Dokumentation der praktischen Übung in Betriebssysteme an der HS-Mannheim im SS2011: Scheduling Simulator

Nikolaus Dafinger
Patrick Elsässer
Christof Hlipala
Oleg Zeiler

Fakultät für Informatik
Hochschule Mannheim
Paul-Wittsack-Straße 10
68163 Mannheim

Zusammenfassung Dieses Dokument beinhaltet die Dokumentation der praktischen Übung in der Vorlesung Betriebssysteme (BTS) im Sommersemester 2011. Die Aufgabe war, ein Programm mit grafischer Oberfläche zu erstellen, welches für verschiedene Prozesse mit vorgegebener Laufzeit und Priorität unterschiedliche Schedulingverfahren durchführt und diese danach auf Effizienz auswertet. Zuerst werden die Teammitglieder und deren Aufgabenbereiche vorgestellt. Die Kommunikation mit den entsprechenden Kommunikationsmedien im Team wird erläutert. Danach wird näher auf die Programmiersprache Java und die Darstellungssprache LaTeX, welche verwendet wurden und die eingesetzte Software eingegangen. Der Aufbau der Software wird anhand des MVC-Prinzips erläutert. Danach wird die Software an sich mit einem kleinen Programmteil vorgestellt.

In der folgenden Dokumentation wird nun das Programm "Scheduling Simulator", sowie die Architektur des Programms, Arbeitsweisen und die eingesetzten Werkzeuge erläutert. Anhand von Code - Beispielen und Bildern wird näher auf die eingesetzten Methoden zur Umsetzung der Aufgabenstellung eingegangen. Des Weiteren wird die Gruppeneinteilung und die daraus folgende Aufgabenverteilung erläutert.

1 Arbeitsweise

Im Folgenden wird das Vorgehen der Gruppe vorgestellt, mit der Aufgabenverteilung im Team, den Kommunikationswegen und den eingesetzten Werkzeugen.

1.1 Aufgabenverteilung im Team

Innerhalb des Teams wurden verschiedene Schwerpunkte gesetzt, was der Gruppe die Arbeitsweise erleichterte. Denn durch die kürzeren Kommunikationswege konnte viel schneller gearbeitet werden. Die Arbeit wurde auf zwei Gruppen verteilt. Das Programmierteam bestand aus Christof Hlipala und Oleg Zeiler, das Dokumentationsteam aus Nikolaus Dafinger und Patrick Elsässer.

Programmierteam Die Aufgaben im Entwicklerteam bestanden darin, zu überlegen welche Klassen gebraucht werden und wie diese miteinander interagieren. Eine Weitere Aufgabe war es, einen Entwurf für die graphische Oberfläche zu entwerfen. Das Programmierteam setzte die Ausarbeitung in Java um. Da wir die Architektur des Programms nach dem Model - View - Controller - Prinzip umgesetzt haben, konnten wir die verschiedenen Komponenten im Programmierteam aufteilen.

Dokumentationsteam Die Aufgaben des Dokumentationsteams bestanden aus der Dokumentation der Software sowie der Erstellung einer Präsentation in LaTeX. Zuerst wurde gemeinsam erarbeitet, was die Dokumentation der Teamarbeit enthalten muss. In Absprache mit dem Programmierteam soll die Software analysiert und dokumentiert werden. Die Vorgehensweise beim Programmieren und im ganzen Projekt soll ebenfalls analysiert werden. Zur Erstellung der Dokumentation sowie der Präsentation wurden zwei GoogleDocs-Dokumente erstellt, auf die alle Teammitglieder Zugriff hatten. So konnte teilweise gleichzeitig an der Dokumentation gearbeitet werden und kommuniziert werden. Da die Dokumentation in LaTeX erstellt werden sollte, wurde schon relativ früh angefangen, sich diese Darstellungssprache anzueignen. Hier kümmerte sich Patrick Elsässer vor allem um die Umsetzung der Präsentation in der LaTeX-Klasse Beamer und Nikolaus Dafinger beschäftigte sich mit der schriftlichen Dokumentation, bei der die Klasse Lecture Notes in Computer Science verwendet wurde. Die Einarbeitung erfolgte mit verschiedenen Medien. Als Vorlage dienten die vom Dozenten Christian Baun bereitgestellten LaTeX-Dokumente mit Erläuterungen. Diese enthalten die meisten nötigen Zeilen zur Erstellung eines LaTeX-Dokumentes. Diverse Tutorials zum Arbeiten mit MikTeX wurden ebenfalls verwendet. Vor allem bei der Installation von zusätzlich benötigten Paketen wurde auch Rat in verschiedenen Foren gesucht. Eine Erfahrung beim Umgang mit LaTeX war, dass sich nach einer gewissen Gewöhnungsphase besser arbeiten ließ. Die Dokumentation wurde auch zeitlich parallel zur Erstellung der Software durchgeführt. Da alle Teammitglieder Zugriff auf SVN hatten, konnte die Programmierung mit verfolgt werden und somit schon einen Eindruck vom Programm erlangt. Außerdem wurden die Eckpfeiler des Programms bereits früh gemeinsam abgesprochen. Zusätzlich gab es auch Teile der Dokumentation, die schon ohne Kenntnis des fertigen Programmes erstellt werden konnten. Diese Arbeitsweise wurde gewählt, damit früher mit der Erstellung der Dokumentation begonnen werden konnte. Andernfalls wäre

