



Hochschule  
Kaiserslautern  
University of  
Applied Sciences



Hochschule Kaiserslautern  
- FACHBEREICH INFORMATIK UND MICROSYSTEMTECHNIK -

# Der Titel der Arbeit

Abschlussarbeit zur Erlangung des akademischen Grades  
Bachelor of Science (B.Sc.)

vorgelegt von

Vorname Nachname

12345

Betreuer Hochschule: Prof. Dr.

Betreuer PENTASYS: B. Sc.

# Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Dies ist ein Zitat.

verstanden, scheinen nun doch vorueber zu sein. Dies ist der Text sein.  
siehe: <http://janeden.net/die-praeambel>

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Was ist Reactive Programming</b>	<b>2</b>
2.1	Was bedeutet " <i>reaktiv</i> " im Kontext der Softwareentwicklung . . . . .	2
2.1.1	Differenzierung zwischen Reactive Programming und Reactive Systems . .	2
2.2	Reactive Programming vs. Functional Reactive Programming . . . . .	2
2.3	Reactive Programming - Ein neues Programmierparadigma? . . . . .	3
2.3.1	Was versteht man unter Programmierparadigmen . . . . .	3
2.3.2	Vergleich: Reactive Programming und Objekt orientierte Programmierung	3
2.4	Überblick über bekannte Frameworks und ihre Eigenschaften . . . . .	3
2.4.1	Reactivex.io . . . . .	3
2.4.2	Allgemeine Übersicht . . . . .	3
2.4.3	Übersicht spezielle für die Entwicklung mit Java . . . . .	3
2.5	Testen von reaktivem Code mit dem JUnit Framework . . . . .	4
<b>3</b>	<b>Einführung in Reactive Programming mit RxJava</b>	<b>5</b>
3.1	Wie funktioniert Reactive Programming . . . . .	5
3.1.1	Synchronität . . . . .	5
3.1.2	Parallelisierung . . . . .	5
3.1.3	Push vs. Pull . . . . .	5
3.2	Rx.Observable . . . . .	5
3.3	Rx.Observer . . . . .	5
3.3.1	Rx.Subscriber . . . . .	6
3.4	Operationen und Transformationen . . . . .	6
3.4.1	Exkursion: Streams API Java 8 . . . . .	6
3.4.2	Operation filter() . . . . .	6
3.4.3	Transformation map() . . . . .	6
3.4.4	Transformation flatMap() . . . . .	6
3.4.5	Operation merge() . . . . .	6
3.4.6	Operation zip() . . . . .	6

<b>4</b>	<b>Beispiel: Implementierung eines Systemmonitors</b>	<b>7</b>
4.1	API für Systemwerte . . . . .	7
4.1.1	Beschreibung Klasse 1 . . . . .	7
4.1.2	Beschreibung Klasse 2 . . . . .	7
4.1.3	Beschreibung Klasse 3 . . . . .	7
4.1.4	Beschreibung Klasse 4 . . . . .	7
4.2	Client für API: GUI zur Repräsentation der Systemwerte . . . . .	7
4.2.1	Beschreibung Klasse 1 . . . . .	7
4.2.2	Beschreibung Klasse 2 . . . . .	7
	<b>Literaturverzeichnis</b>	<b>I</b>
	<b>Abbildungsverzeichnis</b>	<b>II</b>
	<b>Tabellenverzeichnis</b>	<b>III</b>

# Kapitel 1

## Einleitung

Durch den starken Wachstum der IT entstehen immer wieder neue Möglichkeiten Anwendungen zu realisieren. Durch den technischen Fortschritt werden schon bekannte Muster und Architekturen weiter entwickelt oder Ideen für die neu entstandenen Anforderungen umgesetzt. Eine dieser Ideen ist *REACTIVE PROGRAMMING*. Vor noch nicht allzu langer Zeit waren Monolithen die auf eigens gehosteten Servern ausgerollt wurden der Stand der Dinge. Durch die mittlerweile entstandene Vielfalt an Endgeräten wie Smartphones, das Deployment in der Cloud oder die Menge an spezialisierten Programmiersprachen, Frameworks und Entwicklungswerkzeugen haben sich Anforderungen herausgestellt die sich mit bekannten Lösungen wie zum Beispiel einer objektorientierten Herangehensweise nicht zur vollen Zufriedenheit erfüllen lassen. Für einen Benutzer ist es üblich, dass Änderungen sofort sichtbar sind und angefragte Daten in Bruchteilen einer Sekunde bereit stehen, und das zu jedem Zeitpunkt. Kann eine Applikation dies nicht leisten, kann die User Experience<sup>1</sup> in Mitleidenschaft gezogen werden, was bei der Menge an Diensten gleicher Art dazu führen kann, dass Benutzer auf die Dienste von Mitbewerber zurück greifen. Um diesem vorzubeugen wurde aus vorhandenen Konzepten ein grundlegendes Manifest für *Reaktive Systeme*<sup>2</sup> erstellt. Die im Manifest angeführten Eigenschaften werden im Verlauf dieser Arbeit noch genauer aufgegriffen. Es sei nur jetzt schon gesagt, dass zur Umsetzung der Kriterien für ein Reaktives System, die Verwendung von Reactive Programming nicht notwendig ist, jedoch meist sinnvoll scheint.

