

# Quality Measurements

King of Jawa

Pascal Bürklin, Isabel Geissmann, Jannik Jaberg, Nikolai Rutz

21. Mai 2018



# KING OF JAWA

## Inhaltsverzeichnis

<b>1</b>	<b>Richtigkeit</b>	<b>2</b>
<b>2</b>	<b>Angemessenheit</b>	<b>2</b>
<b>3</b>	<b>Sicherheit</b>	<b>3</b>
<b>4</b>	<b>Modifizierbarkeit</b>	<b>3</b>
<b>5</b>	<b>Analysierbarkeit</b>	<b>4</b>
<b>6</b>	<b>Rahmenbedingungen</b>	<b>4</b>
6.1	Bugs und Bug Tracker . . . . .	4
6.2	Unit Tests . . . . .	6
6.3	Playtesting . . . . .	7
6.4	CI . . . . .	8
6.5	Restliche Rahmenbedingungen . . . . .	8

## 1 Richtigkeit

**Ziel:** Um die Richtigkeit von King of Jawa zu garantieren setzen wir uns als Ziel keine ‘ungenauen’ Berechnungen zu machen. Weiter sollen Rechenschritte in ihre eigenen Methoden ausgelagert werden um zu vermeiden dass lange Berechnungen den Aufnahmefähigkeiten eines Developers zu sprengen. Messbar machen wir dies mittels dem Mantis-Bugtracker, welchen Pascal auf seiner Website aufgesetzt hat. Uns ist es wichtig dass wir jeden Bug bis zu seinem gesetzten Due-Date fixen oder zumindest bearbeiten. Bei gravierenden Bugs wollen wir höchstens 2 Tage warten bis zum Fix.

**Resultat:** Ungenaue Berechnungen wurden alle behoben und die Verwendung des Bugtrackers wird im Punkt Bugs und Bug Tracker beschrieben.

**Fazit:** PASSED

## 2 Angemessenheit

**Ziel:** Weiter ist die Angemessenheit ein wichtiger Punkt. Da manchmal eine Abschätzung, ob ein Algorithmus sinnvoll oder nicht ist, ist für uns zu diesem Zeitpunkt noch schwierig, wollen wir dennoch dies als Nebenziel setzen, die Implementierten Algorithmen und/oder Datenstrukturen zu reflektieren und zu entscheiden, ob diese im entsprechenden Konsens angemessen sind. Was jedoch sinnvoll, machbar und konsequent messbar ist, ist Spaghetti-Code zu vermeiden. Deshalb haben wir uns dafür entschieden Methoden mit über 50 Zeilen in weitere Untermethoden abzukapseln. Dies sollte Verständlichkeit und Übersichtlichkeit stark fördern. Die Ausnahme bestätigt die Regel: Klar ist dies nicht zu 100% umsetzbar - es gibt immer Ausnahmen wie beispielsweise run() Methoden, welche trotz über 50 Zeilen immer noch übersichtlich sein können. Deshalb setzen

wir uns als Mass ungefähr 90% der Methoden 50 Zeilen nicht überschreiten dürfen.

**Resultat:** Bezüglich der Angemessenheit haben wir uns richtig ins Zeug gelegt. Spaghetti-Code haben wir kaum mehr, da Pascal darauf bestanden hat nach jedem Milestone die Codebase zu refactoren und optimieren. Auch bei den Algorithmen und Datenstrukturen haben wir uns einiges überlegt. Anstelle von Listen haben wir in gewissen Fällen Hashtables oder Maps benutzt. Bei der Länge der Methoden haben wir gemerkt, dass dies nicht immer durchsetzbar ist. Bei den Ressourcengebäudeleveln beispielsweise hat jede Klasse eine Level Methode, welche über 100 Zeilen lang ist, wie auch einige run() Methoden, die über 50 Zeilen lang sind.

