CES Softwareentwicklungspraktikum

Analyse- und Entwurfsdokument

Lena Blum, Alexander Fischer und William Hulin

Matr.-Nr. 302253, 303979 und 293858
email: [Lena.Blum|??|??]@rwth-aachen.de

Inhaltsverzeichnis

Kapitel 1

Vorwort

1.1 Aufgabenstellung und Struktur des Dokuments

Aufgabenstellung

Im Rahmen des Softwareentwicklungspraktikums (CES_SS2012) soll eine Software zur Simulation eines Stehaufkreisels erstellt werden. Die Simulation soll dabei einmal für den Fall verschwindender Reibung sowie einmal fr den Fall einer reibungsbehafteten Bewegung durchgeführt werden.
Als Programmiersprache soll C++ verwendet werden. Der Quellcode soll derart strukturiert und kommentiert sein, dass spätere Modifikationen und Erweiterungen durch weitere Mitarbeiter möglich sind.

1.2 Projektmanagement

Protoyping (MATLAB/ FORTRAN)	*Alexander*
Dokumentation	*Lena*
C++ Code:	
1. Solver+RHS	
2. Schnittstellenspezifikation	
3. GUI	
4. Exception-Handling	

1.3 Lob und Kritik

Der Niko ist voll cool

Kapitel 2

Analyse

2.1 Anforderungsanalyse

2.1.1 Benutzeranforderungen

Das von Herrn Professor Gauger gestellte Simulationsproblem umfasst die Erstellung einer Software zur Simulation eines Stehaufkreisels.

Die Simulation soll dabei einmal für den Fall verschwindender Reibung sowie einmal fr den Fall einer reibungsbehafteten Bewegung durchgeführt werden.

Im Speziellen wird ein Runge-Kutta56-Verfahren mit adaptiver Schrittweitensteuerung unter Betrachtung einer Erhaltungsgre (*conserved quantity*) zur Simulation des Problems verwendet. Die Realisierung der Simulation findet in C++ statt.

Die Bedienung sowie die Auswertung der Ergebnisse der Simulationssoftware muss durch eine grafische Benutzeroberfläche (*GUI*) möglich sein.

Die Simulationsergebnisse können in einer *ASCII*-formatierten Datei zur weiteren Verarbeitung und Auswertung exportiert werden.

Durch den modularen Aufbau ist die Wartbarkeit und spätere Modifikationen oder Erweiterungen durch externe Mitarbeiter gewährleistet.

Im weiteren umfasst der Arbeitsauftrag die Erstellung einer Benutzerdokumentation für den Endanwender.

Das Hauptproblem besteht im Lösen der Rechten Seite mithilfe des Runge-Kutta56-Verfahrens:

$$\ddot{\theta}(I + ma^2 \sin^2\theta + kma \sin\theta(R - a\cos\theta)(-\dot{x}_c \sin\phi + \dot{y}_c \cos\phi - (R - a\cos\theta)\dot{\theta}))$$
$$= \underbrace{-(I_3 - I)\dot{\phi}^2 \sin\theta\cos\theta}_{=0} - I_3\dot{\phi}\sin\theta\dot{\psi} + (g + a\dot{\theta}^2\cos\theta)(-ma\sin\theta - km(R - a\cos\theta)$$
$$(-\dot{x}_c \sin\phi + \dot{y}_c \cos\phi - (R - a\cos\theta)\dot{\theta}))$$

$$\ddot{\phi} I \sin\theta = -\underbrace{(2I - I_3)}_{=I}\dot{\phi}\dot{\theta}\cos\theta + I_3\dot{\theta}\dot{\psi}$$
$$- km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(a - R\cos\theta)(\dot{x}_c\cos\phi + \dot{y}_c\sin\phi + (a\dot{\phi} + \dot{\psi}R)\sin\theta)$$

$$\ddot{\psi} I_3 = -I_3(\ddot{\phi}\cos\theta - \dot{\phi}\dot{\theta}\sin\theta)$$
$$- km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(R\sin\theta)(\dot{x}_c\cos\phi + \dot{y}_c\sin\phi + (a\dot{\phi} + \dot{\psi}R)\sin\theta)$$

$$m\ddot{x}_c = -km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(\dot{x}_c + (a\dot{\phi} + \dot{\psi}R)\sin\theta\cos\phi + (a\cos\theta - R)\sin\phi\dot{\theta})$$

$$m\ddot{y}_c = -km(g + a\cos\theta\dot{\theta}^2 + a\sin\theta\ddot{\theta})(\dot{y}_c + (a\dot{\phi} + \dot{\psi}R)\sin\theta\cos\phi + (R - a\cos\theta)\cos\phi\dot{\theta})$$

2.1.2 Anwendungsfallanalyse

Anwendungsfälle (Statik: Anwendungsfalldiagramme; Dynamik: Aktivitätsdiagramme; Textuelle Beschreibungen laut Vorlage

Systemanforderungen

Beschreibung funktionaler und nichtfunktionaler Anforderungen basierend auf Anwendungsfällen (Ziel: Konsens zwischen Auftraggeber und Auftragnehmer)

Benutzerdokumentation

kann basierend auf den Anwendungsfällen erstellt werden; siehe Kapitel **??**

2.2 Begriffsanalyse

Identifikation von Klassenkandidaten basierend auf Anforderungen; Assoziationen zwischen Klassenkandidaten (Aggregation, Komposition, Vererbung) und Kardinalitäten

Kapitel 3

Benutzerdokumentation

wohlstrukturierte und gut lesbare Dokumentation basierend auf den Anwendungsfällen

Kapitel 4

Entwicklerdokumentation

wohlstrukturierte und gut lesbare Dokumentation der Software aus Entwicklersicht; zahlreiche Referenzen nach Kapitel **??** *und in den Quellcode in Kapitel* **??**

Die Signatur der Funktion `foo` finden Sie in Zeile **??** des Quellcodes **??** in Sektion **??**.

Anhang A

Quellcode

einfach referenzierbare Version des Quelltexts

A.1 Paket 1

A.1.1 Klasse 1.1

Listing A.1: Dokumentierter Quellcode in `klasse1.1.hpp`

```
1 class foo {
2   public:
3     int bar(const float& f);
4 }
```

A.1.2 Klasse 1.2

A.2 Paket 2

A.3 Paket 3