Präsentation zur Qualitätssicherung

Herzlich willkommen zur Präsentation der Qualitätssicherung unseres Programms zur Simulation von wiederholten Spielen. Nach der Implementation in der letzten Phase folgte in dieser nun das Testen unseres Programmes. (🡪 Folie 2) Neben tests und Bug-Fixes haben wir noch Performance Optimierungen vorgenommen und einige neue Features implementiert.

Da unsere Architektur auf MVC basiert, sind wir beim Testen wie folgt vorgegangen:

* Das Model wurde anhand von JUnit Testklassen getestet. Diese Klassen wurden zu einem beträchtlichen Teil schon während der Implementierungsphase erstellt und jetzt weiter ausgebaut und verfeinert. Insgesamt wurden so in x Testklassen y Tests erstellt. Insgesamt kommen wir im Model auf eine Testüberdeckung von xx%. Das Herzstück, die simulationengine… Pakete wie das plugin- oder repository paket lassen sich nur teilweise mit JUnit sinnvoll testen, da große teile des codes sich auf das laden und speichern von dateien beziehen, was sich am besten manuell testen lässt.
* View und Controller wurden ebenfalls durch ausgiebige manuelle Interaktion mit der GUI getestet.
* Die Fehler, die wir dabei gefunden haben, waren alle schnell behoben und sind nicht erwähnenswert.

(🡪 Performance Folie)

Das einzige größere Problem, auf das wir gestoßen sind, ist das Speichern von Simulationsergebnissen. Wie Ihnen vielleicht auch schon aufgefallen ist, waren die simulationsergebnis dateien oft sehr groß, bei großen multikonfigurationen konnte es bei der serialisierung sogar zu einem outofmemory error kommen. Grund war, dass die Ergebnisse nicht komprimiert wurden und deutlich mehr abgespeichert wurde als nötig. Wie man hier sehen kann, hat uns die Komprimierung etwa einen Faktor 1000 beim Speicherbedarf eingespart, man kann jetzt also ohne probleme auch große simulationen abspeichern.

Ein weiteres Performance-Ziel, das wir uns für diese Phase gesetzt haben, war die Optimierung der Laufzeit der Simulationen. Dazu haben wir einen Profiler verwendet, der misst, welche Methodenaufrufe am meisten Zeit benötigen. Wie man hier sehen kann, ist das Auswerten der (mglw. Gemischten) Strategien mit 99.8% der gesamten Zeit ein Bottleneck. Das liegt vor allem daran, dass Strategien wie tit for tat oder grim auf frühere Spiele Bezug nehmen. Indem wir den entsprechenden Mechanismus etwas optimiert haben, konnten wir eine durchaus beeindruckende Beschleunigung erreichen. Etwa dauerte die Standardkonfiguration vor der Optimierung 2 – 3 Minuten, jetzt läuft sie in sagenhaften 2 Sekunden.

Weiterhin haben wir uns ein paar neue Features ausgedacht und implementiert:

* Man kann nun Simulationen wieder aus der History entfernen, um sie übersichtlich zu halten, wenn man viele Simulationen gestartet hat.
* Bei dem Diagramm für die Strategieverteilungen kann man nun auswählen, welche Gruppen betrachtet werden sollen. So kann man die Entwicklung der verschiedenen Gruppen getrennt untersuchen.

Ein weiteres Ziel unserer Qualitätssicherung war das Finden von spieltheoretischen Erkenntnissen.

Durch die extreme Vielfalt an Möglichkeiten, die man bei der Parametrisierung einer Simulation hat, ist es schwierig, universelle Konzepte festzustellen. Einige Erkenntnisse konnten wir trotzdem gewinnen:

* Wie bereits zu Beginn vermutet, stellt sich ein Gleichgewicht in der regel erst dann ein, wenn alle Agenten dieselbe Strategie haben. Das liegt einfach daran, dass, solange noch große Unterschiede in den Strategien bestehen, auch noch große Anpassungen erfolgen. Das hat mehrere Konsequenzen:  
  Simulationen mit reinen strategien haben meist nur zwei mögliche ausgänge: entweder hat am ende jeder dieselbe (reine) strategie, oder es stellt sich nie ein gleichgewicht ein. Das passiert zum beispiel, wenn
* Ranggleichgewicht sinnlos:
* Desto höher die Vergleichswahrscheinlichkeit oder die Adaptionswahrscheinlichkeit bei einem Adaptionsmechanismus ist desto schneller stellt sich ein Gleichgewicht ein
* Bei den Erfolgsquantifizierungsalgorithmen "Payoff in the Last Adaptionstep" und "Sliding Mean" stellt sich viel schneller ein Gleichgewicht ein als bei "Total Capital" und "Total Payoff"