentliche Meetings im Stil von Daily Meetings, um frühzeitig Probleme, die die Arbeitsfortsetzung blockieren, zu klären.

Nach dem Ende jedes Sprints werden wir eine allumfassende Retrospektive initiieren, um Feedback über die Arbeitsweise zu erhalten und rückwirkend Verbesserungsvorschläge für die nächste Sprint-Phase zu identifizieren. Außerdem sollten nach jeder Iterationsphase fertige Funktionen bereitstehen nach dem Schema des vertikalen Prototyps.

(!) *Disclaimer: Alles bisher aufgeführte entspricht nur einem Grobentwurf. Dieser könnte sich im Verlaufe des Projekts verändern.*