**Fazit: FAILED**

### 3 Sicherheit

**Ziel:** Um Spass am Spiel zu garantieren ist die Sicherheit ein weiterer Kernpunkt den QAs. Messbar ist dies nicht wirklich, jedoch probieren wir allfällige Sicherheitslücken im Code, die durch Injection exploited werden können zu minimieren. Beispielsweise wird bei einer removeRequest für eine Lobby der Socket des Owners und der Socket des Clients, der die Request gesendet hat, verglichen um sicherzugehen, dass nicht ein anderer Client Lobbies schliessen kann.

**Resultat:** Wie im Ziel erwähnt wollen wir möglichst viele Sicherheitslücken schliessen. Dies haben wir getan, sobald uns eine Lücke auffiel, schlossen wir sie. Jedoch sind sicherlich nicht alle Lücken geschlossen und wir können auch keine Garantie geben, dass unser Spiel nicht hackable oder cheatable ist. Dennoch sind wir sehr zuversichtlich, dass wir Spielspass ohne Sicherheitslücken liefern können.

**Fazit: PASSED**

### 4 Modifizierbarkeit

**Ziel:** Da die Spielidee von King of Jawa beliebig erweiterbar ist, setzen wir uns als weiteres Kernmerkmal die Modifizierbarkeit. Dem haben wir mit unserem Kommunikationsprotokoll bereits einen fundamentalen Grundbaustein gesetzt. Mit Hilfe eines entity-component-system erreichen wir dies. Dies zu verwirklichen ist es ziemlich einfach, nämlich wir haben ein Entity-Component-System oder nicht. Wenn wir diese Bedingungen alle berücksichtigen, sollten Änderungen an Gebäuden oder hinzugefügen neuere Gebäude sehr einfach sein.

**Resultat:** Das System wurde im Verlauf des Projekts drei Mal refactored und verbessert. Mittlerweile ist es ziemlich einfach neue Gebäude zu implementieren und bestehenden Gebäuden Attribute zu geben. Dafür hat sich Nikolai ein Balancing-System überlegt und implementiert, dies unterstützt das Entity-Component-System extrem. So einfach wie wir uns dies vorgestellt haben ist es jedoch nicht. Man kann nicht einfach nur eine neue Klasse erstellen, sondern muss noch in vier weiteren Klassen Anpassungen machen.

**Fazit: FAILED**

## 5 Analysierbarkeit

**Ziel:** Für die Analyse implementieren wir einen Bugtracker, wie bereits im Punkt Richtigkeit beschrieben und einen Debugmode, für Client und Server.

**Resultat:** Mithilfe unseres Debugmodes und dem Bugtracker haben wir einen schnellen Weg geschaffen, Fehler in unserem Projekt zu finden und zu beheben. Auch die CI half uns oft Nullpointer zu erkennen und finden. Mittlerweile haben wir sogar ein ErrorLog File welches hilft Fehler zu analysieren und auszumertzen. Mehr dazu bei den Punkten Bugs und Bug Tracker und CI.

