

# CES Softwareentwicklungspraktikum

Analyse- und Entwurfsdokument

Lena Blum, Alexander Fischer und William Hulin

Matr.-Nr. 302253, 303979 und 293858

email:

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

# Inhaltsverzeichnis

<b>1</b>	<b>Vorwort</b>	<b>2</b>
1.1	Aufgabenstellung und Struktur des Dokuments . . . . .	2
1.2	Projektmanagement . . . . .	2
<b>2</b>	<b>Analyse</b>	<b>3</b>
2.1	Anforderungsanalyse . . . . .	3
2.1.1	Benutzeranforderungen . . . . .	3
2.1.2	Anwendungsfallanalyse . . . . .	4
2.2	Begriffsanalyse . . . . .	4
<b>3</b>	<b>Entwurf</b>	<b>6</b>
3.1	Grobentwurf: Subsysteme . . . . .	6
3.2	Detailentwurf: Klassen . . . . .	7
3.2.1	Statik . . . . .	9
3.2.2	Dynamik . . . . .	9
3.3	Graphical User Interface . . . . .	9
3.4	Use-Case-Diagramm . . . . .	11
<b>4</b>	<b>Benutzerdokumentation</b>	<b>12</b>
<b>5</b>	<b>Entwicklerdokumentation</b>	<b>13</b>
<b>A</b>	<b>Quellcode</b>	<b>14</b>

# 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

Protoyping (MATLAB/ FORTRAN)	Alexander
Dokumentation	Lena
Coding:	
Parameterset, Solver, Solution, Rkv56Parset, Rkv56, DESolution,«interface»RightSide, RHS, Rkv56Modified	Alexander
«interface »Out putInterface, Out putToolbox, Main, ExceptionHandlingModule, MathException, NonCriticalME, CriticalME, ParameterException	William
GUI	Lena

# 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 Erhaltungsgröße (*conserved quantity*) zur Simulation des Problems verwendet.

Das Rkv56 Verfahren wurde durch ein StepperDopr853-Verfahren ersetzt, um eine höhere Genauigkeit zu erreichen.

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 einfache Erweiterbarkeit der Software durch Dritte gewährleistet.

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

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

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

### 2.1.2 Anwendungsfallanalyse

#### Systemanforderungen

##### Funktionale Anforderungen

Dem Anwender ist es möglich die Simulationsparameter  $k$  (Reibung) sowie  $\dot{\psi}(\text{rad/s}), \theta(\text{rad}), R(\text{cm}), a(\text{cm}), m(\text{g})$  und die Toleranz der Erhaltungsgroße 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 -  $\theta, \psi, \phi, x_c, y_c, \dot{\theta}, \dot{\psi}, \dot{\phi}$  - in Form von *LineCharts* in eine *GUI* eingebettet angezeigt.

Die auf der *GUI* ausgegebenen Plots können als Bilddatei oder im Tecplotformat exportiert werden.

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 an die Stelle zurückgesetzt, an der der Fehler auftrat.

##### Nicht-Funktionale Anforderungen

Die Exportfunktion der Simulationssoftware schreibt Tecplot konforme ASCII-kodierte Ausgabedateien. Vormalig exportierte Dateien können wieder importiert und geplottet werden. Ebenso können ältere Parameterkonfigurationen importiert werden. Der Nutzer kann sich eine Statistik über das Verhalten der Erhaltungsgröße und der Kontrollgrößen zur Schrittweitensteuerung ausgeben lassen.