man zeitlich in Verzug geraten, da erst spät, nämlich nach der Erstellung des vollständigen Programmes mit der Dokumentation begonnen werden konnte.

1.2 Kommunikation

Nun werden die Kommunikation im Team sowie die Planung erläutert. Da die Stundenpläne im Team unterschiedlich waren, war es schwierig, einen gemeinsamen Termin für die Treffen zu finden. Deswegen wurde neben einigen gemeinsamen Treffen vor allem über das Internet oder Telefon kommuniziert. Zur Planung und Absprache waren gemeinsame Treffen nötig, vor allem gegen Anfang des Projektes. Diese fanden meist freitags nach der BTS-Vorlesung statt. Beim ersten Treffen wurden die Herangehensweise an das Projekt sowie die wichtigsten weiteren Eckpfeiler festgelegt. Beispielsweise wurde sich bereits auf die Programmiersprache Java geeinigt. Es wurde ein Dokument zur gemeinsamen Abstimmung in GoogleDocs erstellt. In den weiteren Treffen wurden die zu erledigenden Aufgaben ermittelt und auf die entsprechenden Gruppenmitglieder verteilt. Das bereits erwähnte Dokument in GoogleDocs hatte verschiedene Zwecke. Die zu erledigenden Aufgaben und ihre Zuweisung zu den Gruppenmitgliedern waren hier festgehalten. Später konnte man hier auch den Fortschritt bei der Programmierung eintragen. Weiterhin konnte das Dokument wichtige Nachrichten oder Erinnerungen an die anderen Mitglieder enthalten, wie etwa der Termin des nächsten Gruppentreffens. Ein kleiner Nachteil hierbei ist, dass Aktualisierungen nicht per Mail etc. mitgeteilt werden, man muss dieses Dokument also von Zeit zu Zeit öffnen, um neue Informationen nicht zu verpassen. Deshalb wurde auch per Email kommuniziert, was ein zuverlässigeres und, da ohne manuelle Aktualisierung, schnelleres Medium ist. Hier ging es vor allem um wichtige Nachrichten, die schnell den Empfänger erreichen sollten, wie kurzfristige Änderungen und Nachrichten. Im späteren Verlauf des Projektes wurden auf diesem Wege auch Daten ausgetauscht, beispielsweise das neueste TeX-Dokument zur Revision. Oft gab es die Situation, dass mehrere Gruppenmitglieder gemeinsam und gleichzeitig an einem Teil des Projektes arbeiteten, Hier wurde dann über Instant Messaging wie ICQ und Skype kommuniziert. Bei der Programmierung wurde ein Versionierungssystem eingesetzt, die Kommunikation erfolgte hier zusätzlich zu den erwähnten Medien über die Kommentare. Besonders oft benötigte Informationen wurden auch im Wiki von GoogleCode eingetragen.

1.3 Benutzte Werkzeuge

Bei der Vervollständigung dieses Projektes wurden verschiedenste Softwares und Standards eingesetzt. Diese werden im Folgenden erläutert. Zuerst wird auf die Programmierung, danach auf die Erstellung der Dokumentation und Präsentation eingegangen.

