Übung 3: Event Sourcing

Michael Schleichardt

27. Juni 2012

1 Einführung

1.1 Event Sourcing

Anwendungen mit Event Sourcing sammeln alle Änderungen des Applikationszustandes als Sequenz von Events.[1] Folglich sind Änderungen der Applikation revisionssicher und es kann nicht nur der Zustand abgerufen werden, sondern auch der Weg dort hin.

1.2 Anwendungsbeispiel

Als Anwendungsbeispiel wurde das Aufstellen eines Mühlespiels gewählt. Der Zustand der Applikation besteht aus der Verteilung der Steine auf dem Spielfeld und der Information welcher Spieler am Zuge ist. Die Events sind die Positionierung der eigenen Spielsteine. Brettspiele wie Mühle sind besonders gut für Event Sourcing geeignet, da man alle vorherigen Schritte analysieren kann und ggf. alternative Vorgehensweisen testen kann. Schon allein mit der Startphase eines Mühlespiels können die Techniken des Event Sourcing angewandt werden.

1.3 Technologien

Zur Umsetzung wurde CoffeeScript gewählt. CoffeeScript ist dynamische Scriptsprache, die in JavaScript kompiliert wird, jedoch eine schlankere Syntax und Klassen bietet und im Kontext von Rails, NodeJS und dem Play Framework eingesetzt wird.[2] Als Javascript Präprozessor ist CoffeeScript gut geeignet mit Events und dem JSON-Datenformat zu arbeiten. Für Tests wurde QUnit eingesetzt. Des weiteren wurde jQuery eingesetzt.

1.4 GUI

Im Spiel werden HTML5 mit Twitter Bootstrap und CSS3 eingesetzt. In Abbildung 1 wird die gesamte GUI dargestellt. Links oben befindet sich das Spielfeld in dem die Steine der Spieler positioniert werden. Rechts daneben befindet sich der Stapel mit den noch zur Verfügung stehenden Steinen der Spieler. Die Steine werden durch Drag and Drop bewegt.

Unter dem Spielfeld befinden sich Buttons und darunter ein Texteingabefeld, welches das aktuelle Log der Applikation als JSON enthält. Die Events als Text zu speichern mach die Nachrichten unabhängig vom Quellcode und ist leichter zu Pflegen bei Änderungen am Code. Die Aufgaben der Buttons werden in den nächsten Abschnitten erläutert.

[1] Vgl. http://martinfowler.com/eaaDev/EventSourcing.html, abgerufen am 27.06.2012
[2] Vgl. http://de.wikipedia.org/wiki/CoffeeScript, abgerufen am 27.06.2012

