

Anwendungen der linearen Optimierung

# PROJEKTDOKUMENTATION

## **Iterator 2013**

*Autoren:*

Thomas Winter (285044)  
Florian Kurz (285154)

Vorzulegen bei: *Prof. Dr. Grütz*  
Vorzulegen am: *28.06.2013*

## Inhalt

I.	Abbildungsverzeichnis.....	3
II.	Tabellenverzeichnis.....	3
1.	Hinweis .....	4
2.	Einleitung .....	4
2.1.	Projektziele .....	4
2.2.	Rahmenbedingungen .....	4
2.3.	Projektorganisation.....	5
2.3.1.	Projektverantwortlicher / Auftraggeber: .....	5
2.3.2.	Projektmitglieder .....	5
3.	Projektphasen .....	5
3.1.	Planungsphase .....	5
3.1.1.	Programmaufbau .....	5
3.1.2.	Wireframes.....	6
3.1.3.	Komponentendiagramm .....	8
3.1.4.	Klassendiagramm .....	9
3.2.	Entwicklungsphase.....	10
3.2.1.	Verwendete Technologien.....	10
3.2.2.	Umsetzung.....	11
3.3.	Projektabschluss .....	15
4.	Ausblick und Erweiterungen .....	16
5.	Fazit .....	16

## I. Abbildungsverzeichnis

Abbildung 1: Wireframe der Benutzerschnittstelle .....	6
Abbildung 2: Komponentendiagramm.....	8
Abbildung 3: Klassendiagramm .....	9
Abbildung 4: GUI Tableaueingabe.....	11
Abbildung 5: GUI Debugdarstellung .....	11
Abbildung 6: Dependency Injection - beans.xml .....	12
Abbildung 7: ORI-Dateiformat.....	13
Abbildung 8: XML-Dateiformat.....	13

## II. Tabellenverzeichnis

Tabelle 1: Projektmitglieder .....	5
Tabelle 2: Tableaufbau .....	12
Tabelle 3: Erweiterungen.....	16

## 1. Hinweis

Alle theoretischen Grundlagen zum Simplexalgorithmus und dem Grundverständnis des Operations Research basieren auf der Lehrveranstaltung „*Operation Research*“ an der HTWG Konstanz. Auf eine detaillierte Erklärung wird in diesem Dokument mit Verweis auf die Lehrveranstaltungsunterlagen und der zugehörige Basislektüre verzichtet.

Die Funktionsweise des Programms basiert auf dem vorhandenen Iterator (1996) der HTWG Konstanz und wird deshalb in dieser Dokumentation nicht weiter erläutert.

Auf ein Glossar wird verzichtet. Grundbegriffe der Softwareentwicklung werden vorausgesetzt, weitere und spezielle Begriffe sind mittels Fußnoten definiert.

## 2. Einleitung

Im Folgenden wird die Entwicklung des „*Iterator 2013*“ dokumentiert, welcher im Rahmen der Lehrveranstaltung Anwendungen der linearen Optimierung an der HTWG Konstanz erfolgte.

### 2.1. Projektziele

Als Strategieziel wurde im Kommittent festgelegt, dass durch die Neuentwicklung des Iterators in Java für die HTWG Konstanz bis zum 28.06.2013 ...

ZIELE:

- Die Funktionalität des bestehenden Iterators, welcher unter Windows 7 nicht mehr lauffähig ist, soll durch eine Neuentwicklung (unter Windows 7 und Folgeversionen lauffähig) bis zum Ende der Projektlaufzeit funktional abgelöst werden können.
- Die Neuentwicklung soll durch einen hohen Grad (81%) an Wartungs- und Erweiterungsmöglichkeiten ausgezeichnet sein.
- Durch eine professionelle Schichtenarchitektur soll eine hohe Flexibilität im Hinblick auf neue GUI-Technologien (JavaFX) und deren Integration in den zu entwickelnden Iterator ermöglicht werden.
- Durch die Anzeigemöglichkeit eines Debugmodus soll der User Einblick in die internen Algorithmen des Programmes erhalten.
- Für eine präzisere Berechnung bei Brüchen und Gleitkommazahlen sollen die Möglichkeiten innerhalb eines 64Bit-Betriebssystems recherchiert werden.
- Durch eine ansprechende Dokumentation soll eine einfache Einarbeitung in die Programmlogik und deren Weiterentwicklung ermöglicht werden.