Programmierung

Java Beim ersten Treffen wurde abgewägt, welche Programmiersprache verwendet werden sollte. Schnell wurde sich auf Java geeinigt. Java ist eine sehr leistungsfähige objektorientierte Programmiersprache, die sich in der Syntax an C++ orientiert und sehr weit verbreitet ist. Ein großer Vorteil von Java ist die Systemunabhängigkeit. Java kann, falls die Programmumgebung installiert ist, auf jedem beliebigen Betriebssystem ausgeführt werden (Quelle: http://java.sun.com/docs/overviews/java/java-overview-1.html). Es gibt sogar die Möglichkeit, ein Java Applet zu erstellen, welches dann mit demselben Funktionsumfang in einem entsprechend ausgestattetem Browser ausgeführt werden kann. Alle Teammitglieder hatten bereits Vorkenntnisse in Java, was die Einarbeitung und die Dokumentation verkürzte. Ein weiterer Vorteil ist das Vorhandensein einer leistungsstarken Entwicklungsumgebung, die zudem kostenlos verfügbar ist: Eclipse. Will man beispielsweise eine App für eine Apple-Gerät erstellen, muss man Mitglied im Apple Developer Program werden, was 99 Dollar im Jahr kostet **??**. Die derzeitige Version von Java ist Version 6.0.26 und wurde am 7. Juni 2011 veröffentlicht **??**.

Eclipse Eclipse ist eine der beliebtesten Plattformen zu Erstellung von Java-Anwendungen. Wie bereits erwähnt ist es kostenlos verfügbar und quelloffen und wird von der Eclipse Foundation betrieben. Eclipse wurde ursprünglich im Jahre 2001 von IBM initiiert und von einer Gruppe von Softwareverkäufern unterstützt. Später wurde dann die nicht profitorientierte Eclipse Foundation gegründet. Die Foundation wird heute weiterhin durch verschiedene Organisationen getragen und lebt von der Weiterentwicklung durch die Community Eclipse ist leicht durch Plug-Ins erweiterbar und kann, wenn ursprünglich nur für Java verfügbar, auch für andere Programmiersprachen (beispielsweise zur Entwicklung von Android-Apps) genutzt werden **??**. Die Projektgruppe verwendete die Version Helios Service Release 2. Mit SVN Kit wurde ein zusätzliches Plug-In installiert, welches noch näher erläutert wird.

Google Code Eine Vorgabe für das Projekt war, unsere Ergebnisse nach deren Erstellung auf Google Code zu veröffentlichen. Google Code ist eine kostenlose Plattform für Open Source-Entwickler, die seit ca. 2005 online ist. Die Programmierer werden durch verschiedene Funktionen bei der Erstellung, Verwaltung und Veröffentlichung von Programmen und deren Quellcode unterstützt. Man kann seinen Code unter
verschiedenen Lizenzen veröffentlichen. Google Code bietet die Versionierungsverwaltung SVN an, die mit verschiedenen Programmierumgebungen verknüpft werden kann, auch mit dem von uns verwendeten Eclipse. Nützlich ist auch das integrierte Wiki für die Kommunikation im Team. Das Code Projekt zeichnet sich durch seine gute Verfügbarkeit (es wird nur ein Computer mit Internetverbindung benötigt), seine Skalierbarkeit und seinen großen Funktionsumfang aus. **?? ??**

SVN Kit Zum gemeinsamen Bearbeiten und Sichten des Quellcodes wurde das Eclipse-Plug-In SVN Kit installiert. SVN Kit ist eine Versionsverwaltungssoftware, die rein auf Java spezialisiert ist **??**. Sie nutzt das Protokoll Apache Subversion(SVN), eine freie und offene Versions- und Dateiverwaltungssoftware **??**. Die neueste Version, welche auch beim Projekt zum Einsatz kam, ist 1.3.5.

