

1. Übungszettel in Software Engineering

Tatsächlich benötigte Zeit

Übersicht

Klasse	Schätzung [h]	Realität [h]	Differenz [h]	Realität/Schätzung
GUIConsole	6,25	13	6,75	2,08
DotsNBoxesEngine	7,5	10	2,5	1,33
Player	0,5	0,5	0	1
Summe	14,25	23,5	9,25	1,62

Wir benötigten ca. 62% mehr Zeit als prognostiziert. Hauptsächlich lag das daran, dass wir unsere Fähigkeiten über- und Detailfragen unterschätzt haben.

Im Laufe der Bearbeitung haben wir gelernt, die Programmierung von einzelnen Methoden besser aufzuteilen, indem wir Schnittstellen definieren und die Ergebnisse mit Git schnell zusammenzuführen. Außerdem hilfreich waren Diskussionen und das gegenseitige Überprüfen von Code.

Durch diese Lernfortschritte, die Erfahrung und die Erfassung der Diskrepanz, hoffen wir, Zeitabschätzungen in Zukunft realistischer geben zu können.

Klasse GUIConsole

Aufgabe	Schätzung [h]	Realität [h]	Begründung
Eingabe	0,75	4,5	Zunächst entstanden viele Redundanzen, sodass es schwierig war, Optimierungen (an vielen Stellen) umzusetzen. Außerdem machte die Stream-Natur des Inputs Probleme bei der Prüfung (Öffnen und Schließen des Inputs sorgte für Endlosschleife.
Darstellung Spielfeld	1	1	
Anzeige	0,5	1,5	Doppeltes Hochzählen um aktuellen Spieler zu ermitteln führte zu eine NullPointerException, die nicht direkt gefunden wurde.
Tests	2,5	3	Schnittstellen zwischen Klassen waren zunächst zu undeutlich abgestimmt. Dadurch war es schwierig brauchbare Tests zu schreiben. In Folge wurde viel manuell getestet, was viel Zeit in Anspruch nahm.
Fehlerkorrekturen	1,5	3	Es gab unerwartet viel zu optimieren.
Summe	6,25	13	Abweichung: 6,75h

Klasse DotsNBoxesEngine

Aufgabe	Schätzung [h]	Realität [h]	Begründung
Datenstruktur Karte	1	1	
Zug durchführen	2	4	Es hat viel Zeit gekostet zu ermitteln, welcher Spieler als nächstes an der Reihe ist.
Punkteverwaltung	0,5	0,5	
Tests	2,5	3	siehe Klasse GUIConsole
Fehlerkorrekturen	1,5	1,5	
Summe	7,5	10	Abweichung: 2,5h

Klasse Player

Genau wie erwartet. Wenig Aufwand.

Teilaufgaben und Schätzungen

Aufgabe	Begründung/Kommentar	Schätzung [h]
Klasse GUIConsole	Starten, Einstellen, Spielen	SUMME: 6,25
- Eingabe - - Spieleranzahl - - Spielfelddimensionen - - Spielernamen - - Zug (Spieler x setzt Wand y)	Einarbeitung in Eingabemethoden auf der Konsole, gemäß Erfahrung geringe Komplexität, schnell umsetzbar	0,75
- Darstellung Spielfeld	Darstellung soll Datenstruktur der Karte interpretieren, sodass eine Anpassung der Darstellung möglich ist, ohne die Datenstruktur grundsätzlich zu ändern. Darstellung muss sich bei großen Feldern anpassen.	1
- Anzeige - - Punkte - - Gewinner - - Zugzahl	Anzeige zum richtigen Zeitpunkt, sonst keine besondere Komplexität. Inhalte werden über Methoden der jeweiligen Klassen gut abrufbar sein.	0,5
- Tests schreiben	Schwierig, da möglichst viele Fälle (umfangreich) abgedeckt sein müssen und Übung fehlt. Erfordert viel Kreativität.	2,5
- Fehlerkorrekturen & Optimierungen	Etwas geringere Komplexität als die Tests. Nachdem die Tests laufen, sollten nicht mehr viele Korrekturen/Optimierungen nötig sein	1,5

Klasse DotsNBoxesEngine	Einhaltung der Regeln, Verwaltung des Spielstandes, Punkteberechnung	SUMME: 7,5
- Datenstruktur Karte (2D-Array) mit Initialisierung	Grundsätzliche geringe Komplexität, aber durch Zweidimensionalität wenig intuitiv	1
- Zug durchführen (boolean: true, wenn Zug erfolgreich) - - Ist der Spieler an der Reihe? - - Ist Zug gültig? - - Ist Käsekästchen voll? - - Ist das Spiel beendet? - - Wer ist als nächstes an der Reihe?	Höchste Komplexität, da hier die komplette Spiellogik liegt (die zuerst verstanden sein muss) und viele Hilfsmethoden geschrieben werden müssen.	2
- Punkteverwaltung	Geringe Komplexität, da es sich gut über getter und setter der Klasse Player umsetzen lässt	0,5
- Tests schreiben	Wie in Klasse GUIConsole	2,5
- Fehlerkorrekturen	Wie in Klasse GUIConsole	1,5

Klasse Player	Verwaltung Name, Punktestand	SUMME: 0,5
- Name setzen - Name holen - Punkte setzen - Punkte holen - Punkte erhöhen um Wert x - Tests schreiben - Fehlerkorrekturen	Im Grunde nur getter- und setter-Methoden. Daher sehr geringe Komplexität	
GESAMTZEIT		14,25