#### Benutzerdokumentation

## 2.2 Begriffsanalyse

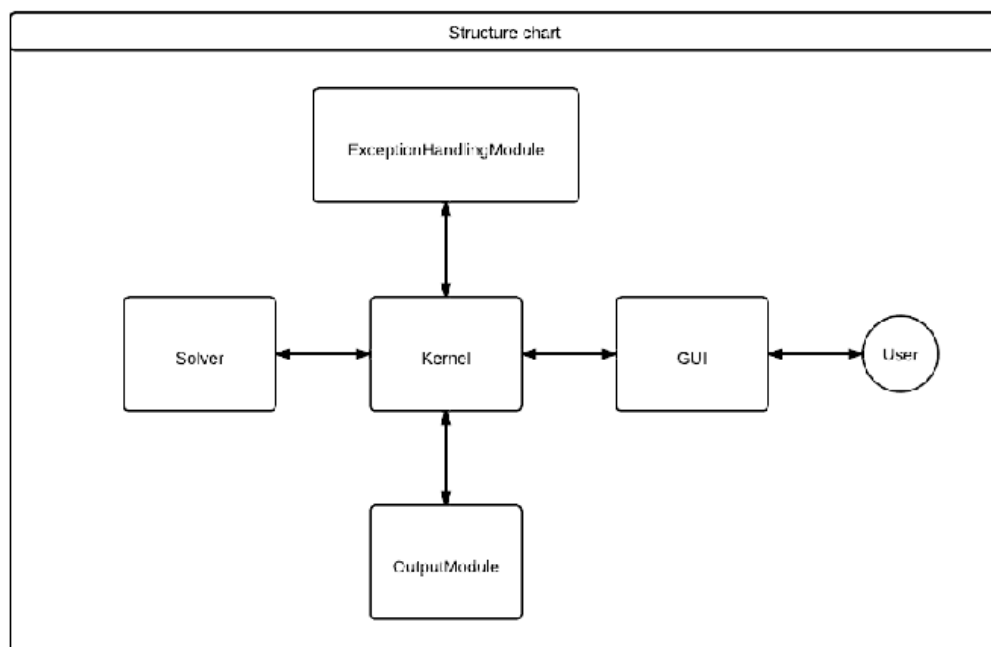
- LineChart - Zwei Achsen Diagramm mit Kartesischem Koordinatensystem. Die einzelnen Datenpunkte sind durch gerade Linien verbunden.

- GUI- *Eine grafische Benutzeroberfläche (GBO oder GUI) ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt.*