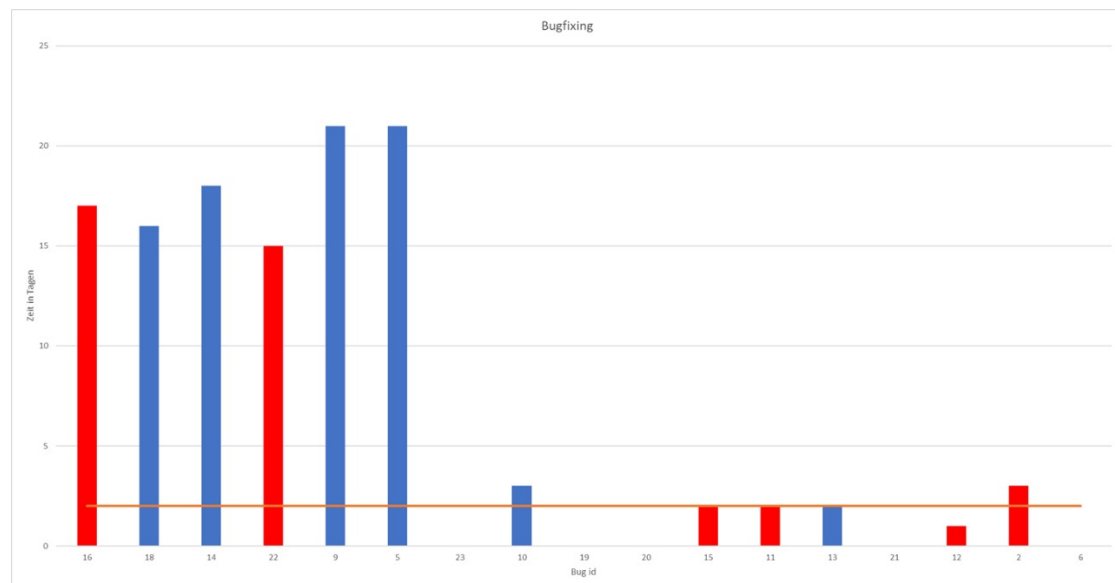
**Fazit:** PASSED

## 6 Rahmenbedingungen

### 6.1 Bugs und Bug Tracker

**Ziel:** Alle Bugs die auftreten werden in unserem Bugtracker festgehalten und gemäss Beschreibung im Ziel des Punkt Richtigkeit bearbeitet.

**Resultat:** Wie bereits im QA beschrieben haben wir Mantis als Bug Tracker gewählt und diesen auch immer benutzt. Die im Milestone 3 angegebenen Fristen von 2 Tagen haben wir jedoch gesprengt. Es war uns nicht möglich jeden Bug so zu behandeln. Deshalb entschieden wir uns um und setzten spezifische Due Dates für jeden Bug. Gravierende Bugs bearbeiteten wir immer innerhalb von 2 Tagen aber andere auch noch kurz vor knapp bis zum jeweiligen Meilenstein.



Hier sieht man alle getrackten Bugs in Abhängigkeit zur Zeit in der sie behoben wurden. Rot sind hoch priorisierte Bugs mit Due Date von 2 Tagen. Die Blauen sind irgendwelche Bugs mit Due Date bis zum nächsten Meilenstein oder noch später. Wenn

kein Balken vorhanden ist, wurde der Bug in unter 24h behoben. Also sind drei rote Bugs über ihr Due Date geschossen. Bei den Bugs mit der ID 16 und 22 war der Grund ganz simpel. Wir haben sie nicht geschlossen. Bug 2 ging uns effektiv durch die Lappen und wir versäumten sein Due Date.