LaTeX Zur Erstellung der Dokumentation und der Präsentation wurde LaTeX verwendet, eine Umgebung zum Textsetzen, mit dem vor allem wissenschaftliche Publikationen, aber auch Bücher und vieles weitere erstellt werden können. Es basiert auf TeX, welches in den Siebzigern vom Informatiker Donald Knuth erstellt wurde, der damit seine eigenen Bücher setzte. LaTeX ist eine Sammlung von Eingaben für TeX und erleichtert damit dessen Bedienung **??**. LaTeX kann beliebig erweitert werden, man kann beispielsweise neue Dokumentenklassen einführen oder zusätzliche Schriftarten und Schriftsätze. In unserer Gruppe wurden die beiden Klassen Beamer für Präsentationen und Lecture Notes in Computer Science vom Springer-Verlag für die Dokumentation verwendet.

MiKTeX MiKTeX bietet eine vollständige kostenlose Umgebung zur Erstellung von LaTeX-Dokumenten. Es besteht aus verschiedenen Komponenten: Einem Editor, standardmäßig TeXworks; sowie verschiedenen Programmen für Einstellungen und die Verwaltung von Paketen. Benötigt ein Dokument zusätzliche Pakete, können diese beim Setzen in TeXworks automatisch nachgeladen werden. Klassen wie die Lecture Notes in Computer Science müssen allerdings von Hand nachgeladen werden. Die verwendete Version beträgt 2.9. Nun werden kurz die beiden verwendeten Klassen vorgestellt: Klasse Beamer Die Klasse Beamer wurde speziell zur Erstellung von Präsentationen entwickelt. Sie enthält viele nützliche Werkzeuge, mit welchen, LaTeX-Erfahrung vorausgesetzt, schnell ansprechende Präsentationen erstellt werden können. Es gibt viele Vorlagen, welche das Arbeiten weiter erleichtern. KlasseLecture Notes in Computer Science Diese Klasse wurde vom Springer Verlag entwickelt, der diese auch für wissenschaftliche Veröffentlichungen nutzt. Sie ist international sehr weit verbreitet, und gut vom Herausgeber dokumentiert. **??**

2 Architektur

Auf den folgenden Seiten werden wir die Architektur des Scheduling Simulators beschreiben. Es folgt nun eine kurze Beschreibung der Architektur, sowie eine Erläuterung warum wir uns für diese entschieden haben.

2.1 Aufbau: Model-View-Controller-Prinzip

Das MVC - Prinzip ist ein Architekturmuster, welches die Software-Entwicklung in drei Einheiten unterteilt:

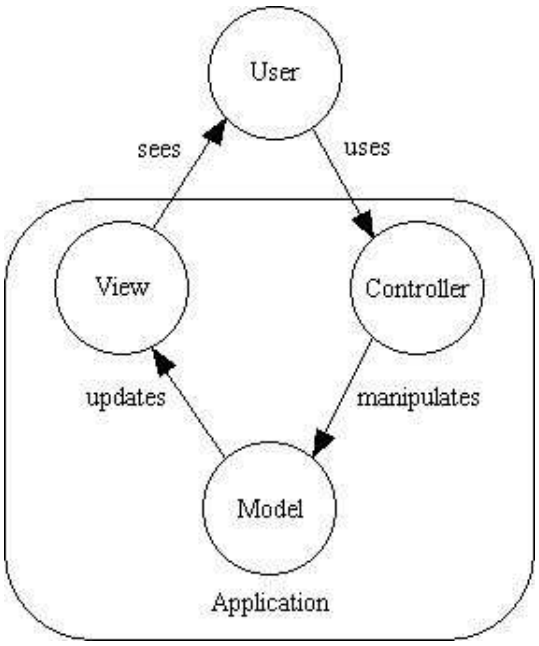

Abbildung 1. MVC-Prinzip 1

- Das Datenmodell (Model), welches die darzustellenden Daten enthält und ggf. die Geschäftslogik. Das Datenmodell ist von der Steuerung sowie der Präsentation unabhängig.
- Die Präsentation (View), welche für die Darstellung der benötigten Daten verantwortlich, sowie die Entgegennahme von Benutzerinteraktionen verantwortlich ist.
- Die Steuerung (Controller), welche die Präsentation verwaltet, Benutzeraktionen von ihr entgegen nimmt und diese anschließend auswertet und entsprechend agiert.