### 2.2. Rahmenbedingungen

- Design analog des bestehenden Iterators (*Siehe Dokumentation Iterator von 1996*<sup>1</sup>)
- Funktionsweise analog des bestehenden Iterators (*Siehe Dokumentation Iterator von 1996*)
- Lauffähigkeit unter Windows 7 (64Bit)
- Entwicklung in Java<sup>2</sup> (1.7)
- Entwicklungsumgebung Eclipse 4.2.2<sup>3</sup>
- Fertigstellung bis zum 28.06.2013 (Vorlesungsende Sommersemester 2013)

---

<sup>1</sup> Dokumentation Iterator (Version 1996) – HTWG Konstanz

<sup>2</sup> Java: Programmiersprache <http://www.java.org>

<sup>3</sup> Eclipse: Entwicklungsumgebung <http://www.eclipse.org>

## 2.3. Projektorganisation

### 2.3.1. Projektverantwortlicher / Auftraggeber:

Als Auftraggeber und Abnahmeverantwortlicher wird Herr Professor Dr. Grütz der HTWG Konstanz benannt.

### 2.3.2. Projektmitglieder

Tabelle 1: Projektmitglieder

Name	Matr.Nr.	Studiengang	Rolle
Winter, Thomas	285044	Wirtschaftsinformatik B.Sc.	Konzeption, Entwicklung GUI
Kurz, Florian	285154	Wirtschaftsinformatik B.Sc.	Konzeption, Entwicklung Backend

## 3. Projektphasen

In diesem Abschnitt sollen die einzelnen Phasen zur Erstellung des Iterators 2013 aufgezeigt werden. Die Planungsphase, in welcher die ersten Überlegungen zur Neukonzeption aufgezeigt werden sollen, wird gefolgt von der eigentlichen Entwicklungsphase und der Retrospektive anhand des Projektabschlusses.

### 3.1. Planungsphase

Der bestehende Iterator (1996) wurde in der Programmiersprache Pascal entwickelt. Eine Architektur des alten Systems ist nicht ersichtlich. Die Weiterentwicklung und vor Allem die Wartung des Iterators sind daher nur unter erschwerten Bedingungen möglich.

Für den Iterator 2013 wurde deshalb ein durchgängiges Architekturkonzept erstellt und implementiert.

#### 3.1.1. Programmaufbau

Eine zeitgemäße Schichtenarchitektur soll eine einfache Wartbarkeit und Erweiterbarkeit ermöglichen. Basis des Programmaufbaus bieten die Richtlinien der ISO25010 – SquaRE<sup>4</sup>.

#### Präsentationsschicht

Die in JavaFX realisierte Benutzeroberfläche dient der Eingabe der Tableaudaten und der Steuerung der Logikschicht.

#### Logikschicht

Kern der Applikation bildet der Berechnungsalgorithmus, welcher die Tableaus mittels Simplex Algorithmus verrechnet. Auch die Tableauverwaltung wird von dieser Schicht übernommen.

#### Persistenzschicht

Um Tableaus für die spätere Bearbeitung zu sichern und gesicherte Tableaus zu laden, wird eine eigene Schicht implementiert.

---

<sup>4</sup> ISO/IEC 25010:2011: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models  
[http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=35733](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733) (Abruf am 19.04.2013)

### 3.1.2. Wireframes

Das Benutzerinterface des Iterators 2013 basiert auf den Interfaces des bestehenden Iterators.

