

Diplomarbeit: Objektorientierte Entwicklung eines GUI-basierten Tools für die ereignisbasierte Simulation verteilter Systeme

Von Paul C. Bütow

1. Prüfer: Prof. Dr.-Ing. M. Oßmann
2. Prüfer: Prof. Dr. rer. nat. H. Faßbender

Fachhochschule Aachen - 18. August 2008

1 Einleitung

- Was ist ein verteiltes System?
- Motivation

2 Grundlagen

- Client/Server
- Prozesse
- Protokolle
- Uhren
- Ereignisse

3 Der Simulator

- Konfigurationsmöglichkeiten
- Alle bereits eingebauten Protokolle
- Beispiele / Vorführungen
- Implementierung von Protokollen (Protokoll-API)

4 Ende

- Ausblick
- Zahlen und Fakten

Was ist ein verteiltes System?

- Zitat A. Tanenbaum und M. van Steen, Verteilte Systeme:
“Ein verteiltes System ist eine Menge voneinander unabhängiger Computer, die dem Anwender wie ein einzelnes, kohärentes System erscheinen”
- Anwender muss sich nur mit dem vor ihm befindlichen Computer auseinandersetzen
- Verteiltes System stellt die Kommunikation mit anderen Computern sicher
- Gemeinsame Nutzung von Ressourcen

Motivation

- Betrachtung von verteilten Systemen aus einer anderen Sicht (Lehr- und Lernzwecke)
- Transparente Darstellung von verteilten Systemen
- Entwicklung eines Simulators (VS-Simulator oder auch VS-Sim.)
 - Flexibilität
 - Einfachheit in der Bedienung
 - Erweiterungsmöglichkeiten

Grundlagen - Client/Server

- Client/Server Kommunikation
- Mindestens einen Client und einen Server
- Verschicken von Nachrichten
 - Client kann nur Servernachrichten verarbeiten
 - Server kann nur Clientnachrichten verarbeiten



Grundlagen - Prozesse

- Simulation von (beliebig vielen) verteilten Prozessen
- Jeder Prozess kann Rollen einnehmen
 - Prozess ist Server
 - Prozess ist Client
 - oder Prozess ist gleichzeitig Client und Server

Protokolle

- Ein Protokoll definiert das Verhalten von Clients und Severn
 - Was in den Nachrichten verschickt wird
 - Wie auf den Erhalt einer Nachricht reagiert wird
 - Was bei Wecker-Ereignissen passiert

Protokolle

- Jede Nachricht gehört einem Protokoll an
 - Nachricht nur verarbeitbar, wenn Empfänger das Protokoll der Nachricht versteht
 - Alle anderen eintreffenden Nachrichten werden nicht verarbeitet

Uhren und Zeit

- Simulation hat eine globale Uhr
- Jeder Prozess hat:
 - Eigene Prozessuhr / Uhrabweichung
 - Lamport-Zeitstempel
 - Vektor-Zeitstempel

Ereignisse

- Simulation: Hintereinanderausführung von Ereignissen
- Ereignis bei lokaler Prozesszeit oder globaler Zeit
 - Prozessabsturz/Prozesswiederbelebung
 - Aktivierung oder Deaktivierung eines Protokolls client- oder serverseitig
 - Starten von Client- bzw. Serveranfragen
- Weitere (interne) Ereignisse
 - Zufällige Ereignisse
 - Wecker-Ereignisse
 - Nachrichtenempfangs-Ereignisse

Verschiedene Einstellungsmöglichkeiten

- Vom Anwender einstellbar und abspeicherbar
 - Globale Simulationseinstellungen
 - Separate Einstellungen für jeden Prozess
 - Separate Einstellungen für jedes Protokoll für jeden Prozess
- Vom Entwickler einstellbar
 - `prefs/VSDefaultPrefs.java`
 - Alle Standardeinstellungen
 - Spracheinstellungen

