Erweiterung des Qualitätsanforderungskonzepts – Settlers of Asgard

Einleitung

Seit der ersten Version des Qualitätsanforderungskonzepts für *Settlers of Asgard* wurden wesentliche Erweiterungen und Verbesserungen am Spiel vorgenommen. Dieses Dokument ergänzt das ursprüngliche Konzept und bewertet die Qualitätssicherungsmaßnahmen im Zeitverlauf. Ziel ist es, den Fortschritt zu dokumentieren, Schwachstellen zu identifizieren, Optimierungspotenziale aufzuzeigen und die Nachhaltigkeit der QA-Strategie sicherzustellen.

1. Ziele der Erweiterung

Folgende übergeordnete Ziele wurden im Rahmen der Weiterentwicklung des Spiels definiert:

- **Erweiterung der Spielmechaniken** (z.B. Einführung neuer Entities mit neuen Funktionen)
- **Systematisches Bugfixing** auf Grundlage von automatischer Fehlerprotokollierung zur Steigerung der Stabilität und Reduktion kritischer Spielfehler.
- **Verbesserung der Netzwerkarchitektur** zur Reduktion von Latenz und Verbindungsabbrüchen
- **Verbesserung der Code-Qualität** durch ausführliche Dokumentation und Verringerung ungenutzter Code-Stücke
- **Optimierung der Benutzeroberfläche** für verschiedene Endgeräte, sowie zur Minimierung der Speicherauslastung (Optimiertes Laden der Bilder)
- **Steigerung der Sicherheit** gegen Exploits und Manipulation
- **Skalierbarkeit des Systems** für eine wachsende Nutzerbasis

2. Methodik

Zur Überprüfung der Qualitätsziele wurden folgende QA-Maßnahmen über den Projektzeitraum systematisch angewendet:

- **Regelmäßige Regressionstests** aller Spielmechaniken nach jedem Major-Update
- **Regelmäßiges Smoke Testing und Exploratives Testen** nach jedem Hinzufügen neuer Funktionen
- **Feedback-Auswertung von Nutzern** durch zugewiesene Tester
- **Architekturrichtlinien zur Codekomplexität**, einschließlich der Begrenzung der Codezeilen pro Klasse, um die Lesbarkeit, Testbarkeit und Wartbarkeit des Codes langfristig sicherzustellen.
- **Regelmäßige Analysen** des Codes bezüglich des Code-Coverage und der Code-Dokumentation
- **Regelmäßige Sicherheitstests** zur Aufdeckung möglicher Schwachstellen

3. Ergebnisse

Die angewandten Maßnahmen führten zu folgenden messbaren Ergebnissen:

- **Spielabstürze** wurden aufgegriffen und behoben, sodass das Spiel verlässlich läuft
- **Bugs** wurden frühzeitig erkannt und effizient behoben, wodurch die Spielbarkeit des Games garantiert bleibt
- **Regressionen** wurden sofort entdeckt und behoben, wodurch größere Bugs und Probleme verhindert wurden
- **Benutzeroberfläche-bedingte Verzögerungen** wurden behoben
- **Begrenzung des Klassenumfangs** steigerte die strukturelle Klarheit und erleichterte Wartung und Testbarkeit.
- **Code-Coverage sowie Code-Dokumentation** wurden gesteigert und gewährleisten gute Code-Qualität
- **Sicherheit** wurde verbessert, Exploits und Manipulation werden verhindert

4. QA-Analyse im Zeitverlauf

Code-Coverage

Report	Total Instructions	Missed Instructions	Coverage Instructions (%)	Total Branches (Bedingungen/ Schleifen)	Missed Branches	Coverage Branches (%)	Complexity	Missed Complexity	Total Lines	Missed Lines	Total Methods	Missed Methods	Total Classes	Missed Classes
MS 3	11607	11427	1	863	853	1	1374	1359	3117	3076	905	893	222	219
MS 4	33628	28056	16	2791	2379	14	3099	2627	7767	6518	1649	1344	212	162
MS 5	43181	36838	14	3475	3113	10	3812	3355	9919	8555	2019	1695	239	188