Einträge anzeigen 1-18 / 18							
<a href="#">Berichte drucken</a> <a href="#">CSV-Export</a> <a href="#">Excel-Export</a>							
	P	ID	Kategorie	Auswirkung	Status	Aktualisiert	Zusammenfassung
<input type="checkbox"/>	↑	0000023	Engine	kleinerer Fehler	erledigt (Hidin)	2018-05-08	Nullpointer when building outside map
<input type="checkbox"/>	↑	0000022	Gameplay	kleinerer Fehler	zugewiesen (Hidin)	2018-05-07	Highscore wont get saved when people give up
<input type="checkbox"/>	→	0000010	UI	kleinerer Fehler	erledigt (Hidin)	2018-05-06	Der User-State wird nicht angezeigt.
<input type="checkbox"/>	→	0000019	UI	Trivial	erledigt (Hidin)	2018-05-06	Player bleibt in Playerliste im UI nachdem er geschlossen wurde.
<input type="checkbox"/>	→	0000020	1 1 Gameplay	kleinerer Fehler	erledigt (Hidin)	2018-05-06	Spectator gewinnt
<input type="checkbox"/>	⚠	0000015	Gameplay	Feature-Wunsch	erledigt (Hidin)	2018-05-06	Victory
<input type="checkbox"/>	→	0000009	Engine	Feature-Wunsch	zugewiesen (Hidin)	2018-05-06	Ships
<input type="checkbox"/>	↑	0000011	Gameplay	kleinerer Fehler	erledigt (Hidin)	2018-05-06	BuildingLimits werden nicht Ordnungsgemäss verwaltet.
<input type="checkbox"/>	↑	0000013	Gameplay	Feature-Wunsch	erledigt (Hidin)	2018-05-06	Buildings sollten Levelbar sein!
<input type="checkbox"/>	↑	0000017	1 Gameplay	Absturz	zugewiesen (Hidin)	2018-05-06	Absturz bei versuch Buildung zu setzen
<input type="checkbox"/>	→	0000021	UI	kleinerer Fehler	erledigt (Hidin)	2018-05-06	Inselfarbe nach Lobby leave
<input type="checkbox"/>	✓	0000018	Gameplay	Unschönheit	neu	2018-05-06	Highscore wird nicht nach Punkten geordnet.
<input type="checkbox"/>	↑	0000016	UI	Absturz	erledigt (cpt.underground)	2018-05-05	Absturz bei Game start
<input type="checkbox"/>	↑	0000012	Connection	Unschönheit	erledigt (Hidin)	2018-05-05	Ping wird nicht Ordnungsgemäss verarbeitet
<input type="checkbox"/>	→	0000014	UI	Feature-Wunsch	neu	2018-05-04	Settings-tab im Mainmenu.
<input type="checkbox"/>	↑	0000002	1 Gameplay	Feature-Wunsch	erledigt (Hidin)	2018-05-04	Lobbyanzeige: State für Spieler
<input type="checkbox"/>	⚠	0000006	2 Connection	Absturz	erledigt (isabel)	2018-05-01	Chat crashed Programm stürzt ab
<input type="checkbox"/>	✓	0000005	Gameplay	Feature-Wunsch	zugewiesen (cpt.underground)	2018-05-01	Tile-Rotation
<input type="checkbox"/> Alle auswählen         Verschieben <input type="button" value="OK"/>							

In diesem Bild sieht man alle Bugs die wir bis jetzt getrackt haben. An diesem Screenshot sieht man gut, dass es noch nicht bearbeitete Bugs gibt, wie beispielsweise den Feature-Wunsch für ein Settings Button im Menü mit der ID 0000014. Dieser ist zugewiesen und hat das Due Date bis zum Meilenstein 5, was man im nächsten Bild gut sieht.

Eintragsdetails ansehen					
<a href="#">Erinnerung senden</a> <a href="#">Zu Notizen springen</a> <a href="#">Zu Historie springen</a>					
ID	Projekt	Kategorie	Sichtbarkeit	Meldungsdatum	Zuletzt aktualisiert
0000014	King of Jawa	UI	privat	2018-05-04 15:26	2018-05-04 15:26
Reporter	Hidin	Bearbeitung durch			
Priorität	normal	Auswirkung	Feature-Wunsch	Reproduzierbar	N/A
Status	neu	Lösung	offen		
Produktversion	MS3				
Zielversion	MS5 SNAPSHOT	Behoben in Version			
Zusammenfassung	0000014: Settings-tab im Mainmenu.				
Beschreibung	Es sollte dem User möglich sein über einen Settings,tab keybindings und den namen zu ändern				
Tags	Keine Tags zugeordnet.				
Tags zuordnen	(Trenne durch „“) <input type="text"/> Existierende Tags <input type="button" value="Zuordnen"/>				
<input type="button" value="Bearbeiten"/> <input type="button" value="Zuordnen zu"/> <input type="button" value="[Ich selbst]"/> <input type="button" value="Status wechseln zu:"/> <input type="button" value="Rückmeldung"/> <input type="button" value="Beobachten"/> <input type="button" value="Klonen"/> <input type="button" value="Schließen"/> <input type="button" value="Verschieben"/> <input type="button" value="Löschen"/>					

Wie bereits gesagt könnten wir die Fristen nach dem Entscheid gut einhalten, was wir jedoch garnicht im Griff hatten war die Schliessung von Bugs (im nächsten Bild zusehen).