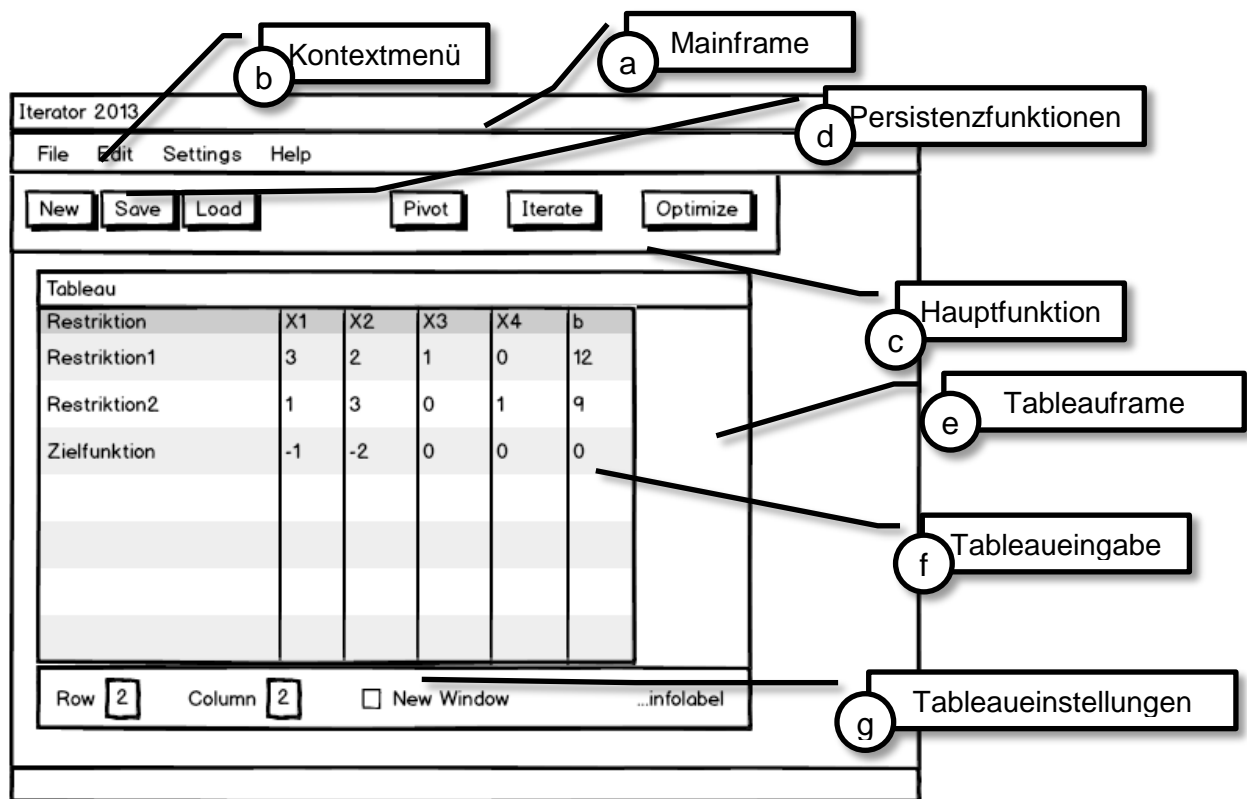


Abbildung 1: Wireframe der Benutzerschnittstelle

#### a) Mainframe

Das Mainframe bildet den Rahmen des Programms. Es enthält alle Funktionen und steuert die Tableauverwaltung.

#### b) Kontextmenü

Das Kontextmenü ermöglicht den Abruf auf alle Programmfunktionen. Es gliedert sich in folgende Funktionsgruppen:

- Datei:
  - Programmfunktionen (Beenden)
  - Tableaufunktionen (Neues Tableau)
  - Persistenzfunktionen (Speichern, Laden)
- Bearbeiten:
  - Tableaufunktionen (Pivot, Iteration, Optimieren)
- Einstellungen:
  - Einstellungsmöglichkeiten (Sprache)
- Hilfe:
  - Allgemeine Informationen
  - Dokumentation
  - Debug

**c) Hauptfunktionen**

Für eine schnelle Tableauberechnung stehen die Hauptfunktionen (Pivot, Iteration und Optimierung) zur Verfügung.

**d) Persistenzfunktionen**

Die Persistierung der Tableaus (Speichern und Laden) wird über diese Schaltflächen gesteuert.

**e) Tableauframe**

Die Tableauframes werden für jedes Tableau erstellt. Sie sind als eigenständige Frameinstanzen realisiert.

**f) Tableaueingabe**

Die Tableaueingabe ist analog der Tableaueingabe des Iterators 1996.

**g) Tableaueinstellungen**

Über die Tableaueinstellungen kann die Spalten und Zeilenanzahl des Tableaus geändert werden.

### 3.1.3. Komponentendiagramm

Anbei ein vereinfachtes<sup>5</sup> Komponentendiagramm.