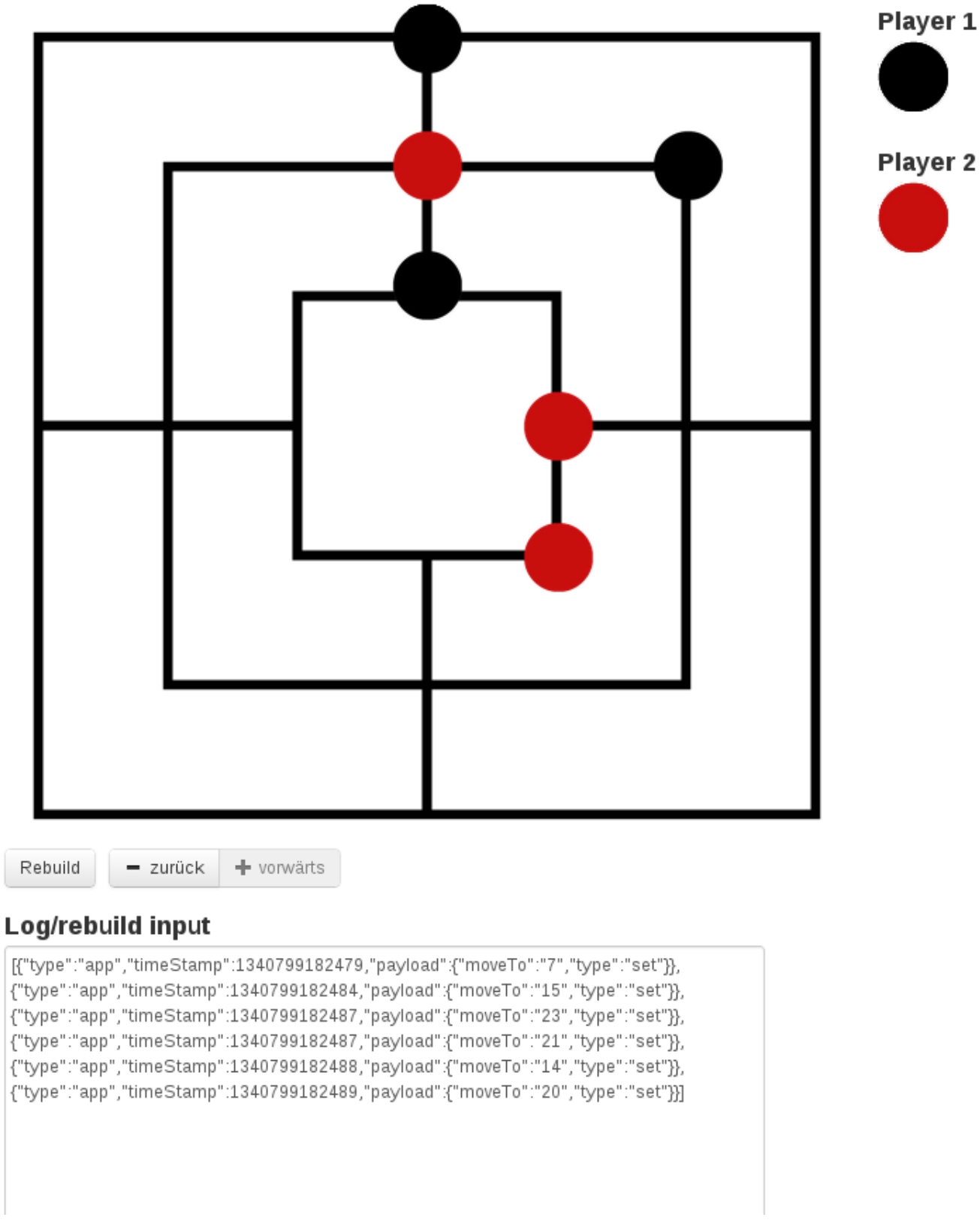
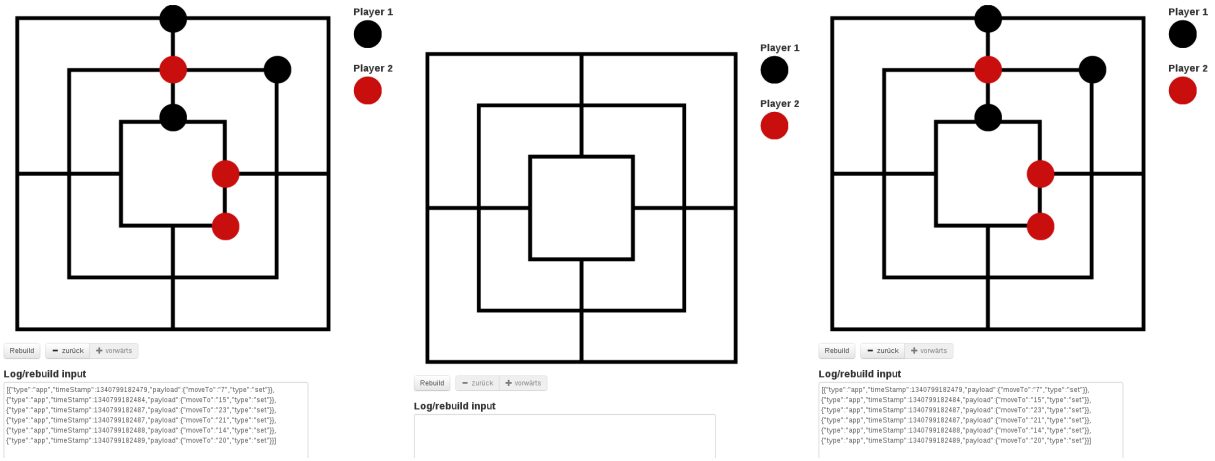