Eintragsdetails ansehen
Erinnerung senden Zu Notizen springen Zu Historie springen
<< >>

ID	Projekt	Kategorie	Sichtbarkeit	Meldungsdatum	Zuletzt aktualisiert
0000017	King of Jawa	Gameplay	öffentlich	2018-05-05 11:20	2018-05-06 21:55

Reporter	cpt.underground	Bearbeitung durch	Hidden		
Priorität	dringend	Auswirkung	Absturz	Reproduzierbar	nicht getestet
Status	<span>zugewiesen</span>	Lösung	offen		
Produktversion	MS4 SNAPSHOT				
Zielversion	MS4 SNAPSHOT	Behoben in Version			

**Zusammenfassung** 0000017: Absturz bei versuch Building zu setzen

**Beschreibung**

Game start geht base auch dann aber keine Möglichkeit ein anderes Gebäude zu setzen

```
Exception in thread "Thread-1" java.lang.NullPointerException
    at server.game.GameContainer.buildBuilding(GameContainer.java:122)
    at server.game.GameContainer.handleBuildRequest(GameContainer.java:65)
    at server.game.GameManager.run(GameManager.java:52)
    at shared.net.protocol.ProtocolManager.call(ProtocolManager.java:43)
    at shared.net.protocol.ProtocolManagement.receivePackage(ProtocolManagement.java:6)
    at shared.net.NetworkThread.run(NetworkThread.java:48)
    at java.lang.Thread.run(Thread.java:748)
```

**Tags** Keine Tags zugeordnet.

**Tags zuordnen** (Trenne durch „“)  Existierende Tags Zuordnen

Bearbeiten Zuordnen zu [Ich selbst] Status wechseln zu: neu Beobachten Klonen Schließen Verschieben Löschen

Hier handelt es sich um eine hoch priorisierten Bug, der zu einem Absturz führte. Der Bug war innert 2 Stunden behoben deutlich vor der angegebenen Zielversion für MS4. Er war aber offen bis zum 17. Mai 2018. Fazit dazu, wir hielten die Dates relativ gut ein doch haben das Bug Tracking Tool viel zu wenig benutzt. Viele Bugs kamen nicht einmal bis in den Bugtracker und andere wurden nur ineffizient eingetragen.

**Fazit: just PASSED**


## 6.2 Unit Tests

**Ziel:** Der Code wird fortlaufend mit Unit Tests überprüft. Im Konzept einigten wir uns darauf ,dass jede Klasse die mehr als 100 Zeilen hat, mit Unit Tests überprüft werden muss. Alle Methoden dieser Klassen müssen mit Unit Tests abgedeckt werden. Nach dem Tutorat wurde uns jedoch klar, dass dies nicht umsetzbar ist. Daraufhin überlegten wir uns eine neue Messung. Entschieden haben wir uns dafür, dass wir wichtige Klassen testen. Dazu gehören:



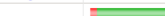




- Chain
- Serialization
- Package
- PackageManager












- User
- Session

**Resultat:** Wir haben Jacoco zum Projekt hinzugefügt, um die Code Coverage zu betrachten. Momentan (Stand: 17. Mai 2018) liegt die Coverage der wichtigen Klassen wie folgt.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Session		100 %		n/a	0	1	0	1	0	1	0	1
Total	0 of 3	100 %	0 of 0	n/a	0	1	0	1	0	1	0	1

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
User		100 %		n/a	0	1	0	1	0	1	0	1
Total	0 of 3	100 %	0 of 0	n/a	0	1	0	1	0	1	0	1