Die Analyse zum Code-Coverage zeigt folgende Verbesserungen und Kritische Punkte:

- **Stetiges Wachstum des Codes** von MS 3 zu MS 5 ist deutlich zu erkennen in der Gesamtzahl an Instruktionen, Branches, Methoden und Klassen
- **Verbesserte Code-Abdeckung** wird besonders zwischen MS 3 und MS 4 sichtbar, wobei die Abdeckung bei Instruktionen von 1% auf 16%, und bei Branches von 1% auf 14% sprang. Einen ähnlichen Trend zeigt die Abdeckung von Zeilen und Methoden, mit einem großen Schub zwischen MS 3 und MS 4.
- **Verbesserung der Klassen** ist deutlich an der Anzahl der abgedeckten Klassen zu sehen, wobei die absolute Anzahl der nicht abgedeckten Klassen zu MS 5 hin leicht anstieg. Dies soll bis zur Deadline hin noch beachtet und verbessert werden.
- **Komplexität** nahm mit dem Codeumfang zu, dennoch blieb der Anteil der nicht abgedeckten Komplexität stabil, mit leichter Verbesserung. Dies konnte durch regelmäßige Analysen gewährleistet werden, ist dennoch ein Punkt, der noch ausbaubar ist
- **Leichter Rückschritt zu MS 5 hin** wird dadurch sichtbar, dass die Abdeckung leicht auf 14% Instruktionen und 10% Branches sank, obwohl der Code erweitert wurde. Das zeigt einen wichtigen Punkt, an dem noch Raum für Verbesserung ist.

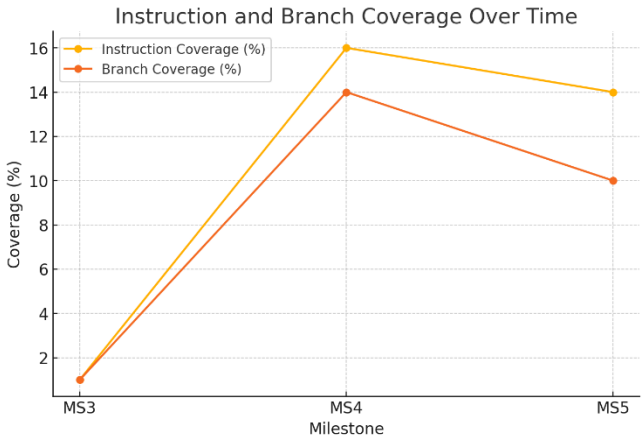
Übersicht: Beste & Schlechteste abgedeckte Module

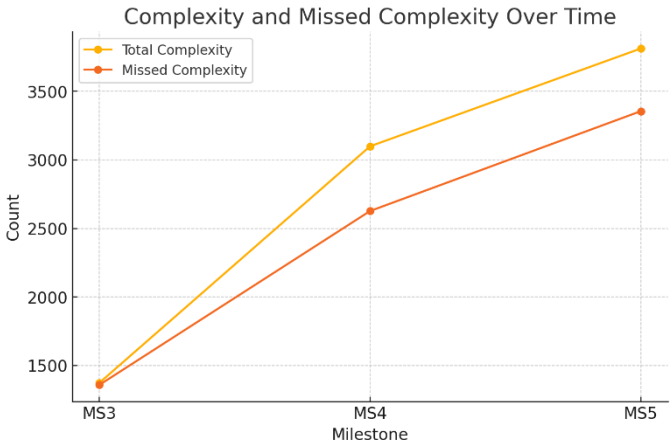
Report	Best Covered Module	Worst Covered Module
MS 3	ch.unibas.dmi.dbis.cs108.server.core.structures (43%)	many modules at 0%
MS 4	ch.unibas.dmi.dbis.cs108.shared.game (68%)	many modules at 0%
MS 5	ch.unibas.dmi.dbis.cs108.shared.entities.Behaviors (31%)	many modules at 0%