Wir haben uns für das Model-View-Controller-Prinzip wie in Abbildung 1 entschieden, da es uns die Möglichkeit bot, die verschiedenen Komponenten des Scheduling Simulators zu trennen. Des Weiteren behielten wir so eine bessere Übersicht und konnten die verschiedenen Komponenten getrennt voneinander implementieren. Ein weiterer Vorteil war, dass wir die unterschiedlichen Programmteile innerhalb der Gruppe aufteilen konnten.

2.2 Funktion

Nun wird die Funktionsweise des Scheduling Simulators erläutert:

Die Programmstruktur haben wir nach dem Model - View - Controller - Prinzip in 3 Packages unterteilt:

Im Model Package sind alle Verfahren in eigenen Klassen sowie eine Klasse "MyThread" untergebracht. Die verschiedenen Verfahren rufen die Klasse MyThread auf und erstellen die benötigten Threads. Werden also 10 Threads benötigt und als gewünschtes Verfahren wurde "FirstComeFirstServed" ausgewählt, so ruft die Klasse FirstComeFirstServed die Klasse My Thread 10 mal auf und erstellt die Threads. Die einzelnen Verfahren werden durch den Swing - Worker gestartet, damit sich die graphische Oberfläche nicht aufhängt.

Im View Package ist die graphische Oberfläche realisiert. Die Klasse "SchedulingSimulatorViewer" ist ein eigener Thread. Die View erhält vom Controller Parameter, welche sie für die Zeichnung der graphischen Oberfläche benötigt. Die farbige Darstellung der laufenden Threads, werden mit der Java Graphics2D Bibliothek gezeichnet.

Im Controller Package ist Klasse "SchedulingSimulatorController" implementiert. Diese erstellt den Thread für die graphische Oberfläche. Des Weiteren vermittelt die Klasse SchedulingSimulatorController zwischen der Oberfläche und den Verfahren.

Code Es folgt nun ein kleiner Code - Ausschnitt aus unserer "SchedulingSimulatorController" Klasse:

```
public void play(SchedulingVerfahren sv) {
// diese Methode startet das
     if(sv instanceof FirstComeFirstServed)
// Scheduling verfahren
     scheduler = (FirstComeFirstServed)sv;
// es wird geprüft, welches
     else if(sv instanceof LongestJobFirst)
// Verfahren übergeben wurde
     scheduler = (LongestJobFirst)sv;
     else if(sv instanceof LongestRemainingTimeFirst)
     scheduler = (LongestRemainingTimeFirst)sv;
     else if(sv instanceof PrioritaetgesteuertesScheduling)
     scheduler = (PrioritaetgesteuertesScheduling)sv;
     else if(sv instanceof RoundRobin)
     scheduler = (RoundRobin)sv;
     else if(sv instanceof ShortestJobFirst)
     scheduler = (ShortestJobFirst)sv;
     else if(sv instanceof ShortestRemainingTimeFirst)
     scheduler = (ShortestRemainingTimeFirst)sv;

     scheduler.erzeugeThreads();      // hier werden die Threads
     scheduler.verwalteThreads();             // hier werden die erzeugte
```

```
            // Threads verwaltetet
    }
```

Ein weiteres Code-Beispiel: (Swing - Worker):

```
new SwingWorker<Object, String>() {  // mit Swing - Worker werden die farbigen Balken
// der verschiedenen Threads gezeichnet
    @Override
    protected Object doInBackground() throws Exception {   //alles was im Hintergrund
// geschiet
    drawPanel.reset();
    button.setEnabled(false); // alle Knöpfe deaktivieren
    anzahlThreads.setEnabled(false);
    verfahrenAuswahl.setEnabled(false);
    int cpuLaufZeit[] = dialog.getLaufzeit();
    farbe = new Color[Integer.valueOf(anzahlThreads.getText())];
    for(int c = 0; c < Integer.valueOf(anzahlThreads.getText()); c++) {
    farbe[c] = randomColor();
    }

    Object sv =verfahrenAuswahl.getSelectedItem();

   if(sv == "FirstComeFirstServed")
{controller.play(new FirstComeFirstServed(Integer.valueOf //Zeile wegen Formatierung getrenn
(anzahlThreads.getText()),cpuLaufZeit, controller));
   }

Hier würden die weiteren Verfahren folgen.
```

Oberfläche des Programms Die Oberfläche des Programmes ist in Abbildung 2 dargestellt. Im Hauptfenster kann zuerst ausgewählt werden, wieviele Threads insgesamt laufen sollen und welches Schedulingverfahren verwendet werden soll. Nach einem Klick auf Start wird ein weiteres Fenster geöffnet, in dem die Laufzeit, die Ankunftszeit und, falls erforderlich, die Priorität angegeben werden kann. Mit einem Klick auf OK gelangt man zurück zum Hauptfenster. Hier werden dann die Threads mit jeweils eigener zufällig generierter Farbe dargestellt.