Abbildung 1: Screenshot mit allen Elementen



(a) Zustand erzeugen (b) Zustand löschen (c) Zustand wiederherstellen

Abbildung 2: Complete Rebuild

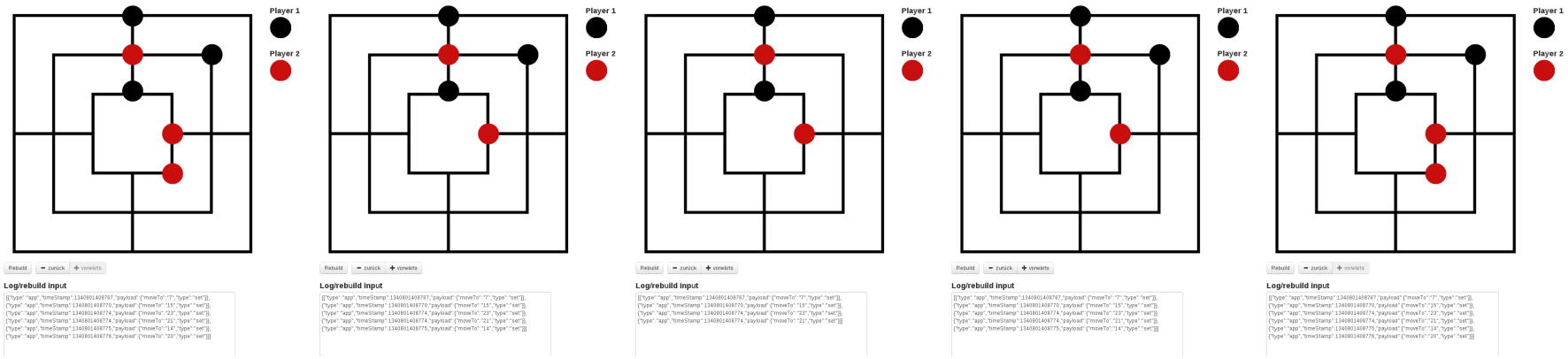
2 Features

2.1 Complete Rebuild

Das Spiel gibt jeweils das aktuelle und vollständige Log aus. Wird die Log Ausgabe kopiert, das Spiel durch Neuladen des Fensters zurückgesetzt. Ist erstmal der Zustand verloren. Die Logdaten können jedoch wieder in das leere Textfeld eingefügt werden und durch einen Klick auf den Button „Rebuild“ wird der Anwendungszustand wiederhergestellt, wie in Abbildung 2 veranschaulicht.

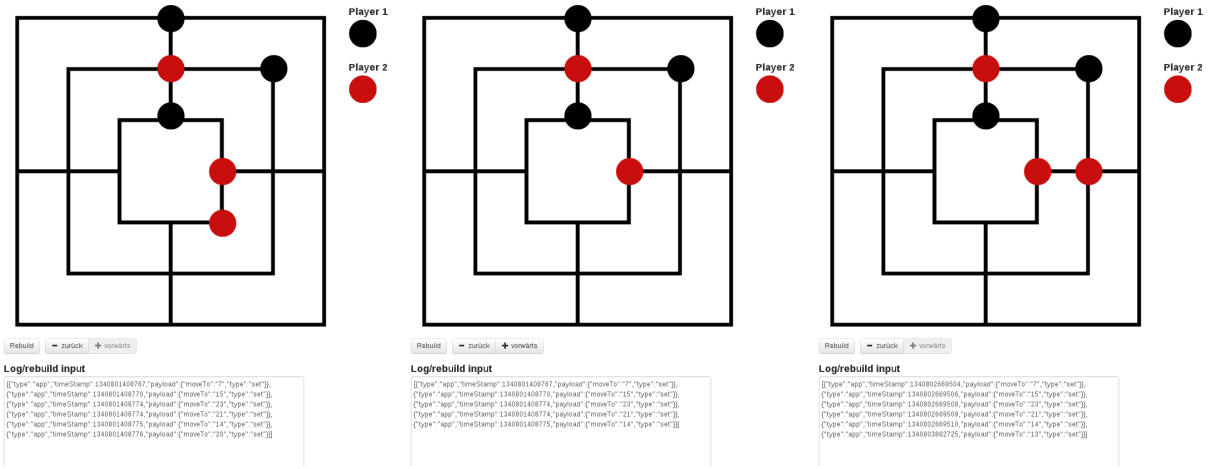
Das verwendete Log ist in JSON und speichert in „payload.moveTo“, wo ein Stein hingesetzt wurde. Es wird implizit angenommen, dass jede ungerade „set“-Operation zu Spieler 1 gehört und die geraden zu Spieler 2. Nachfolgend wird ein Auszug dargestellt:

[{"type":"app","timeStamp":1340799876883,"payload":{"moveTo":"7","type":"set"}},
{"type":"app","timeStamp":1340799876887,"payload":{"moveTo":"15","type":"set"}},
{"type":"app","timeStamp":1340799876890,"payload":{"moveTo":"23","type":"set"}},
{"type":"app","timeStamp":1340799876891,"payload":{"moveTo":"21","type":"set"}},
{"type":"app","timeStamp":1340799876892,"payload":{"moveTo":"14","type":"set"}},
{"type":"app","timeStamp":1340799876893,"payload":{"moveTo":"20","type":"set"}}]



(a) Ausgangslage (b) 1x zurück (c) 2x zurück (d) 1x vorwärts (e) 2x vorwärts

Abbildung 3: Temporal Querys



(a) Aktueller Spielstand (b) 1x zurück (c) Stein anders gesetzt

Abbildung 4: Alternative Zeitlinie

2.2 Temporal Querys

Das Spiel kann nicht nur den aktuellen Zustand darstellen, sondern auch vorherige.

Wurde mindestens ein Zug gespielt, wird der zurück-Button aktiv (nicht der im Browser, sondern unter dem Spielbrett) und ermöglicht einem pro Klick einen Zug zurückzuschauen.

Wurde mindestens einmal ein Zug zurückgesprungen, wird der vorwärts-Button aktiv, der pro Klick vorwärts in der Historie browst. Das zweimalige zurückbrowsen und dann zurück zum aktuellen Stand wird in Abbildung 3 gezeigt (Steine verschwinden und tauchen wieder auf).

Wurde mindestens ein Schritt zurückgebrowst, besteht die Möglichkeit eine alternative Zeitlinie für das Spiel zu starten in dem man einen Spielstein setzt. Dadurch wird der Vorwärts-Button gesperrt und die vorigen „nachfolgenden“ Spielzüge gehen verloren. Dies ist nützlich, wenn man einen Stein fehlplatziert hat und es gleich wieder korrigieren möchte oder man gesehen hat, dass man ab einem Punkt schlecht gespielt hat und dort hin zurückkehrt und den Fehler nicht wiederholt. Abbildung 4 zeigt wie Spieler 2 einen Zug gemacht hat, zurück geklickt hat und dann einen Spielstein neben den anderen positioniert hat statt darunter.

2.3 Event Replay

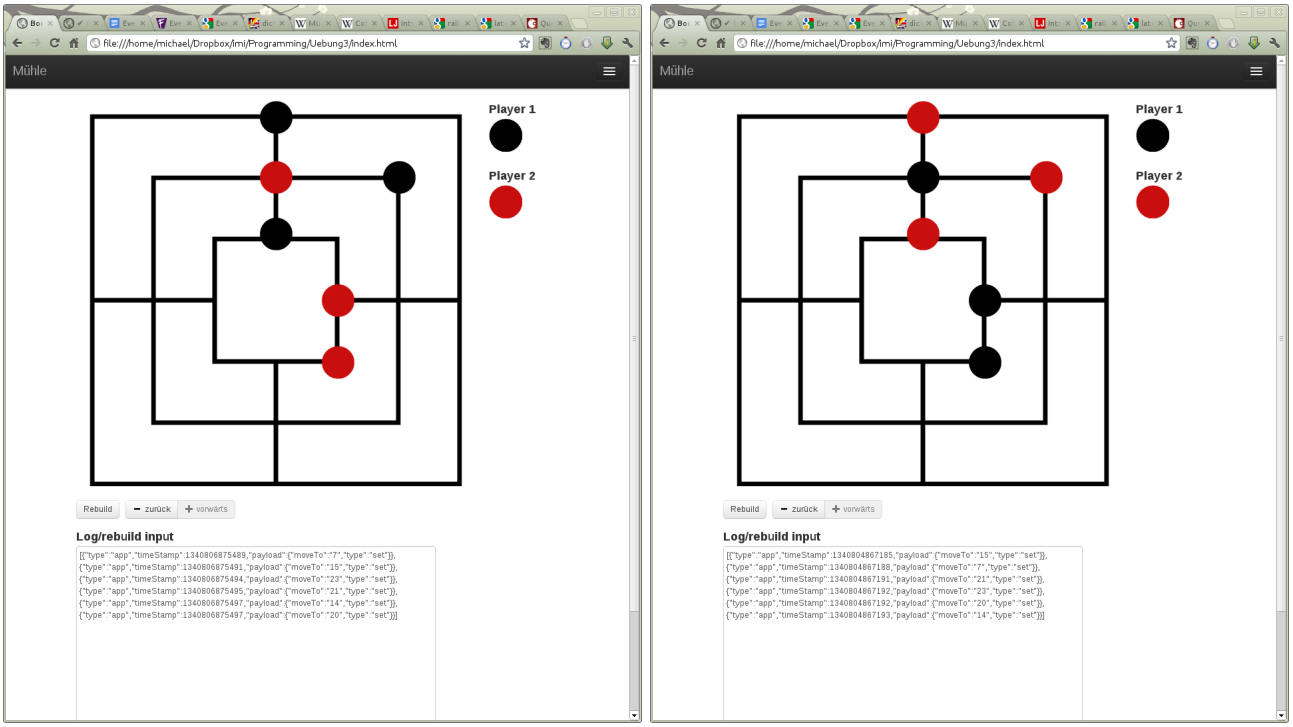
Event Replay ermöglicht es Events zu ändern bzw. ihre Reihenfolge zu ändern durch Manipulierung der Log-Events. Z.B. werden im Log die ungeraden Events mit ihren Nachfolgern ausgetauscht und auf „Rebuild“ geklickt.

