



DIPLOMARBEIT

Objektorientierte Entwicklung eines GUI-basierten Tools für die Simulation ereignisbasierter verteilter Systeme

Durchgeführt an der

Fachhochschule Aachen

mit Erstprüfer und Betreuer Prof. Dr.-Ing. Martin Oßmann

und Zweitprüfer Prof. Dr. rer. nat. Heinrich Fassbender

durch

Cand. Inform. (FH) Paul C. Bütow

Matthiashofstr. 15

D-52064 Aachen

Aachen, den 21. Juni 2008

Danksagung

Ohne die Hilfe bestimmter Personen wäre die Anfertigung dieser Diplomarbeit viel schwieriger gewesen. Daher möchte ich mich bei den Folgenden bedanken:

- Martin Oßmann für die Betreuung der Diplomarbeit
- Andre Herbst, für das Testen des Simulators; durch seine Hilfe wurden viele Mängel und Bugs aufgedeckt
- Meinen Bruder Florian Bütow, für Tipps und Tricks rund um Java, für die Bereitstellung eines Buches sowie für das Testen des Simulators
- Meine Eltern Jörn und Leslie Bütow, die mir das Studium ermöglichten und stets für alle Dinge ein offenes Ohr hatten

Inhaltsverzeichnis

1. Einleitung	7
1.1. Was ist ein Verteiltes System?	7
1.2. Motivation	7
2. Grundbegriffe	8
2.1. Client/Server Modell	8
2.2. Prozesse und deren Rollen	8
2.3. Nachrichten	9
2.4. Lokale und globale Uhren	9
2.5. Ereignisse	9
2.6. Protokolle	9
3. Der Simulator	10
3.1. Graphical User Interface (GUI)	11
3.1.1. Toolbar	11
3.1.2. Visualisierung	11
3.1.3. "Sidebar"	11
3.1.4. Loggfenster	11
3.1.5. Expertenmodus	11
3.2. Ereignisse	11
3.2.1. Prozessabsturz	11
3.2.2. Prozesswiederbelebung	11
3.2.3. Aktivierung und Deaktivierung von Protokollen	11
3.2.4. Weitere Protokollereignisse	11
3.3. Protokolle	11
3.3.1. Beispiel (Dummy) Protokoll	11
3.3.2. Das Ping-Pong Protokoll	11
3.3.3. Das Broadcast-Sturm Protokoll	11

3.3.4. Das Protokoll zur internen Synchronisierung in einem synchronen System	11
3.3.5. Christians Methode zur externen Synchronisierung	11
3.3.6. Berkeley Algorithmus zur internen Synchronisation	11
3.3.7. Das Ein-Phasen Commit Protokoll	11
3.3.8. Das Zwei-Phasen Commit Protokoll	11
3.3.9. Der ungenügende (Basic) Multicast	11
3.3.10. Der zuverlässige (Reliable) Multicast	11
3.4. Zeitformate	11
3.4.1. "Normale Zeit"	11
3.4.2. Die Logische Uhr von Lamport	11
3.4.3. Die Vektor-Zeitstempel	11
3.5. Einstellungen	11
3.5.1. Simulationseinstellungen	11
3.5.2. Prozesseinstellungen	11
3.5.3. Protokolleinstellungen	11
4. Die Implementierung	12
4.1. Gliederung der Pakete	13
4.2. Editoren	13
4.3. Ereignisse	13
4.3.1. Interne Ereignisse	13
4.4. Protokolle	13
4.4.1. Protokoll-API	13
4.5. Serialisierung von Simulationen	13
4.5.1. Rückwärtskompatibel	13
4.6. Programmierrichtlinien	13
4.7. Entwicklungsumgebung	13
5. Ausblick	14
A. Schemata	15
B. Akronyms	16
C. Literaturverzeichnis	17

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1.

Einleitung

1.1. Was ist ein Verteiltes System?

1.2. Motivation

[Bri03] Zitat.

Kapitel 2.

Grundbegriffe

Für das Verständnis wie die Simulation von verteilten Systemen funktioniert, werden hier einige Grundbegriffe beschrieben.