3 Fazit

Abschließend können wir sagen, dass uns die Umsetzung der Aufgabe sehr viel Spaß gemacht hat. Wir haben viele Erfahrungen in Bezug auf die Umsetzung einer Aufgabenstellung gemacht und auch mit den Umgang im Team untereinander. Es ist uns am Anfang etwas schwer gefallen uns in die Latex Sprache

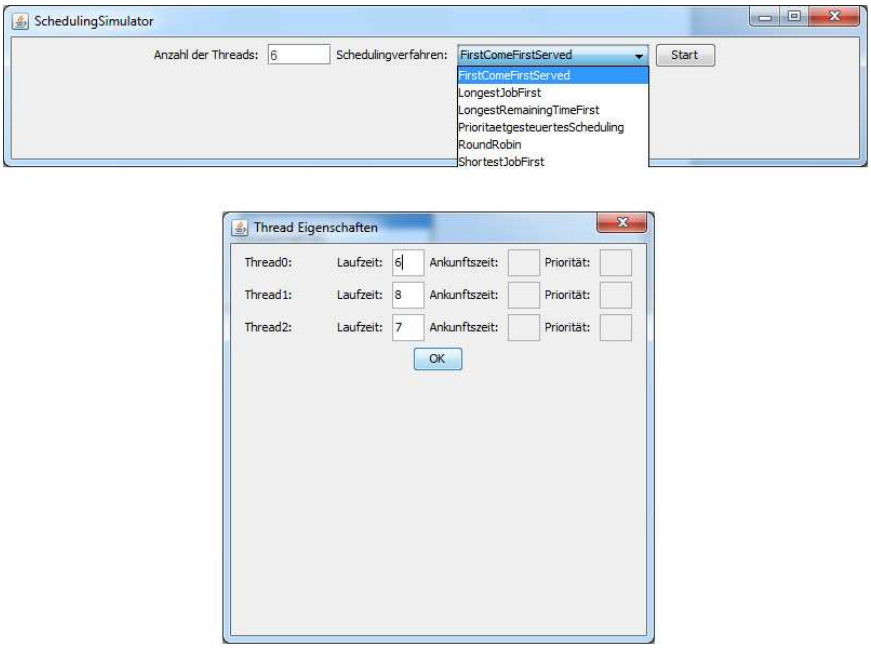

Abbildung 2. Oberfläche des Programms, Abblildung 2

einzulesen, jedoch war auch diese Erfahrung sich in etwas Unbekanntes einzuarbeiten sehr positiv. Einige von uns erwägen LaTeX auch weiterhin einzusetzen, beispielsweise zum Erstellen der Bachelorarbeit. Wir können alle mit Sicherheit sagen, dass uns diese Erfahrung auch für unser aller späteres Informatiker - Leben sehr hilfreich sein wird.

Literatur

1. Information for LNCS Authors. Springer.
   http://www.springer.com/computer/lncs?SGWID=0-164-7-72376-0
2. Subversion Homepage. Subversion
   http://subversion.tigris.org/
3. SVN Kit Homepage. SVN Kit http://svnkit.com/
4. Google Code Getting Started
   http://code.google.com/p/support/wiki/GettingStarted
5. Golem.de : Entwicklerseite Google Code vorgestellt
   http://www.golem.de/0503/37027.htm
6. About Eclipse
   http://www.eclipse.org/org/#about
7. Apple Developer Program
   http://developer.apple.com/programs/
8. Java Download
   http://www.java.com/de/download/chrome.jsp?locale=de
9. LaTeX Introduction
   http://snap.nlc.dcccd.edu/reference/latex/latex_intro.pdf