Vorher:

[{"type":"app","timeStamp":1340799876883,"payload":{"moveTo":"7","type":"set"}},
{"type":"app","timeStamp":1340799876887,"payload":{"moveTo":"15","type":"set"}},
{"type":"app","timeStamp":1340799876890,"payload":{"moveTo":"23","type":"set"}},
{"type":"app","timeStamp":1340799876891,"payload":{"moveTo":"21","type":"set"}},
{"type":"app","timeStamp":1340799876892,"payload":{"moveTo":"14","type":"set"}},
{"type":"app","timeStamp":1340799876893,"payload":{"moveTo":"20","type":"set"}}]

Nachher:

[{"type":"app","timeStamp":1340799876883,"payload":{"moveTo":"15","type":"set"}},



(a) Vorher (b) Nachher

Abbildung 5: Event Replay


```
{"type":"app","timeStamp":1340799876887,"payload":{"moveTo":"7","type":"set"}},
{"type":"app","timeStamp":1340799876890,"payload":{"moveTo":"21","type":"set"}},
{"type":"app","timeStamp":1340799876891,"payload":{"moveTo":"23","type":"set"}},
{"type":"app","timeStamp":1340799876892,"payload":{"moveTo":"20","type":"set"}},
{"type":"app","timeStamp":1340799876893,"payload":{"moveTo":"14","type":"set"}}]
```


Wie in Abbildung 5 gezeigt, werden somit die roten durch die schwarzen Steine ausgetauscht und umgekehrt. Implementiert ist es als Complete Rebuild und nicht als Reversion, da die Anzahl der Spielzüge so klein ist, dass es kaum zu Performanceproblemen kommen kann und weniger Code zu warten ist, da Complete Rebuild schon implementiert ist. Prinzipiell ist Reversion auf Mühle mit dem Log anwendbar.

3 Fazit

Applikationen, die Event Sourcing einsetzen, bieten sehr mächtige Analyse- und Korrekturmöglichkeiten, die nur sehr schwer mit klassischen zustandsbehafteten Systemen erreichbar sind.

Das Verwenden von JSON als Eventdarstellungsform hat sich bewährt. Es hat Code-Änderungen möglich gemacht und sorgt auch für eine Entkopplung der Objekte und Funktionen. Ohne die funktionalen Programmiermöglichkeiten von CoffeeScript, JavaScript und jQuery wäre es erheblich schwieriger gewesen die Anwendung zu entwickeln. Jedoch fehlte das aus Scala bekannte Pattern Matching zum Parsen der Nachrichten.

Für zustandsbehaftete System sind prozedurale und objektorientierte Programmierung okay, für Event Sourcing ist funktionale Programmierung ein besonders geeignetes Paradigma.