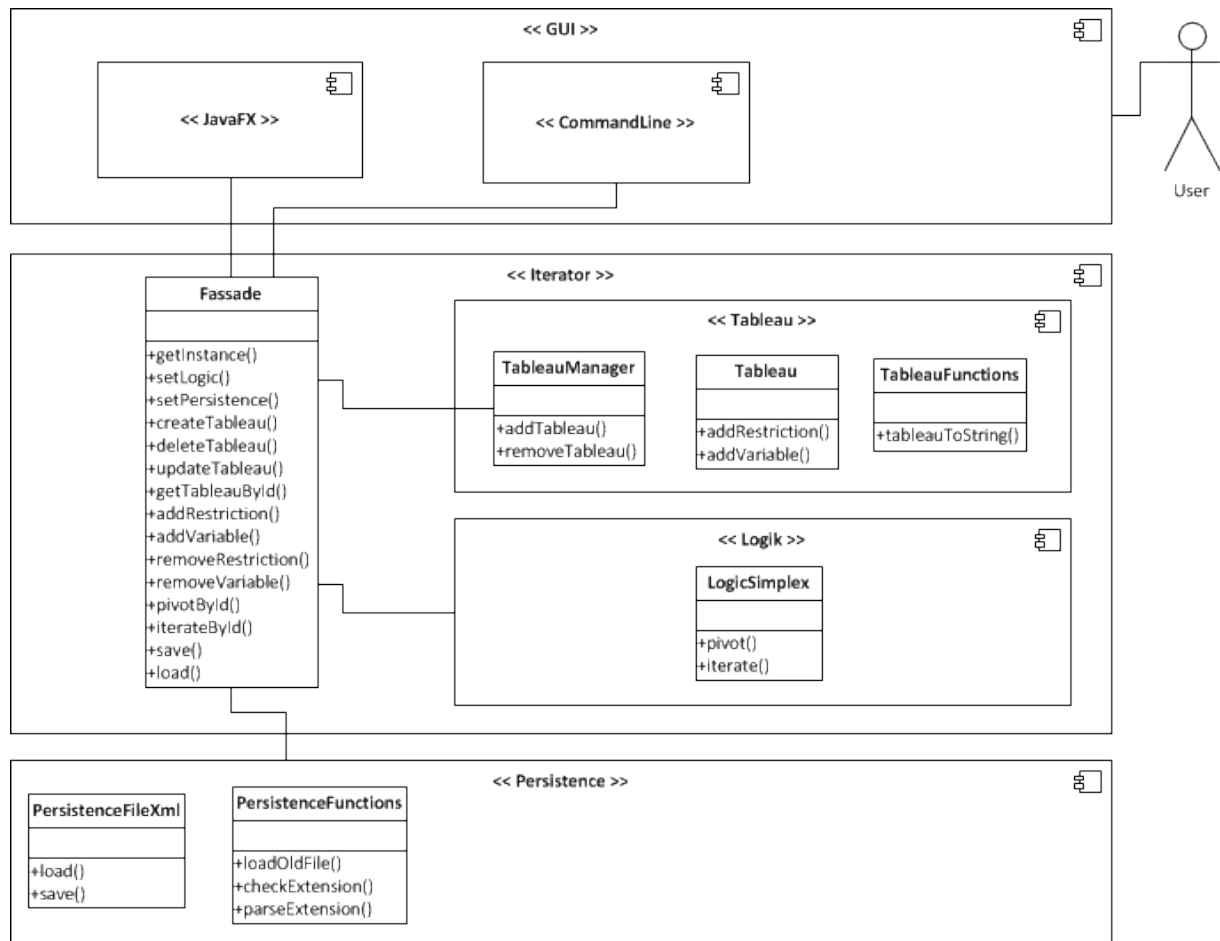


Abbildung 2: Komponentendiagramm

Herzstück bildet die Logikkomponente mit ihrer Fassade. Sie übernimmt die Steuerung aller anderen Komponenten und wird ihrerseits von der GUI angesteuert.

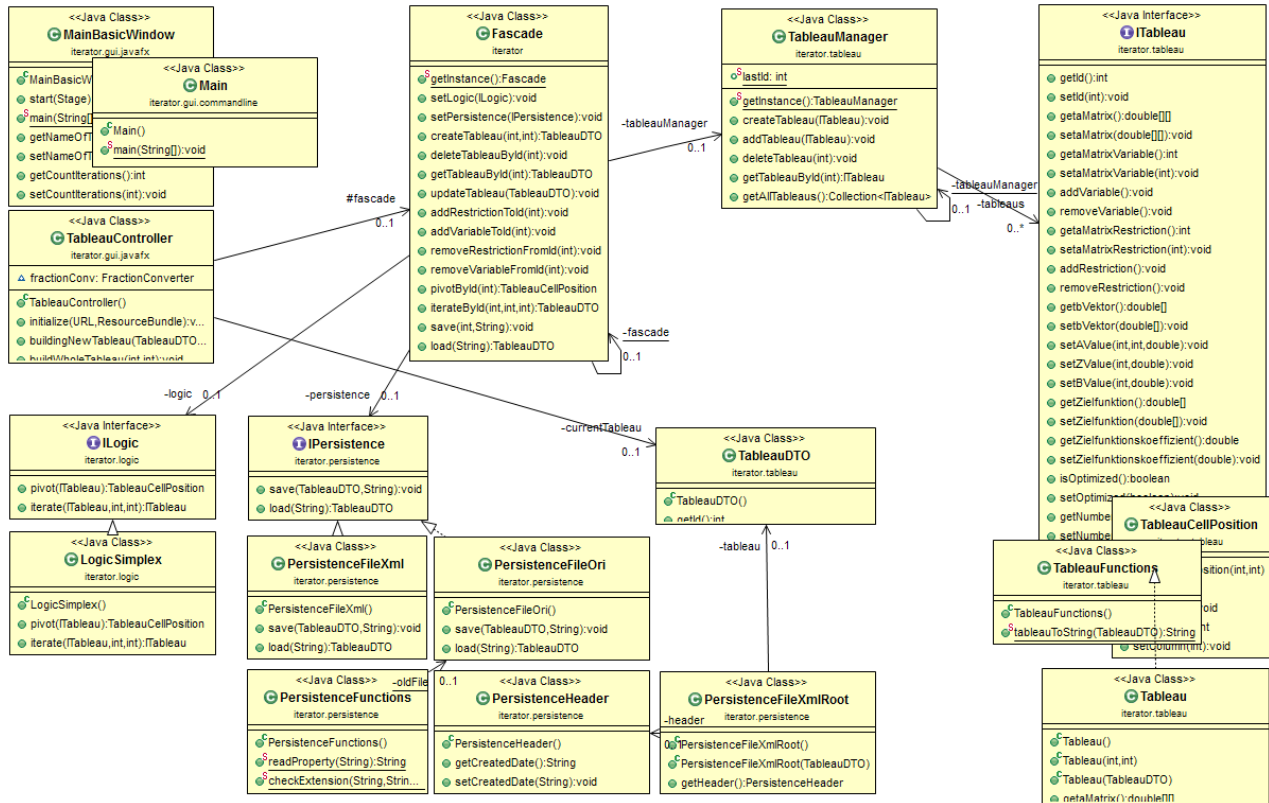
Zur vereinfachten Funktionsprüfung dient das Kommandozeilenbasierte Interface.

