

CES Softwareentwicklungspraktikum

Analyse- und Entwurfsdokument



Center for Computational Engineering Science
RWTH Aachen University



Lena Blum, Alexander Fischer und William Hulin

Matr.-Nr. 302253, 303979 und 293858

email:

[lena.blum|alexander.fischer|william.hulin]@rwth-aachen.de

Inhaltsverzeichnis

1	Vorwort	2
1.1	Aufgabenstellung und Struktur des Dokuments	2
1.2	Projektmanagement	2
1.3	Lob und Kritik	2
2	Analyse	3
2.1	Anforderungsanalyse	3
2.1.1	Benutzeranforderungen	3
2.1.2	Anwendungsfallanalyse	4
3	Entwurf	5
3.1	Grobentwurf: Subsysteme	5
3.1.1	Statik	6
3.1.2	Dynamik	6
3.2	Detailentwurf: Klassen	6
3.2.1	Statik	7
3.2.2	Dynamik	7
3.3	Graphical User Interface	7
3.4	Use-Case-Diagramm	8
4	Benutzerdokumentation	10
5	Entwicklerdokumentation	11
A	Quellcode	12
A.1	Paket 1	12
A.1.1	Klasse 1.1	12
A.1.2	Klasse 1.2	12
A.2	Paket 2	12
A.3	Paket 3	12

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 Simulationssoftware muss sowohl den reibungsfreien, als auch den reibungsbehafteten Fall korrekt simulieren können.

Als Programmiersprache soll C++ verwendet werden. Der Quellcode soll derart strukturiert und kommentiert sein, dass spätere Modifikationen und Erweiterungen durch Dritte möglich sind.

1.2 Projektmanagement

Prototyping (MATLAB/ FORTRAN)	Alexander
Dokumentation	Lena
Coding:	
Parameterset, Solver, Solution, Rkv56Parset, Rkv56, DESolution, <<interface>>RightSide, RHS, Rkv56Modified	Alexander
<<interface >>OutputInterface, OutputToolbox, Main, ExceptionHandlingModule, MathException, NonCriticalME, CriticalME, ParameterException	William
GUI	Lena

1.3 Lob und Kritik

- folgt -

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 muss sowohl den reibungsbehafteten, als auch reibungsfreien Fall korrekt simulieren.

Im Speziellen wird ein Runge-Kutta56-Verfahren mit adaptiver Schrittweitensteuerung unter Betrachtung einer Erhaltungsgre (*conserved quantity*) zur Simulation des Problems verwendet.

Wahrscheinlich wird das Rkv56 Verfahren durch ein BDF-Verfahren oder eine C++ Implementierung eines speziellen Krylow-Verfahrens ersetzt.

<https://computation.llnl.gov/casc/software.html>

Die Realisierung der Simulation findet in C++ statt.

Die Bedienung sowie das Ausgeben der Simulationsergebnisse 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.

Das Kernproblem besteht im Lösen der Rechten Seite des folgenden Differentialgleichungssystems:

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

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

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

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

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

2.1.2 Anwendungsfallanalyse

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

Systemanforderungen

Dem Anwender ist es möglich die Simulationsparameter k (Reibung) sowie $\dot{\psi}$ (rad/s) über eine grafische Eingabemaske festzulegen. Wenn während der Simulation ein Fehler auftritt wird der Anwender über ein Popup-Fenster benachrichtigt. Nach Durchlauf der Simulation bekommt der Anwender die Simulationsergebnisse - $\theta, \psi, \phi, x_c, y_c, \dot{\theta}, \dot{\psi}, \phi$ - in Form von *LineCharts* in eine *GUI* eingebettet angezeigt.

Die auf der *GUI* ausgegebenen Plots können als Bilddatei exportiert werden, ebenso besteht für den Anwender die Möglichkeit, die errechneten Werte in Form von Tabellen zu speichern.

Kommt es während der Laufzeit zu einem kritischen Fehler (ein Fehler, der das korrekte Fortführen des Programmes unmöglich macht) wird der Anwender über ein Popup-Fenster benachrichtigt und das Programm beendet.