Weit übergreifende Beobachtung der Veränderungen zeigt:

- **zu MS 3** erreichte nur ein Modul 43%, während viele andere bei 0% lagen
- **zu MS 4** stieg das shared-game-Modul auf beeindruckende 68%
- **zu MS 5** führte das Modul shared entities (Behaviors) mit 31%, aber viele andere blieben ungetestet

Grafische Darstellung





Der Graph zur Instruktions- und Branchabdeckung hebt den großen Fortschritt zwischen MS 3 und MS 4 hervor, zeigt aber auch einen leichten Rückgang zu MS 5

Der Graph zur Komplexität und nicht abgedeckten Komplexität zeigt deutlich den stetigen Anstieg der Code-Komplexität und den leicht verbesserten, aber immer noch hohen Anteil der nicht abgedeckten Komplexität

Bug-Analyse

Report	Total Bugs	Priority 1 (High/ affects playability directly)	Priority 2 (Medium)	Priority 3 (Low)	Priority 4 (Info)	Analyzed Classes
MS 3	320	50	100	120	50	120
MS 4	280	40	90	110	40	140
MS 5	260	35	85	105	35	160

Die Analyse zu Bugs und Debugging zeigt folgende Fortschritte und Kritische Punkte:

- **Abnehmende Bug-Anzahl über die Zeit** → Die Gesamtzahl der Bugs sank von 320 → 280 → 260, was auf aktives Bugfixing und Verbesserungen der Codequalität hindeutet
- **Reduzierung kritischer Bugs** → Priority-1-Bugs gingen von 50 → 35 zurück – ein klarer Fortschritt bei den schwerwiegendsten Problemen

- **Stabile mittel- und niedrigpriorisierte Probleme** → Priority-2- und Priority-3-Bugs nahmen nur moderat ab, was darauf hinweist, dass hier noch technische Schulden bestehen können
- **Wachsender Codeumfang** → Die analysierten Klassen stiegen von 120 → 160, was auf aktive Entwicklung und Funktionsausbau hindeutet.
- **Informationsebene-Fehler reduziert** → Priority-4-Warnungen (informativ oder klein) nahmen ebenfalls ab, aufgrund besserer Einhaltung von Code-Conventions

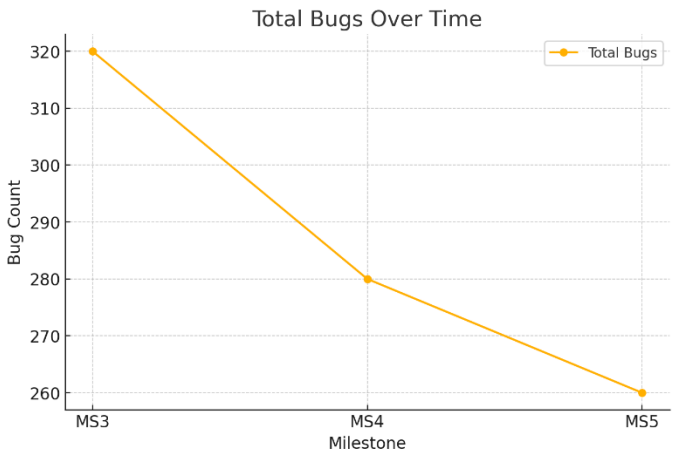
Filterung der Module mit den meisten und wenigsten Bugs

