HOCHSCHULE DER MEDIEN

Projektbericht

Multi-Touch UI-Designer

PFLICHTPROJEKT WS 09/10

Durchführungszeitraum:	Wintersemester 2009/2010
Studiengang:	Medieninformatik (B.Sc.)
Semester:	4. Semester
Student:	Patrick Janas Pfaffenwaldring 64B 4.3 70569 Stuttgart-Vaihingen
Hochschule:	Hochschule der Medien Nobelstraße 10 70569 Stuttgart
Betreuender Dozent:	Prof. Dr. Jens-Uwe Hahn (vertreten durch Herrn Norman Pohl)
Betreuender Mitarbeiter:	Uwe Laufs (Fraunhofer IAO)

Inhalt

Einführung

Worum geht es? / Wieso gerade dieses Projekt?

Im Rahmen des Studiums Medieninformatik (B.Sc.) an der Hochschule der Medien Stuttgart sollte ein typisches Projekt praktisch umgesetzt werden, um das bisherige im Studium erworbene theoretische Wissen anzuwenden wie auch zu vertiefen. Ziel war es hierdurch wichtige Erfahrungen inhaltlicher als auch organisatorischer Art zu machen.

Für die Wahl eines Projektes boten sich zwei Optionen. Einerseits war es möglich eine eigene Idee in einem Projekt umzusetzen. Auf der anderen Seite standen eine Vielzahl hochschulinterner Projektvorschläge zur Verfügung, aus denen man wählen konnte. In der Wahl eines der Projektvorschläge sah ich den Vorteil, dass diese Art von Projekt näher an denen liegt, die ich im späteren Berufsleben antreffen würde. So entwickelt man i.d.R. nicht für sich selbst, sondern hat gewisse Vorgaben bzw. Anforderungen, die durch den Auftraggeber oder mit ihm festgelegt werden. Dadurch eröffnen sich hier ganz andere Aspekte aber auch Schwierigkeiten, die man bei der Realisierung einer eigenen Idee nicht hat. So ist häufig in realen Projekten nicht die Umsetzung der (aussagekräftigen) Spezifikation in ein Softwareprodukt ein Problem, sondern gerade die Anfertigung dieser Spezifikation, also die Analyse des Problems bzw. der Anforderungen. Oft werden die Anforderungen des Auftraggebers nicht richtig verstanden, auch wenn sich beide Seiten, Auftraggeber und Entwickler, einig zu sein scheinen, da beide meist eine unterschiedliche Sichtweise auf die Dinge haben. Auch kommt es vor, dass der Auftraggeber was anderes braucht, als das, was er in Auftrag gibt. Die nebenstehen Abbildung soll diese Thematik verdeutlichen bzw. veranschaulichen. Insbesondere aus diesem Grund habe ich mich letztlich dafür entschieden, nicht eine eigene Idee in einem Projekt zu realisieren, bei der man genau weiß, wie die Anwendung später aussehen soll, sondern ein Projekt aus der Liste der Projektvorschläge zu wählen, um auf dem obenen genannten Gebieten Erfahrungen zu sammeln.