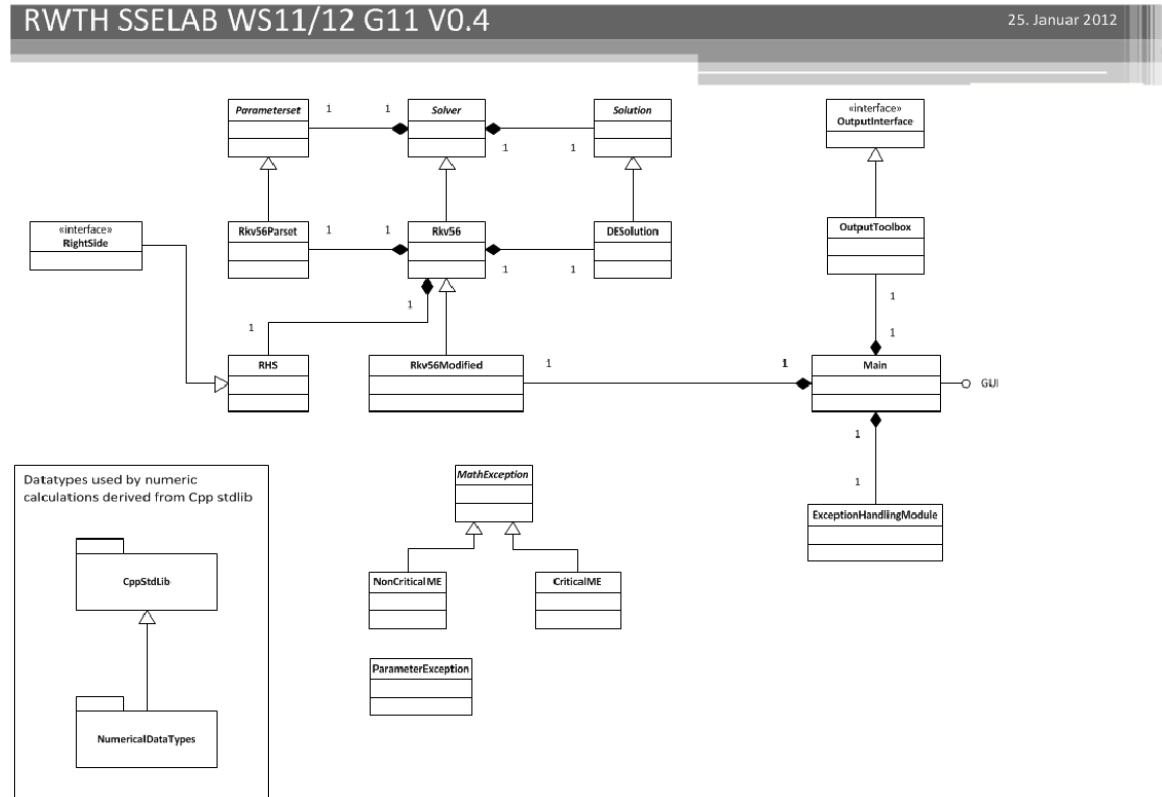
# Kapitel 3

## Entwurf

### 3.1 Grobentwurf: Subsysteme



## 3.2 Detailentwurf: Klassen



### StepperDopr853 - Dormand-Prince 853 method

Entwicklungsschritte vom Prototypen zum fertigen Löser

Um Referenzdaten erzeugen zu können und frühzeitig mathematische Fehler ausschließen zu können haben wir uns für die Implementierung eines Prototypen entschieden. Nach der ersten Implementierung eines rkv56 Verfahrens in Matlab entschieden wir uns, zu Gunsten einer höheren Genauigkeit und Geschwindigkeit, weiter Implementierungen in Fortran95 zu programmieren. Der fertige Fortran Prototyp, ebenfalls ein Runge-Kutta 56 mit adaptiver Schrittweitensteuerung, benötigte für die Lösung des TippeTop Problems<sup>1</sup> Lösung des Prototypen für  $\psi$  in  $rot/s$

Auf Grund der Erfahrung mit dem Prototypen entschieden wir uns für die Verwendung eines Dopr853 Verfahrens. Die erste Implementierung unter Verwendung des Datentyps double (auf 15 Nachkommastellen genau) lieferte leider signifikant falsche Ergebnisse

<sup>1</sup> $k = 0.3, h_{min} = 1E - 8, h_{max} = 1E - 6, rtol = atol = 1E - 4, y0 = (/0.0, 0.0, 250.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0/), T = [0, 2.75], dG < 1E - 63160922 Einzelschritte.$



Die korrekte Lösung der rkf56 Fortran95 Implementierung ist rot dargestellt, die falsche Lösung des Dopr853\_double Algorithmus in blau. Es lag nahe das diese Unterschiede in der Lösung auf Ungenauigkeiten in der Auswertung der steifen rechten Seite und den Berechnungen des Dopr853 Algorithmus zurückzuführen waren. Wir entschieden uns also für die Verwendung eines genaueren Datentyps, und zwar NTL::RR aus der NTL library<sup>2</sup>

Unter verwendung des NTL::RR Datentyps kann der fertige Löser (Dopr853 in C++) die Lösung des Problems<sup>1</sup> in 852 Schritten berechnen.

Lösung für  $\psi$  (Drehgeschwindigkeit) mit Dopr853: Der Dopr853 Algorithmus ist ein Algorithmus aus der Familie der Runge-Kutta Algorithmen der Ordnung 8. Für jeden Schritt werden 12 Auswertungen der rechten Seite des DGL benötigt. Der ursprüngliche Algorithmus nutzte eine Fehlerschätzung der Ordnung 6, was sich allerdings in einigen Fällen als unzureichend herausstellte, da dieser Fehlerschätzer jeweils die letzte Auswertung nicht berücksichtigte. Hairer, Nørsett und Wanner<sup>1</sup> konstruierten Abschätzungen der fünften und dritten Ordnung, die auch den letzten Punkt berücksichtigen. Der Fehler kann also über

$$err = err_5 \frac{err_5}{\sqrt{(err_3)^2 + 0.01(err_5)^2}}$$

abgeschätzt werden.

Die meiste Zeit über gilt  $err_5 \ll err_3$  und damit  $err = O(h^8)$ .