2.1. Client/Server Modell

Der Simulator basiert auf dem Client/Server Prinzip. Bei jeder sinnvollen Simulation gibt es mindestens einen teilnehmenden Client und einen Server. Bei komplexen Simulationen können auch mehrere Clients und/oder Server mitwirken.

2.2. Prozesse und deren Rollen

Ein verteiltes System wird anhand von Prozessen simuliert. Jeder Prozess nimmt hierbei eine oder mehrere Rollen ein. Beispielsweise kann ein Prozess die Rolle eines Clients einnehmen und ein weiterer Prozess die Rolle eines Servers. Ein Prozess kann auch Client und Server gleichzeitig sein. Es ist auch möglich, dass ein Prozess die Rollen mehrerer Server und Clients auf einmal einnimmt. Ob das sinnvoll ist hängt vom Szenario ab. Um einen Prozess eindeutig zu kennzeichnen besitzt jeder Prozess eine eindeutige Prozess-Identifikationsnummer (PID).

2.3. Nachrichten

Damit das Client/Server Modell angewandt werden kann, müssen Nachrichten verschickt werden können. Eine Nachricht kann von einem Client- oder Serverprozess verschickt werden und kann beliebig viele Empfänger haben. Um eine Nachricht eindeutig zu kennzeichnen besitzt jede Nachricht eine eindeutige Nachrichten-Identifikationsnummer (NID).

2.4. Lokale und globale Uhren

In einer Simulation gibt es **genau eine** globale Uhr. Sie stellt die aktuelle und **immer korrekte** Zeit dar. Eine globale Uhr geht nie falsch.

Zudem besitzt jeder beteiligter Prozess eine eigene lokale Uhr. Sie stellt die aktuelle, jedoch nicht zwangsmäßig global-korrekte, Zeit des jeweiligen Prozesses dar. Wenn die Prozesszeit nicht korrekt ist (nicht der globalen Zeit gleicht), dann wurde die Prozessuhr entweder im Laufe einer Simulation neugestellt oder sie besitzt eine Uhrabweichung. Eine Uhrabweichung gibt an, um wieviel eine Uhr falsch geht. Wenn eine lokale Uhr nicht neugesetzt wird und auch keine Uhrabweichung hat, dann gibt sie stets die korrekte globale Zeit wieder.

2.5. Ereignisse

Eine Simulation besteht aus der Hintereinanderausführung von endlich vielen Ereignissen. Beispielsweise kann es ein Ereignis geben, welches einen Prozess eine Nachricht verschicken- oder selbst abstürzen läßt. Jedes Ereignis tritt zu einem bestimmten Zeitpunkt ein. Wenn es zeitgleiche Ereignisse gibt, so werden sie ebenso hintereinander ausgeführt, behalten aber in der Simulation die selben Ausführzeiten.

2.6. Protokolle

Kapitel 3.

Der Simulator

3.1. Graphical User Interface (GUI)

3.1.1. Toolbar

3.1.2. Visualisierung

Zeitachse

Prozesse

Nachrichten

3.1.3. “Sidebar”

Ereignisse

Variablen

3.1.4. Loggfenster

3.1.5. Expertenmodus

Lamport- und Vektorzeit

Loggfilter

Lokale und globale Ereignisse

3.2. Ereignisse

3.2.1. Prozessabsturz

11

3.2.2. Prozesswiederbelebung

3.2.3. Aktivierung und Deaktivierung von Protokollen

Kapitel 4.

Die Implementierung

4.1. Gliederung der Pakete

4.2. Editoren

4.3. Ereignisse

4.3.1. Interne Ereignisse

4.4. Protokolle

4.4.1. Protokoll-API

4.5. Serialisierung von Simulationen

4.5.1. Rückwärtskompatibel

4.6. Programmierrichtlinien

4.7. Entwicklungsumgebung

Kapitel 5.

Ausblick

Anhang A.

Schemata

Anhang B.

Akronyms

GUI Graphical User Interface

NID Nachrichten-Identifikationsnummer

PID Prozess-Identifikationsnummer

VS Verteiltes System

Anhang C.

Literaturverzeichnis

[Bri03] Brian.H.Wilcox. Robotic vehicle system engineering. Internet, April 2003.
<http://team.caltech.edu/members/RoboticVehicleSystemEngineering3.pdf>.