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ProtocolManager		32 %		33 %	5	9	12	21	3	6	0	1
Package		94 %		67 %	13	45	8	90	0	25	0	1
ProtocolManagement		42 %		n/a	1	2	2	3	1	2	0	1
PackageManager		100 %		75 %	3	12	0	28	0	6	0	1
Total	81 of 542	85 %	20 of 58	65 %	22	68	22	142	4	39	0	4

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Teestream		0 %		n/a	6	6	17	17	6	6	1	1
ConsoleLog		92 %		50 %	9	13	11	65	3	7	0	1
Serialization		97 %		95 %	5	67	2	142	0	12	0	1
Pair		0 %		n/a	3	3	6	6	3	3	1	1
Point		67 %		n/a	1	6	4	14	1	6	0	1
Chain		100 %		100 %	0	46	0	108	0	23	0	1
Area		100 %		62 %	3	15	0	25	0	11	0	1
Total	111 of 1'716	93 %	14 of 176	92 %	27	156	40	377	13	68	2	7

Hier auch noch eine Übersicht zur Overall Coverage Stand: 17. Mai 2018

shared.user	100 %			n/a	0	1	0	1	0	1	0	1
shared.net.session	100 %			n/a	0	1	0	1	0	1	0	1
shared.game.map	100 %			n/a	0	1	0	1	0	1	0	1
Total	20'112 of 27'305	26 %	1'353 of 1'790	24 %	1'463	1'990	3'444	5'130	692	1'071	77	148

Bezüglich den UnitTests muss man sagen es war für alle von uns neu und wir hatten mühe damit. Meistens haben wir die UnitTests erst ganz am Schluss eines Features programmiert. Oder auch wurden Klassen die seit Milestone 2 stehen erst auf Milestone 4 getestet. Dies könnte man noch stark verbessern oder sogar mit TDD durchführen.

**Fazit: just PASSED**

### 6.3 Playtesting

**Ziel:** King of Jawa wird durch eine Testgruppe von 10 Personen getestet. Dafür wird ein Debug Modus implementiert. Dies gibt uns die Möglichkeit die Fehler einfacher nachzustellen um sie danach zu beheben.

**Resultat:** Unser Playtesting haben wir uns etwas anders vorgestellt. Wir haben extra einen Download zur Verfügung gestellt und einen Server auf den man sich mit dem Client

verbinden kann. Doch ist uns aufgefallen, dass die Mitstudierenden alle eigene Probleme haben mit ihrem Projekt und die Kollegen von Pascal sich auch nicht wirklich darum gerissen haben unser Spiel zutesten. Deshalb blieb die meiste Arbeit des testing bei uns hängen.

**Fazit: FAILED**

## 6.4 CI

**Ziel:** Die Benützung von CI.

**Resultat:** Das CI Tool in Git hat uns geholfen Fehler zu entdenken, die bei unseren Maschinen nicht aufgetreten sind. Bemerkt haben wir dies durch Marco. Bei ihm lief das buildscript bei der Abgabe von Milestone 4 nicht durch. Mithilfe der CI haben wir den Fehler finden und beheben können.

**Fazit: PASSED**

## 6.5 Restliche Rahmenbedingungen

- JavaDoc  
JavaDocs haben wir immer geschrieben und gebildet. Punkte gab es dafür auch immer, deshalb nehmen wir an, dass dies Genügend war. Für die Zukunft sollten alle von uns immer direkt nach der Implementation eines Features die JavaDocs schreiben und nicht erst am Schluss. Weiter könnten die Kommentare etwas ausführlicher sein.

**Fazit: PASSED**

- Style Conventions  
Dies einzuhalten war nicht besonders schwer, da IntelliJ einem unter die Arme greift.

**Fazit: PASSED**

- Name Conventions  
Name Conventions haben wir, in Anbetracht ein paar kleinen Ausreissern, immer angepasst.

**Fazit: PASSED**

- Log4J  
Log4J haben wir aktiv benutzt, um zum Beispiel ein ErrorLog File zugenerieren, was die analyse von Fehlern, Bugs und Exceptions enorm vereinfachte.

**Fazit: PASSED**