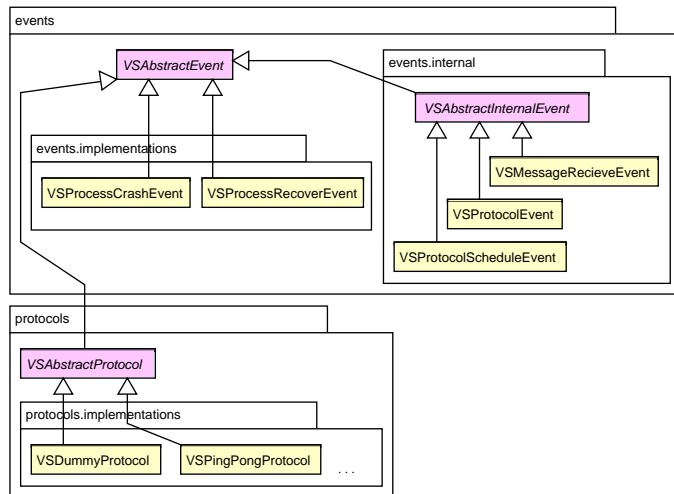
Derzeit verfügbare Protokolle

- Das Beispiel (Dummy) Protokoll
- Das Ping-Pong Protokoll
- Das Broadcast Protokoll
- Das Protokoll zur internen Synchronisierung in einem synchronen System
- Das Protokoll zur Christians Methode zur externen Synchronisierung
- Der Berkeley Algorithmus zur internen Synchronisierung
- Das Ein-Phasen Commit Protokoll
- Das Zwei-Phasen Commit Protokoll
- Der ungenügende (Basic) Multicast
- Der zuverlässige (Reliable) Multicast

Beispiele

- Das Beispiel (Dummy) Protokoll
- Das Ping-Pong Protokoll
- Ping-Pong Sturm
- Das Protokoll zur Christians Methode zur externen Synchronisierung (*wenn genug Zeit*)
- Der zuverlässige (Reliable) Multicast (*wenn genug Zeit*)

Ereignisse und Protokolle / Klassenvererbungen



Methoden einer Protokollklasse

- `public VSDummyProtocol()` (Konstruktor)
- `public void onClientInit()`
- `public void onClientReset()`
- `public void onClientStart()`
- `public void onClientRecv(VSMessage message)`
- `public void onClientSchedule()`
- `public void onServerInit()`
- `public void onServerReset()`
- `public void onServerStart()`
- `public void onServerRecv(VSMessage message)`
- `public void onServerSchedule()`

Geerbte Methoden und Attribute

■ Geerbte Attribute

- `protected VSAbstractProcess process`
- `protected VSPrefs prefs`

■ Geerbte Methoden

- `public void log()`
- `public String toString()`
- `public void sendMessage(VSMessage message)`
- `public void scheduleAt(long time)`
- `public void removeSchedules()`
- ... und viele mehr

Denkbare Erweiterungen

- Wahrscheinlich Veröffentlichung als Open Source
- Neue Ereignisse und Protokolle
- Erweiterungen als Plugins
- Beliebige lange Simulationen
- Scroll- und Zoomfunktionen
- Ereignisse bei Lamport- und Vektor-Zeitstempel
- ... und vieles mehr

VS-Sim.: Zahlen und Fakten

- Quelltext-Dateien: 61
- Java-Pakete: 12
- LOC: 15710
- Generierte Javadocs: 2.2MB
- VS-Sim-1.0.jar: 142KB
- Bereits eingebaute Protokolle: 10
- Einstellungsmöglichkeiten: 163 (ohne Protokolle)

Danke für die Aufmerksamkeit

- Quelltext-Dateien: 61
- Java-Pakete: 12
- LOC: 15710
- Generierte Javadocs: 2.2MB
- VS-Sim-1.0.jar: 142KB
- Bereits eingebaute Protokolle: 10
- Einstellungsmöglichkeiten: 163 (ohne Protokolle)