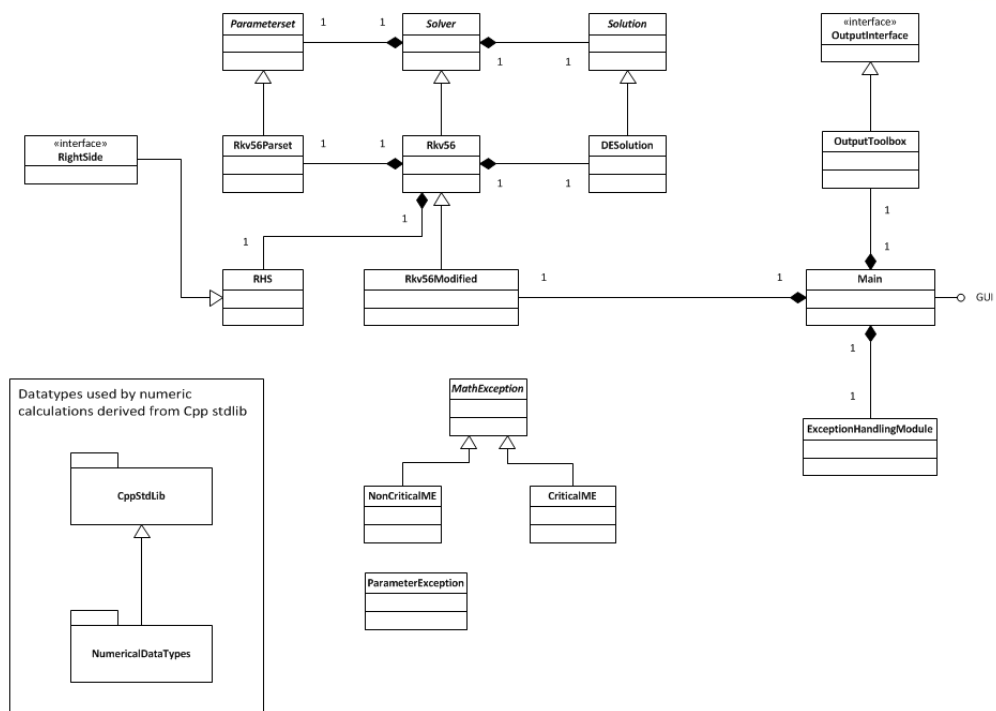
Kapitel 3

Entwurf

3.1 Grobentwurf: Subsysteme

RWTH SSELAB WS11/12 G11 V0.4

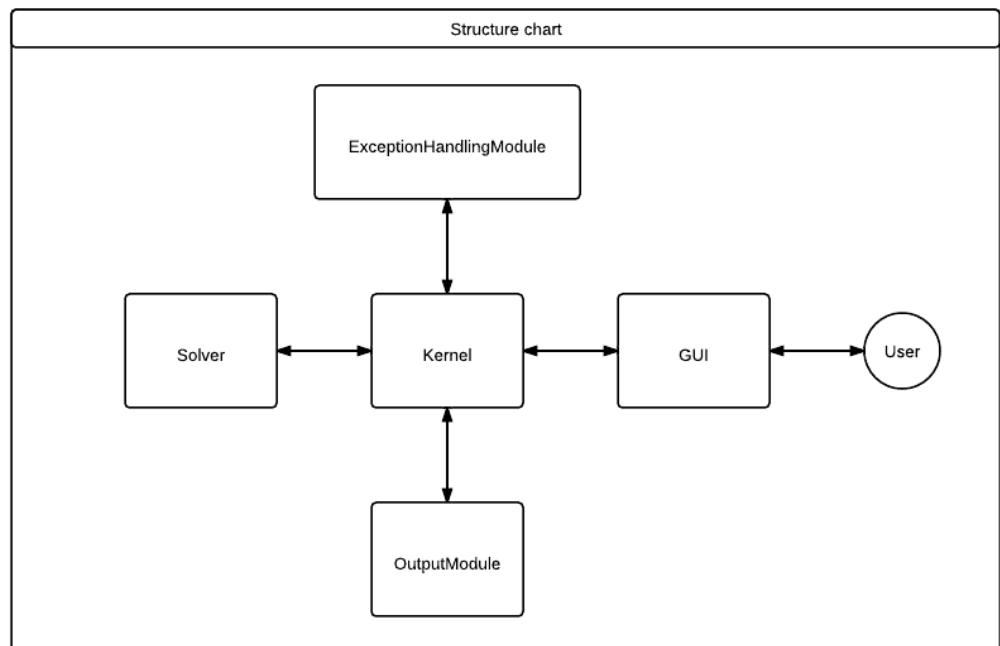
25. Januar 2012



3.1.1 Statik

3.1.2 Dynamik

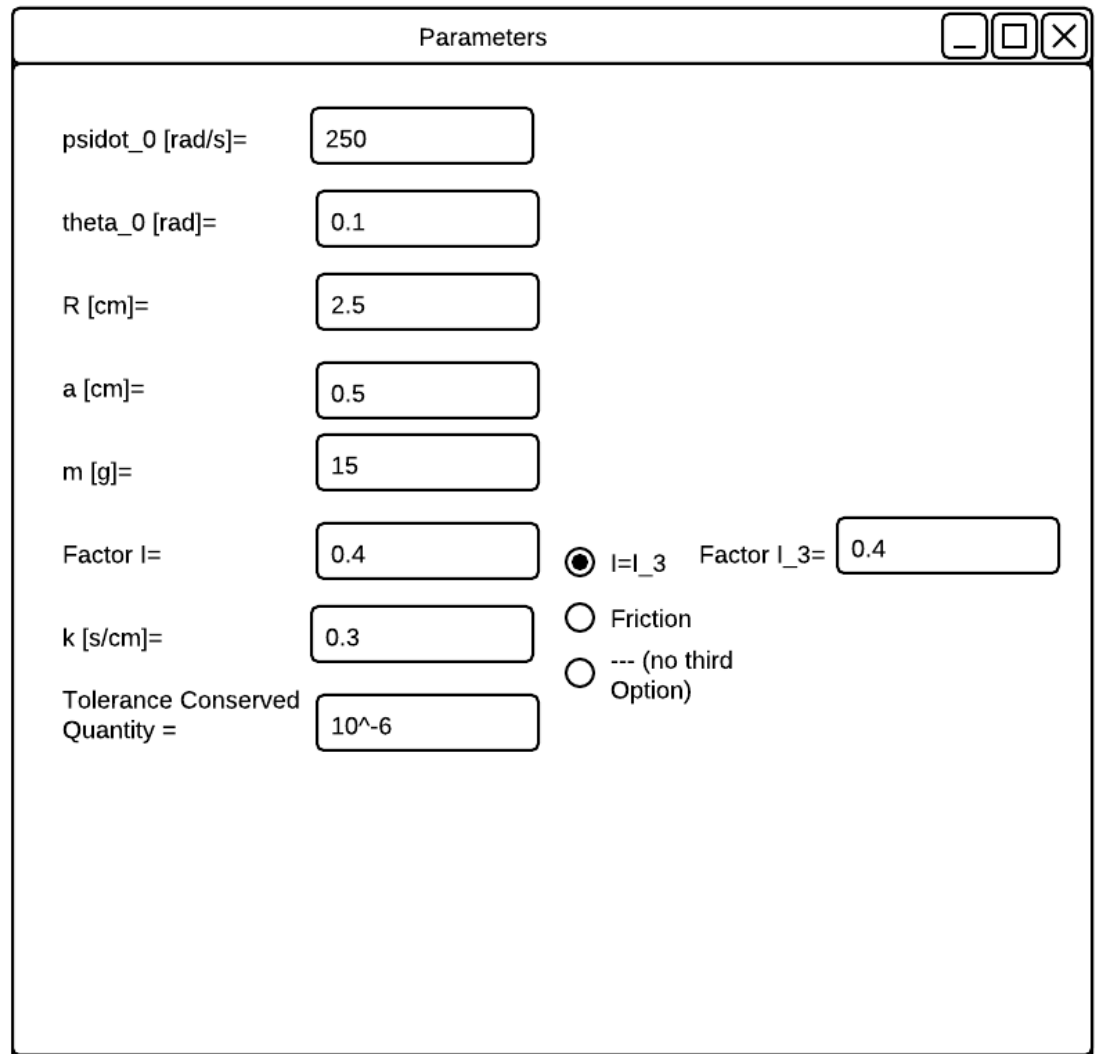
3.2 Detailentwurf: Klassen



3.2.1 Statik

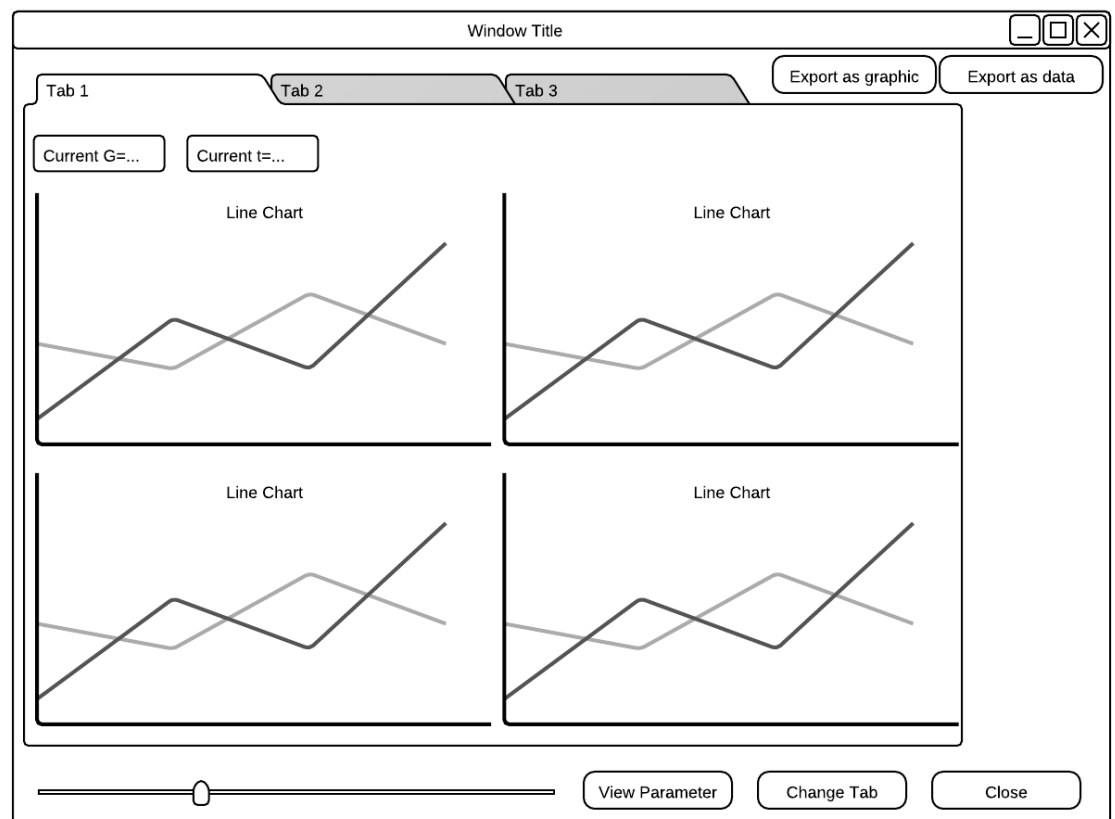
3.2.2 Dynamik

3.3 Graphical User Interface



A graphical user interface window titled "Parameters" with standard window controls (minimize, maximize, close) in the top right corner. The window contains several input fields and a group of radio buttons for parameter configuration.

Parameter	Value
$\dot{\psi}_0$ [rad/s]=	250
θ_0 [rad]=	0.1
R [cm]=	2.5
a [cm]=	0.5
m [g]=	15
Factor I=	0.4
$I=I_3$	<input checked="" type="radio"/>
Factor I_3 =	0.4
k [s/cm]=	0.3
Friction	<input type="radio"/>
--- (no third Option)	<input type="radio"/>
Tolerance Conserved Quantity =	10^{-6}



3.4 Use-Case-Diagramm

Kapitel 4

Benutzerdokumentation

- folgt -

Kapitel 5

Entwicklerdokumentation

wohlstrukturierte und gut lesbare Dokumentation der Software aus Entwickler-sicht; zahlreiche Referenzen nach Kapitel 3 und in den Quellcode in Kapitel A

Die Signatur der Funktion `foo` finden Sie in Zeile 3 des Quellcodes A.1 in Sektion A.1.1.

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