Ziel dieser Arbeit ist es, eine Einführung in die Welt von Reactive Programming zu geben. Dazu wird im nächsten Kapitel eine Einordnung in das Gesamtbild der Softwareentwicklung durchgeführt. Ebenso werden die Eigenschaften und Eigenheiten von Reactive Programming beschrieben. Weiterhin gibt es einen Überblick über einige Frameworks mit deren Hilfe ein reaktives Programmieren realisiert werden kann. Darauf folgend wird eine Implementierung einer Beispielanwendung besprochen, um dem Vorangegangenen eine praktische Anwendung hinzuzufügen.

---

<sup>1</sup>TODO: User Experience Def

<sup>2</sup>[Bon14]: Reactive Manifesto 2.0. [www.reactivemanifesto.org](http://www.reactivemanifesto.org)

# Kapitel 2

## Was ist Reactive Programming

Ist mein neu in der Domäne von reaktiver Entwicklung stellt man schnell fest, dass allerhand mögliche Definitionen und Beschreibungen was Reactive Programming denn zu sein scheint. Bevor jedoch hier eine Definition erläutert wird, muss zwischen unterschiedlichen Begrifflichkeiten differenziert werden: *Reactive Systems*, *Functional Reactive Programming* und natürlich *Reactive Programming*.

### 2.1 Was bedeutet "*reactive*" im Kontext der Softwareentwicklung

Wie die Wortherkunft ja schon verlauten lässt, wird etwas als reaktiv bezeichnet, wenn eine Reaktion durch eine vorangegangene Aktion ausgelöst wird. Diese Reaktion kann über unterschiedliche Wege in Gang gebracht werden. Grundlegend für diese Interaktion ist die Beobachtung von Daten oder Ereignissen welche von außerhalb der Anwendung, zum Beispiel über die Benutzeroberfläche, oder innerhalb der Anwendung verändert oder ausgelöst werden.

#### 2.1.1 Differenzierung zwischen Reactive Programming und Reactive Systems

Das Manifest bezieht sich eigentlich auf die Systeme. Wo kommt nun RP in Spiel?

### 2.2 Reactive Programming vs. Functional Reactive Programming

FRP findet nun mal in Funktionalen Programmiersprachen statt. Java is jedoch OO und mit Java 8 und dem Rx Frameworks werden funktionale Eigenschaften in der OO Sprache eingebracht. Unterschiede müssen noch genau belegt<sup>1</sup> werden. Der grundlegende Gedanke reaktiver

---

<sup>1</sup>[Loh16]

Systeme wurde schon im Jahre 1985 in einem Paper von D. Harel und A. Pnueli beschrieben<sup>2</sup>.

## **2.3 Reactive Programming - Ein neues Programmierparadigma?**

Wie gliedert man einen Programmierstil ein? <sup>3</sup>

### **2.3.1 Was versteht man unter Programmierparadigmen**

Erklärung was sind Paradigmen und wieso werden sie definiert.

### **2.3.2 Vergleich: Reactive Programming und Objekt orientierte Programmierung**

Eventuell bessere mit Funktionaler Programmierung zu vergleichen. Muss noch genauer betrachtet werden. Soll die Unterschiede zu den gängigen, bekannten Methoden aufzeigen.

## **2.4 Überblick über bekannte Frameworks und ihre Eigenschaften**

Überblick quer über die gängigen Programmiersprachen.

### **2.4.1 Reactivex.io**

Kurze Erläuterung zu der Entstehung von Reactive Extensions

### **2.4.2 Allgemeine Übersicht**

Rx Frameworks zu den jeweiligen Sprachen. Frameworks wie z.B. Akka<sup>4</sup>.