<sup>5</sup> Auf die Nennung aller Methoden und Hilfsklassen wurde aus Übersichtsgründen verzichtet.



### 3.1.4. Klassendiagramm

Auch das Klassendiagramm (hier stark vereinfacht dargestellt) soll einen kleinen Überblick über die verwendeten Klassen geben.



### Abbildung 3: Klassendiagramm

### 3.2. Entwicklungsphase

Nachfolgend sollen die einzelnen, in der Entwicklung verwendeten, Technologien und Methoden erläutert werden.

#### 3.2.1. Verwendete Technologien

Es folgt eine Auflistung der verwendeten Technologien. Die Implementierung erfolgte auf Basis der jeweiligen Dokumentationen (siehe angegebenen Internetadressen der jeweiligen Technologie).

#### Graphische Benutzeroberfläche:

- JavaFX 2.2.21
  - <http://www.oracle.com/technetwork/java/javafx>
- JavaFX Dialogs 0.0.3
  - <https://github.com/marcojakob/javafx-ui-sandbox/tree/master/javafx-dialogs/dist>

#### Berechnung / Fractionhandling:

- JScience 4.3
  - <http://jscience.org/>

#### Logging:

- Log4J 1.2.17
  - <http://logging.apache.org/log4j/1.x/>

#### Dependency Injection:

- Spring 3.2.2
  - <http://www.springframework.org/>

#### Objektserialisierung:

- JAXB 2.2.7
  - <https://jaxb.java.net/>

### 3.2.2. Umsetzung

#### GUI:

Die Graphische Benutzeroberfläche lehnt sich, wie bereits erwähnt, stark an den vorhandenen Iterator (1996).