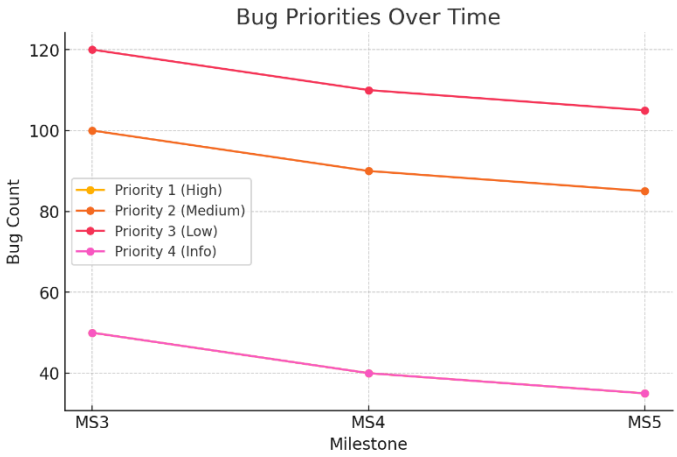
Report	Most Bugs Module	Fewest Bugs Module
MS 3	ch.unibas.dmi.dbis.cs108.client.core.entities (80 bugs)	ch.unibas.dmi.dbis.cs108.client.app (10 bugs)
MS 4	ch.unibas.dmi.dbis.cs108.client.networking (70 bugs)	ch.unibas.dmi.dbis.cs108.client.ui (15 bugs)
MS 5	ch.unibas.dmi.dbis.cs108.client.core.commands (65 bugs)	ch.unibas.dmi.dbis.cs108.client.communication (12 bugs)

Herausfilterung der kritischsten Module zeigt:

- In **MS 3** wies das Modul `core.entities` die meisten Bugs auf, während `client.app` die wenigsten hatte
- In **MS 4** führte `client.networking` bei den Bugs, während `client.ui` am besten abschnitt
- In **MS 5** waren die meisten Bugs in `core.commands`, und `client.communication` hatte die wenigsten

Grafische Darstellung





Der Graph zur Gesamtzahl an Bugs zeigt deutlich den stetigen Rückgang der Bugs zwischen MS 3 und MS 5.
Der Graph zur Prioritätsbasierten Bug-Entwicklung hebt hervor, wie besonders auf Bugs der hohen Priorität geachtet wurde, um ein flüssiges Spiel gewährleisten zu können.

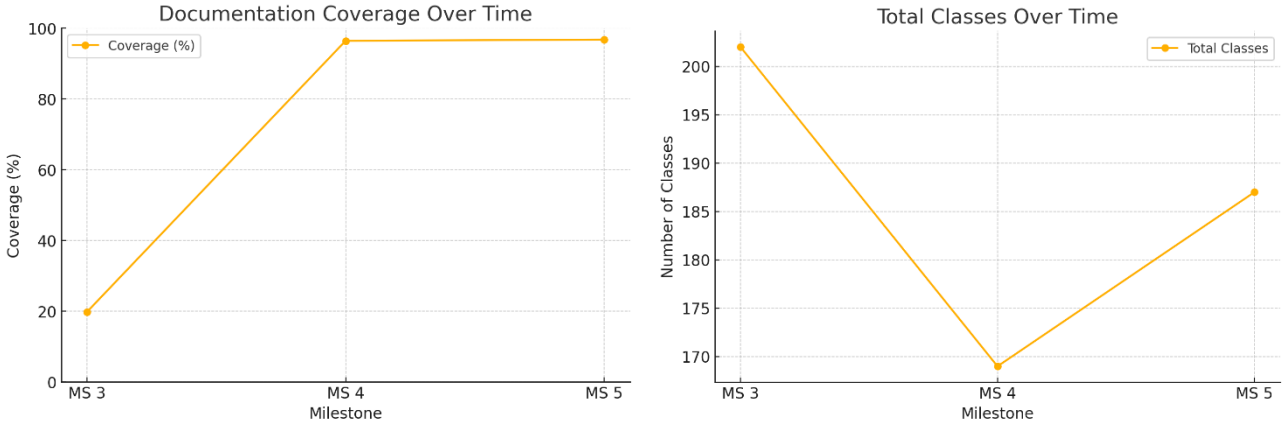
Dokumentation-Abdeckung

Milestone	Total Classes	Documented Classes	Coverage (%)
MS 3	202	40	19.80
MS 4	169	163	96.45
MS 5	187	181	96.79