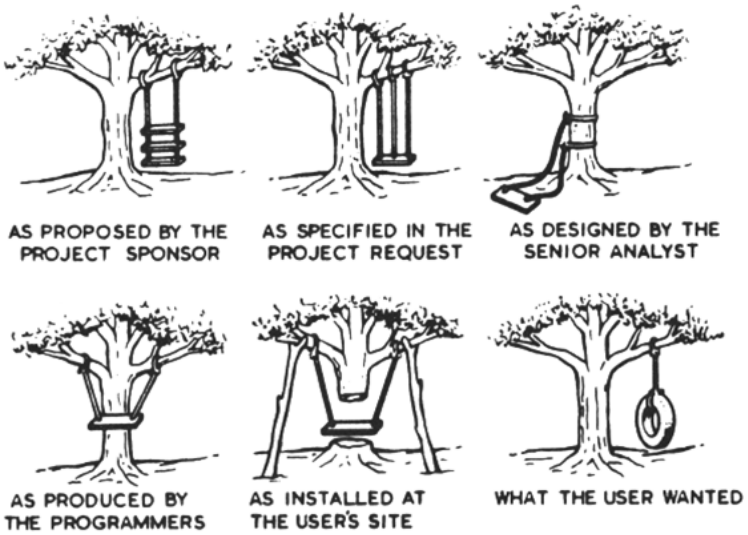


Abb. 1: Anforderungen und unterschiedliche Sichtweisen

Aus den unterschiedlichen Themenbereichen hatte ich ein interessanten Projektvorschlag im Umfeld Multi-Touch entdeckt. Da ich bisher kaum Erfahrung in diesem Bereich, speziell was die Entwicklung multi-touch-fähiger Anwendungen angeht, hatte, entschloss ich mich dazu, das Projekt „Multi-Touch UI-Designer“ zu bearbeiten. Dieses Thema wollte ich dazu nutzen, um mich mehr mit dieser Technologie zu beschäftigken und einen besseren Einblick in diese zu bekommen.

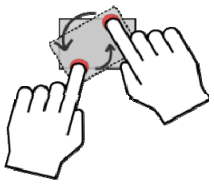


Abb. 2: Multi-Touch

Das interessante an Multi-Touch-Anwendungen ist, dass diese Technologie neue Möglichkeiten in der Mensch-Computer-Interaktion bietet. Durch die Fähigkeit des Eingabegerätes mehr also nur einen Finger zu erkennen, sind diverse Gesten möglich, um eine Anwendung zu steuern. Außerdem können somit mehrere Personen gleichzeitig an einem Multi-Touch-Display miteinander arbeiten –

also ein interessantes Themengebiet, so wie ich finde. Auch war es eine Herausforderung auf einem bereits bestehenden Framework (MT4j) aufzubauen, welches man noch nicht kennt. Da man sich oft in fremden Quellcode einarbeiten bzw. zurechtfinden muss, war dies eine weitere Gelegenheit, hierbei Erfahrungen zu sammeln.

Im nachfolgenden Abschnitt wird die Aufgabenstellung des Projektes „Multi-Touch UI-Designer“, um einen groben Einblick in das Projekt bzw. das Ziel dieses Projektes zu geben, wiedergeben. Sie entspricht einer gekürzten Fassung des Projektvorschlags für die Anmeldung als Medieninformatik-Projekt an der Hochschule der Medien Stuttgart.

Aufgabenstellung

Die Aufgabenstellung ist folgende:

> „Zur einfacheren, schnelleren und intuitiveren Erstellung von Multi-Touch-Benutzungsschnittstellen soll ein UI-Designer für die Java-basierte Multi-Touch-Entwicklungsplattform MT4j (MT4j.org) entwickelt werden, welcher beispielsweise das direkte Platzieren von Oberflächenkomponenten sowie die Zuweisung des Verhaltens der Komponenten (z.B. Auswahl der Aktion, die beim Drücken eines Buttons ausgeführt werden soll) am Multi-Touch-Screen ermöglicht.
>
> Aus den visuell durchgeführten Oberflächenkonfigurationen kann unter Verwendung einer Template-Engine Java-Quellcode generiert werden, und so die visuelle Oberflächenkonfiguration in eine lauffähige Multi-Touch-Applikation transformiert werden.“

MT4j Framework

Überblick

MT4j steht für „Multitouch for Java™“ und ist eine Open Source Java™ Entwicklungsplattform für die schnelle Entwicklung von 2D, 3D bzw. pseudo-3D Applikationen. Die Applikationen basieren dabei auf einer Szenengraph-Struktur wie sie z.B. aus Java™ Swing bekannt ist. Damit lassen sich z.B. Objekte logisch verschachteln (Parent-Child-Beziehung) bzw. gruppieren, um darauf bestimmte Operationen auszuführen.

Das besondere an diesem Framework ist, dass es bereits typische Multi-Touch-Gesten unterstützt. So sind u.a. Operationen auf Objekten wie Rotieren, Vergrößern u. Verkleinern oder das Zoomen möglich. Darüber hinaus beinhaltet es bereits diverse grafische Objekte wie

- Rechtecke, Ellipsen, Polygone, Linien, ...
- UI-Komponenten: Buttons, Textlabels, Sliders, Multi-Touch Keyboard, ...