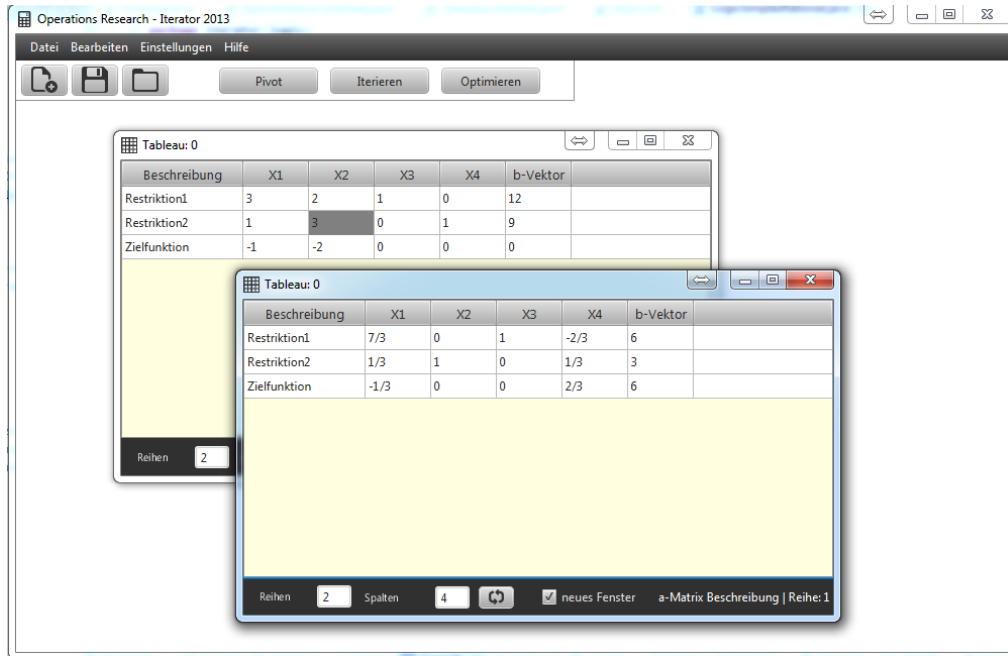


Abbildung 4: GUI Tableaueingabe

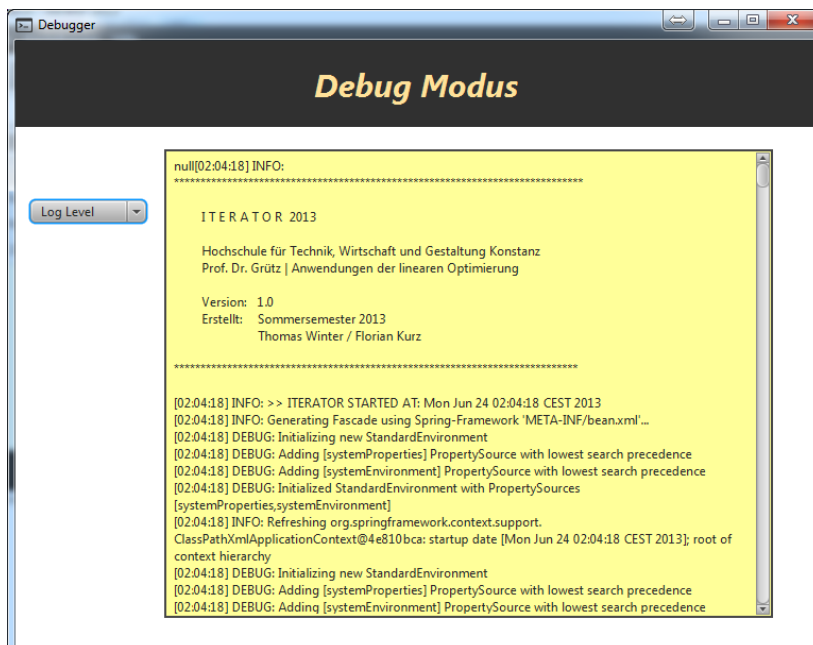


Abbildung 5: GUI Debugdarstellung

## Fassade:

Die Fassade verfügt als Herzstück der Software über alle Funktionalitäten und bildet so die Schnittstelle der GUI zu allen anderen Komponenten. Dank der Dependency Injection können die Komponenten ohne Kompilierung über eine externe Datei (WEB-INF/beans.xml) zugeordnet werden.

```

1 <beans xmlns="http://www.springframework.org/schema/beans"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
3   xmlns:context="http://www.springframework.org/schema/context"
4   xsi:schemaLocation="http://www.springframework.org/schema/beans
5     http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
6     http://www.springframework.org/schema/aop
7     http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
8     http://www.springframework.org/schema/context
9     http://www.springframework.org/schema/context/spring-context-2.5.xsd">
10
11
12   <bean id="LogicSimplexBigDecimal" class="iterator.logic.LogicSimplexBigDecimal" />
13   <bean id="LogicSimplexRational" class="iterator.Logic.LogicSimplexRational" />
14
15   <bean id="persistenceFileXml" class="iterator.persistence.PersistenceFileXml"/>
16
17   <bean id="fascade" class="iterator.Fascade">
18     <property name="logic" ref="LogicSimplexRational"></property>
19     <property name="persistence" ref="persistenceFileXml"></property>
20   </bean>
21
22 </beans>

```

Abbildung 6: Dependency Injection - beans.xml

## Tableau:

Für eine saubere Komponententrennung wird sowohl eine mit Logik versehene Tableau.java Klasse verwendet (Fassade, Logik), sowie auch ein Data-Transfer-Object welches das Tableau anhand von primitiven Datentypen realisiert.