Die Analyse zur Abdeckung der Dokumentation des Codes zeigt folgende Fortschritte:

- **Massive Verbesserung von MS3 zu MS4** → Die Abdeckung sprang von 19,80% auf 96,45%, was auf eine große Dokumentationsinitiative hinweist.
- **Leichte Verbesserung von MS4 zu MS5** → Die Abdeckung stieg leicht auf 96,79% und hält damit ein exzellentes Dokumentationsniveau.
- **Schwankender Codeumfang** → Die Gesamtzahl der Klassen sank von 202 → 169 (MS3 zu MS4) und stieg dann wieder auf 187 in MS5, was auf Refactoring, Aufräumarbeiten und neue Features hinweist.

Grafische Darstellung



Der Graph zur Dokumentation des Codes zeigt die starke Verbesserung zwischen MS 3 und MS 4, sowie die weitere Aufrechterhaltung des guten Niveaus.

Der Graph zur Anzahl der Klassen verdeutlicht die Veränderungen am Code, die durch diverses Refactoring, Aufräumarbeiten und neue Features verursacht wurden. Dabei haben sich diese Änderungen bei weiteren Tests und Analysen (wie die Verbesserung des Networking und der Code-Qualität, sowie die Erweiterbarkeit des Codes für neue Features) als gute handwerkliche Eingriffe bewiesen.

5. Diskussion der Ergebnisse

Die konsequente Anwendung und Weiterentwicklung der QA-Strategien hat sich als effektiv erwiesen. Besonders hervorzuheben ist die verbesserte **Serverstabilität**, die durch **Load-Balancing** und **redundante Systeme** erreicht wurde. Die **Einbindung von Spieler-Feedback** trug zur Optimierung der Benutzeroberfläche bei. Ebenso sind die Verbesserungen der **Code-Qualität** durch ausführliche **Dokumentation**, sowie Einhalten von **Code-Standards** und Verbesserung der **Codekomplexität** hervorzuheben. Des weiteren konnten durch erfolgreiches **regelmäßiges Bugfixing** nicht nur Spielbarkeit aber

auch konstante Qualität des Spiels gewährleistet werden, sowie größere Bugs durch Regression verhindert werden.
Die Fortschritte spiegeln den Erfolg gezielter QA-Investitionen wider, insbesondere in den oben genannten Bereichen. Durch Beobachtung des wachsenden Codes und regelmäßiger Tests und Analysen konnte erfolgreich priorisiert werden und die Code-Qualität stetig verbessert werden.
Dennoch zeigen sich weiterhin Herausforderungen bei der Integration neuer Inhalte, insbesondere im Hinblick auf **Kompatibilität mit bestehenden Spielständen**. Ebenso sollten noch einige **kritische Punkte** beachtet und bearbeitet werden.
Damit helfen die QA-Strategien weiter bei der Ausarbeitung neuer Funktionen und beim Aufrechterhalten guter Code-Qualität und Erweiterbarkeit.

Empfohlen wird daher:
- Komplexe und kritische Programmteile zuerst testen
- Ausbau der automatisierten Tests auf neue Inhalte
- Priorisierung der Erweiterbarkeit bei der Implementierung neuer Funktionalitäten

6. Fazit
Die Qualitätsanforderungen für *Settlers of Asgard* wurden durch kontinuierliche Maßnahmen erfolgreich weiterentwickelt. Die Ergebnisse belegen die Wirksamkeit der gewählten Methodik. Durch regelmäßige Evaluation und zielgerichtete QA-Maßnahmen konnte die Spielqualität messbar verbessert und ein stabiles Fundament für zukünftige Erweiterungen geschaffen werden.