Dies ist nur ein ganz kleiner Ausschnitt aus der Features-Liste. Für alle Features, die dieses Framework bietet bzw. unterstützt, sei auf die Internetseite www.mt4j.org verwiesen.

MT4j selbst basiert auf Processing, welches eine auf die Einsatzbereiche Grafik, Simulation und Animation spezialisierte objektorientierte, stark typisierte Programmiersprache mit zugehöriger integrierter Entwicklungsumgebung ist.

Erstellung einer Applikation

In diesem Abschnitt soll Schritt für Schritt eine „HelloWorld“-Applikation programmiert werden, um die Vorgehensweise bei der Erstellung von Anwendungen, die auf dem MT4j-Framework basieren, zu verdeutlichen. Das HelloWorld-Beispiel ist eines mehrerer Beispiele, die mit dem MT4j-Framework mitgeliefert werden und zu Demonstrationszwecken dienen. Die nebenstehende Abbildung zeigt das fertig programmierte Beispiel.

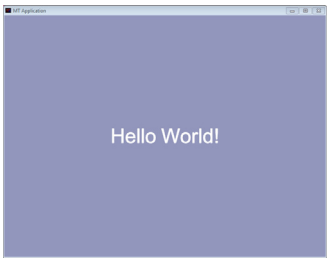


Abb. 3: HelloWorld-Beispiel

Jede MT-Applikation besteht aus einem Objekt der Klasse *MTApplication* und mindestens einem Szenen-Objekt. Um also eine MT-Applikation zu entwickeln, leitet man seine eigene Klasse von der *MTApplication*-Klasse ab und überschreibt die Methode *startUp()*. Diese Methode wird vom MT4j-Framework aufgerufen, sobald das *MTApplication*-Objekt initialisiert wurde, und dient u.a. dazu die einzelnen Szenen der Anwendung zu instanziieren und dem *MTApplication*-Objekt zuzuweisen. Das *MTApplication*-Objekt selbst wird nicht direkt sondern indirekt über den Aufruf seiner Methode *initialize()* erstellt sowie initialisiert. Die Klasse für das HelloWorld-Beispiel könnte folgendermaßen aussehen:

```
package basic.helloWorld;

import org.mt4j.MTApplication;

public class StartHelloWorld extends MTApplication {
    private static final long serialVersionUID = 1L;

    public static void main(String[] args) {
        initialize();
    }
    @Override
    public void startUp() {
        addScene(new HelloWorldScene(this, "Hello World Scene"));
    }
}
```

Abb. 4: *MTApplication*-Klasse des HelloWorld-Beispiels

Wie schon erwähnt besteht jede Applikation aus mindestens einer Szene. In einer Szene werden alle weiteren benötigen Objekte erstellt wie z.B. unsere „Hello World“ Zeichenkette. Die Szenen-Klasse erbt wiederum von der abstrakten Klasse *AbstractScene*. Folgender Quellcode zeigt ein Szenen-Objekt, welches allerdings noch nichts macht:

```
package basic.helloWorld;
import org.mt4j.MTApplication;

public class HelloWorldScene extends AbstractScene {

    public HelloWorldScene(MTApplication mtApplication, String name) {
        super(mtApplication, name);

    }
    @Override
    public void init() {}
    @Override
    public void shutDown() {}
}
```

Abb. 5: Szenen-Klasse *HelloWorldScene* des HelloWorld-Beispiels (ohne Funktionalität)