Für ein einfaches Programmhandling wurde das Tableau intern wie folgt aufgebaut:

Tabelle 2: Tableaufaufbau

A11	A12	A13	A1...	B1
A21	A22	A...	A2...	B2
A...1	A...2	A...	A...	B...
C1	C2	C3	C...	Zielfunktionskoeffizient

A-Matrix (A):	<i>Doppeltes Array</i>
B-Vektor (B):	<i>Einfaches Array</i>
Zielfunktion (C):	<i>Einfaches Array</i>
Zielfunktionskoeffizient:	<i>Primitiver Wert</i>

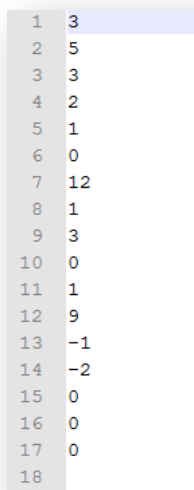
**Logik:**

Für die Berechnung anhand des Gauss'schen Algorithmus wird eine Implementierung des Interfaces „*iterator.logic.ILogic*“ benötigt.

Die aktuelle Implementierung (LogicSimplexRational.java) verwendet zur Gleitkommadarstellung die beschriebene Bibliothek (Technologie) „*JScience*“.

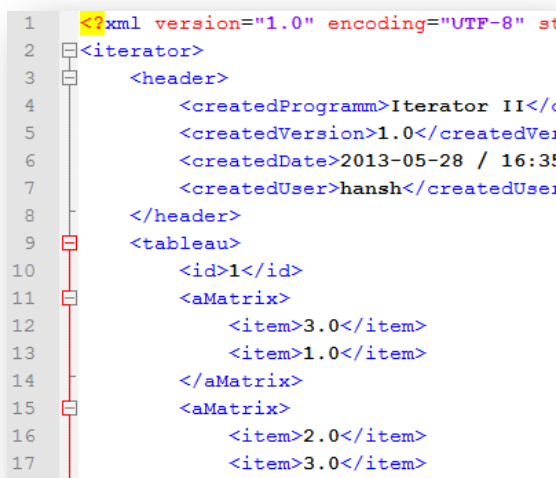
**Persistenz:**

Das Laden von Tableaueingaben ist sowohl im bestehenden ORI-Dateiformat, als auch in einem neu erstellten XML-Format möglich. Gesichert werden alle Tableaus im neuen XML-Format. Das XML-Format wurde um weitere Angaben ergänzt (Version, Datum, Benutzer, etc. ).



1	3
2	5
3	3
4	2
5	1
6	0
7	12
8	1
9	3
10	0
11	1
12	9
13	-1
14	-2
15	0
16	0
17	0
18	

Abbildung 7: ORI-Dateiformat



```
1 <?xml version="1.0" encoding="UTF-8" st
2 <iterator>
3   <header>
4     <createdProgramm>Iterator II</c
5     <createdVersion>1.0</createdVer
6     <createdDate>2013-05-28 / 16:35
7     <createdUser>hansh</createdUser
8   </header>
9   <tableau>
10    <id>1</id>
11    <aMatrix>
12      <item>3.0</item>
13      <item>1.0</item>
14    </aMatrix>
15    <aMatrix>
16      <item>2.0</item>
17      <item>3.0</item>
```

Abbildung 8: XML-Dateiformat

**Konfiguration:**

Alle Einstellungen des Iterators können, wie auch die internationalisierten Anzeigetexte und Meldungen über die Konfigurationsdateien im Ordner „config“ angepasst werden.

**Kompilierung:**

Um das Programm bei Änderungen zu einer Lauffähigen Version (ausführbares JAR-File) zu kompilieren wird das im Ordner „build“ befindliche ANT-Skript benötigt. Beim Ausführen dieses Build-Skriptes wird der Quellcode übersetzt und die erzeugte Datei in den Ordner „build/dist“ geschrieben. Sollten Änderungen an den Konfigurationsdateien vorgenommen worden sein, müssen diese ebenfalls in dem vorgesehenen Ordner „build/dist/config“ überschrieben werden.

- **WINDOWS x86 / 64-Bit**

Für die Windows-User (vor Allem für die Hochschulinternen Rechner der Laborräume) wurde eine EXE-Datei erstellt.

Die erzeugte EXE-Datei wurde mit der Software „*Lunch4J*<sup>6</sup>“ erstellt (Konfigurationsdatei im Ordner „SOURCE“). Zur Ausführung wird die kompilierte JAR-Datei verwendet.

- **Mac OSX / Linux / Sonstige**

Alle Linux-Distributionen, sowie auch Mac OSX User und sonstige Plattformen können den Iterator über die JAR-Datei ausführen.