### **2.4.3 Übersicht spezielle für die Entwicklung mit Java**

RxJava. Reactive Streams Konvention. Java 9 Api Änderung bzgl. Reactive Streams.

### **Framework für JavaFX - RxJavaFX**

Einführung und Eigenschaften erläutern

---

<sup>2</sup>[HP85]

<sup>3</sup>[Bai13]

<sup>4</sup>[Kar16]

## **2.5 Testen von reaktivem Code mit dem JUnit Framework**

Noch nichts genaues. Muss noch geschaut werden wie die Funktionalität von JUnit RP abdeckt.



# Kapitel 3

## Einführung in Reactive Programming mit RxJava

Beschreibung wieso RxJava. Beschreibung was beschrieben wird.

### 3.1 Wie funktioniert Reactive Programming

Einleitung zum Aufbau: Klassenübersicht des Frameworks mit Erklärung.

#### 3.1.1 Synchronität

Sync vs. Async - was bringt RP in dieser Hinsicht

#### 3.1.2 Parallelisierung

Concurrency vs. Parallelism - was tritt wie wann auf bzw. kann wie wann angewandt werden

#### 3.1.3 Push vs. Pull

Wichtigster Unterschied. Observable als Gegenpart zu Iterable - somit Push vs. Pull Vergleich.

### 3.2 Rx.Observable

Interface Übersicht. Nutzen und Anwendung anhand von Beispiel. Hot vs. Cold

### 3.3 Rx.Observer

Was kann Observer -> Interface Übersicht

### **3.3.1 Rx.Subscriber**

Was ist speziell am Subscriber -> Interface Übersicht

## **3.4 Operationen und Transformationen**

Erläuterung von den Stadien der Operation von Beginn über Mitte bis Ende.

### **3.4.1 Exkursion: Streams API Java 8**

Beschreibung was Streams darstellen, wie sich Observables im Vergleich verhalten

### **3.4.2 Operation filter()**

Beispiel und Perlenbild. Einsatz beschreiben

### **3.4.3 Transformation map()**

Beispiel und Perlenbild. Einsatz beschreiben

### **3.4.4 Transformation flatMap()**

Beispiel und Perlenbild. Einsatz beschreiben

### **3.4.5 Operation merge()**

Beispiel und Perlenbild. Einsatz beschreiben

### **3.4.6 Operation zip()**

Beispiel und Perlenbild. Einsatz beschreiben Eventuell noch mehr Operationen

# Kapitel 4

## Beispiel: Implementierung eines Systemmonitors

Beschreibung der Funktionen der Anwendung. Überblick über Projekt/Klassenstruktur. Verwendete Tools und Versionen.

### 4.1 API für Systemwerte

Framework für die Systemwerte kurz erläutern. Grobes Vorgehen beschreiben wie man es in etwa Umsetzen kann.

#### 4.1.1 Beschreibung Klasse 1

#### 4.1.2 Beschreibung Klasse 2

#### 4.1.3 Beschreibung Klasse 3

#### 4.1.4 Beschreibung Klasse 4

### 4.2 Client für API: GUI zur Repräsentation der Systemwerte

#### 4.2.1 Beschreibung Klasse 1

#### 4.2.2 Beschreibung Klasse 2

# Literaturverzeichnis

- [Bai13] BAINOMUGISHA, Carreton Andoni Lombide van Cutsem Tom Mostinckx Stijn Meuter Wolfgang d. Engineer: A survey on reactive programming. In: *ACM Computing Surveys* 45 (2013), Nr. 4, S. 1–34. <http://dx.doi.org/10.1145/2501654.2501666>. – DOI 10.1145/2501654.2501666. – ISSN 03600300
- [Bon14] BONÉR, Farley Dave Kuhn Roland Thompson M. Jonas: the-reactive-manifesto-2.0. (2014). <http://www.reactivemanifesto.org/>
- [HP85] In: HAREL, D. ; PNUELI, A.: *On the Development of Reactive Systems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1985. – ISBN 978–3–642–82453–1, 477–498
- [Kar16] KARNOK, Dávid: *Operator-fusion (Part 1)*. <https://akarnokd.blogspot.de/2016/03/operator-fusion-part-1.html>. Version: 2016 (Advanced Reactive Java)
- [Loh16] LOHMÜLLER, Jan C.: *Reactive Programming – Mehr als nur Streams und Lambdas*. <https://www.informatik-aktuell.de/entwicklung/programmiersprachen/reactive-programming-mehr-als-nur-streams-und-lambdas.html>. Version: 2016

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Listingverzeichnis