Als nächstes wollen wir unsere formatierte Zeichenkette „HelloWorld“ zentriert auf dem Bildschirm darstellen sowie die Hintergrundfarbe verändern. Dazu erstellen wir ein *MTTextArea*-Objekt, welches unsere Zeichenkette repräsentieren wird, und verändern dessen Attribute mit den Methoden *setNoStroke()*, *setNoFill()*, *setText()* und *setPositionGlobal()*. Um das *MTTextArea*-Objekt zur Darstellung zu bringen, muss es noch dem Szenengraph hinzugefügt werden. Dies geschieht mit dem Aufruf *getCanvas().addChild(textField)*. Die Hintergrundfarbe für die Szene wird mit *setClearColor()* gesetzt.

```
package basic.helloWorld;
import org.mt4j.MTApplication;
import org.mt4j.components.visibleComponents.font.FontManager;
import org.mt4j.components.visibleComponents.font.IFont;
import org.mt4j.components.visibleComponents.widgets.MTTextArea;
import org.mt4j.input.inputProcessors.globalProcessors.CursorTracer;
import org.mt4j.sceneManagement.AbstractScene;
import org.mt4j.util.MTColor;
import org.mt4j.util.math.Vector3D;

public class HelloWorldScene extends AbstractScene {

    public HelloWorldScene(MTApplication mtApp, String name) {
        super(mtApp, name);

        MTColor white = new MTColor(255,255,255);
        this.setClearColor(new MTColor(146, 150, 188, 255));
        //Show touches
        this.registerGlobalInputProcessor(new CursorTracer(mtApp, this));

        IFont fontArial = FontManager.getInstance().createFont(mtApp, "arial.ttf",
                50,     //Font size
                white,  //Font fill color
                white); //Font outline color
        //Create a textfield
        MTTextArea textField = new MTTextArea(mtApp, fontArial);

        textField.setNoStroke(true);
        textField.setNoFill(true);

        textField.setText("Hello World!");
        //Center the textfield on the screen
        textField.setPositionGlobal(new Vector3D(mtApp.width/2f, mtApp.height/2f));
        //Add the textfield to our canvas
        this.getCanvas().addChild(textField);
    }
    @Override
    public void init() {}
    @Override
    public void shutDown() {}
}
```

Abb. 6: Szenen-Klasse HelloWorldScene des HelloWorld-Beispiels (mit Funktionalität)

Da wir nur ein Gefühl dafür bekommen sollen, wie eine Anwendung bisher erstellt wird, reicht uns diese Detailtiefe aus und ich gehe an dieser Stelle nicht weiter auf Einzelheiten ein. Wie man allerdings schon anhand dieser sehr einfachen Beispielanwendung sehen kann, besteht Verbesserungspotential bei der Erstellung dieser Art von Anwendungen. So bestehen bisher folgende Nachteile bzw. Probleme:

- Es ist relativ viel Source-Code zu schreiben. Zwar können uns IDEs wie Eclipse die Arbeit vereinfachen, indem sie den Code für z.B. ableitende Klassen oder überschreibende Methoden generieren. Dies hat aber seine Grenzen, so müssen trotzdem Angaben wie Position, Rotation oder Farbe eines Objekts mühsam von Hand eingeben werden.

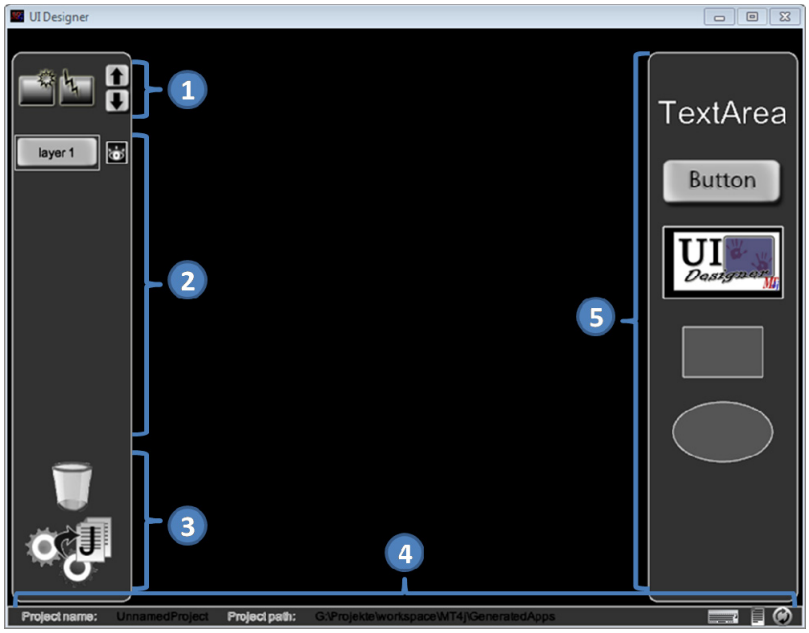
- Die Änderungen, die man an seinem Quellcode vornimmt, sind nicht direkt sichtbar. Wenn man z.B. die Position eines Objektes anpassen muss, lässt sich erst nach Rebuild und Ausführung der Anwendung erkennen, ob sich das Objekt nun tatsächlich an der gewünschten Position befindet. Viel intuitiver wäre es doch das Objekt in einem Editor an die gewünschte Position zu verschieben und somit das Ergebnis direkt zu sehen.
- Insgesamt ist relativ viel Zeitaufwand für immer wiederkehrende Aufgaben nötig. Somit geht Zeit unnötig verloren, die für das Lösen eines zentralen Problems einer Anwendung genutzt werden könnte.