StepperDopr853 wurde als Mehrschrittverfahren mit fehlergesteuerter Schrittweltensteuerung implementiert, die neben dem geschätzten Fehler auch noch die Erhaltungsgröße

$$IR\dot{\phi} \sin^2 \theta + I_3(R \cos \theta - a)(\dot{\phi} \cos \theta + \dot{\psi}) = const =: G$$

berücksichtigt. ( $\Delta G$  pro Schritt  $< 1E-6$ ). Der Löser unterstützt sowohl eine dichte Ausgabe *dense output*, als auch die Ausgabe von n äquidistant verteilten Werten.

*Frei nach NumericalRecipes3rdEdition - Chapter 17.2.4 Dopr853 - An Eight-Order Method*

textitImplementierung nach Numerical Recipes Software 2007, "Routine Implementing an Eighth-order Runge-Kutta Method," Numerical Recipes Webnote No. 20, at <http://www.nr.com/webnotes?20><sup>3</sup>

<sup>2</sup><http://www.shoup.net/ntl/>

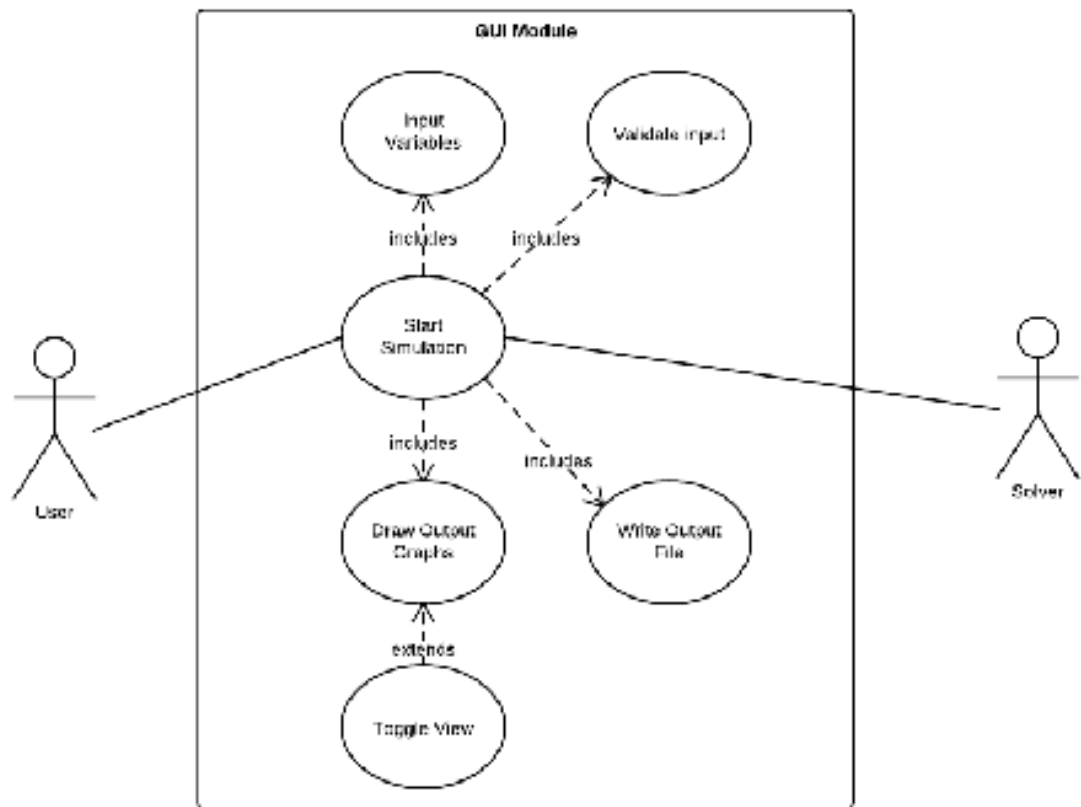
<sup>3</sup>Hairer, E., Nørsett, S.P., and Wanner, G. 1993, Solving Ordinary Differential Equations I. Nonstiff Problems, 2nd ed. (New York: Springer). Fortran codes at <http://www.unige.ch/hairer/software.html>

### 3.2.1 Statik

### 3.2.2 Dynamik

## 3.3 Graphical User Interface

## 3.4 Use-Case-Diagramm



## Kapitel 4

# Benutzerdokumentation

## Kapitel 5

# Entwicklerdokumentation

Anhang A

Quellcode