---

<sup>6</sup> <http://launch4j.sourceforge.net/index.html>

### 3.3. Projektabschluss

Mit der Abgabe des Programms und des hier Dokumentierten Programmaufbaus zum 25.06.2013 wurden alle Ziele aus dem Kommittent erfolgreich abgehandelt.

#### Zielerreichung

Als Strategieziel wurde im Kommittent festgelegt, dass durch die Neuentwicklung des Iterators in Java für die HTWG Konstanz bis zum 28.06.2013 ...

#### ZIELE:

- ✓ Die Funktionalität des bestehenden Iterators, welcher unter Windows 7 nicht mehr lauffähig ist, soll durch eine Neuentwicklung (unter Windows 7 und Folgeversionen lauffähig) bis zum Ende der Projektlaufzeit funktional abgelöst werden können.
- ✓ Die Neuentwicklung soll durch einen hohen Grad (81%) an Wartungs- und Erweiterungsmöglichkeiten ausgezeichnet sein.
- ✓ Durch eine professionelle Schichtenarchitektur soll eine hohe Flexibilität im Hinblick auf neue GUI-Technologien (JavaFX) und deren Integration in den zu entwickelnden Iterator ermöglicht werden.
- ✓ Durch die Anzeigemöglichkeit eines Debugmodus soll der User Einblick in die internen Algorithmen des Programmes erhalten.
- ✓ Für eine präzisere Berechnung bei Brüchen und Gleitkommazahlen sollen die Möglichkeiten innerhalb eines 64Bit-Betriebssystems recherchiert werden.
- ✓ Durch eine ansprechende Dokumentation soll eine einfache Einarbeitung in die Programmlogik und deren Weiterentwicklung ermöglicht werden.

#### Weitere Funktionalität

Funktionalitäten, welche über die, im Kommittent gesetzten Ziele, hinausgehen:

- ✓ Dank der XML-basierten Persistenzschicht wurde das veraltete .ORI-Dateiformat abgelöst
- ✓ Durch das durchgängige Architekturkonzept kann das exportierte JAR-File auch in deren Applikationen, welche eine genaue Berechnung des Simplexalgorithmus benötigen implementiert werden.

Ausgehend von der Zielerreichung und den weiteren Funktionalitäten, kann das Projekt „*Iterator 2013*“ als großer Erfolg gewertet werden.

## 4. Ausblick und Erweiterungen

Folgende Punkte wären für einen weiteren Ausbau des *Iterator 2013* denkbar.

Tabelle 3: Erweiterungen

<b>Funktion</b>	<b>Beschreibung</b>
<i>Hilfefunktion</i>	Erweiterung der Hilfefunktion durch einen "geführte" Iteration. Als studentische Hilfestellung könnte das Programm mit einfachen beschriebenen Schritten eine Tableauiteration und -optimierung durchführen.
<i>Simplex-Skript</i>	Durch Implementierung einer Skripting-Funktion könnte die Simplexalgorithmik zur Laufzeit verändert und angepasst werden. Somit wären Optimierungstests zur Laufzeit möglich.
<i>GUI-Unabhängigkeit</i>	Dank der verwendeten GUI (JavaFX) könnten auch andere Endgeräte und Plattformen (z.b. Tablets, Web-basierte Systeme u.s.w) unterstützt werden.

Neben den oben aufgeführten Erweiterungen wird vor allem die Wartung und Fehlerbehebung in kommenden Projekten vereinfacht machbar sein. Aufgrund der zeitlichen Restriktion konnten nur sporadische Tests durchgeführt werden. Ein durchgängiges Testkonzept wurde nicht erstellt.

## 5. Fazit

Bereits bei der Recherche wurde klar, welche Probleme eine genaue Gleitkommazahlenberechnung bereiten würde. Dank einer ausgedehnten Vorprojekt- und Planungsphase konnten jedoch diese Probleme mit der beschriebenen externen Bibliothek gelöst werden.

Auch die innovative GUI war eine Herausforderung. Vor Allem die Usability zeigte sich zunehmend als schwer realisierbar.

Trotz des weit größeren Aufwandes für die graphische Benutzeroberfläche und des weiteren Rahmenprogramms, zeigte das Projekt, wie einfach ein Simplexalgorithmus in einer Programmiersprache wie JAVA implementiert werden kann. Dies lässt die Vermutung zu, dass die Methoden des Operation Research in betriebswirtschaftlichen Problemstellungen viel einfacher realisiert und angewandt werden können wie zu anfangs gedacht.

Somit stellt sich neben den gewonnenen Kenntnissen über die Entwicklung vor allem auch ein Lerneffekt aus der Lehrveranstaltung „Anwendungen der linearen Optimierung“ ein.