Diese Probleme bzw. Nachteile sollten mithilfe des in diesem Projekt zur entwickelnden Anwendung ausgemerzt werden. Die Erstellung von MT-Applikationen sollte vereinfacht und intuitiver werden. Im nachfolgenden Kapitel wird der entwickelte Multi-Touch UI-Designer vorgestellt.

Multi-Touch UI-Designer

Um die Arbeitsweise mit dem Multi-Touch UI-Designer zu erläutern, dient das nachfolgende Schritt-für-Schritt-Beispiel. Zuvor soll noch die nachfolgende Grafik den Aufbau des Multi-Touch UI-Designers skizieren bzw. erläutern.

Überblick



1 Erstellen & Löschen von Layern, Ändern der Reihenfolge von Layern

2 Darstellung der vorhandenen Layer, Ein- und Ausblenden von Layern, Auswählen der „Work“-Layer

3 Löschen von Objekten, Generieren des Source-Codes für die aktuelle Szene

4 Informationen zum Projekt

5 „ToolBox“: Verfügbare Objekte/Steuerelemente

Abb. 7: GUI des Multi-Touch UI-Designers

Informationen über das aktuelle Projekt werden am unteren Bildschirmrand dargestellt 4. Zu den Projektinformationen zählen bisher der Projektname, der Projektpfad sowie eine Projektbeschreibung, die über den Button aufgerufen werden kann. Diese Informationen werden in einer separaten XML-Datei gespeichert, welche über den Button neugeladen werden kann.

Auf der linken Seite befindet sich die sogenannte ToolBar. Diese ist in drei Bereiche unterteilt. Im Bereich 1 ist es möglich, neue Layer zu erstellen bzw. die aktuelle Layer zu löschen. Mit Hilfe einer Layer lassen sich z.B. Objekte logisch gruppieren und somit gemeinsam ein- oder ausblenden, was die Arbeit durch bessere Übersicht erleichtern kann. Darüber hinaus lässt sich die Reihenfolge

der Layer mit  bzw.  ändern. Dies kann z.B. bei 2D-Applikationen Sinn machen, so definiert sich dadurch u.a. in welcher Reihenfolge Objekte gezeichnet werden sollen.

Im Bereich 2 werden alle Layer der aktuellen Szene dargestellt. Hier wählt man auch die Layer aus, auf der man arbeiten möchte. Über die Buttons  bzw.  lassen sich zudem wie bereits schon oben erwähnt einzelne Layer ein- bzw. ausblenden.

Im Bereich 3 gibt es einerseits einen Papierkorb  . Über diesen lassen sich vorher erstellte Objekte wieder löschen. Der darunterliegende Button  ermöglicht die Generierung des Source-Codes für die aktuelle Szene. Diese Funktionalität ist von zentraler Bedeutung.

Auf der rechten Bildschirmseite befindet sich die sogenannte Toolbox 5. Hier sind die verfügbaren grafischen Objekte bzw. UI-Komponenten enthalten, die man in seiner eigenen Szene verwenden kann. Per Drag&Drop lassen sich so diese Objekte auf eine bestimme Position in der aktuellen Szene (schwarzer Bereich im Zentrum) ziehen.

An einem kurzen Beispiel soll nun die Vorgehensweise beim Entwerfen einer Szene und dem Generieren des entsprechenden Source-Codes zu dieser Szene veranschaulicht werden. Wir wollen, dass in unserer Szenen der Text „Hallo“ dargestellt wird. Dazu ziehen wir per Drag&Drap aus der ToolBox das Objekt *TextArea* auf die Szene und platzieren den Text nah dem Bildschirmmittelpunkt. Abbildung 9 zeigt das Resultat. Da unser *TextArea*-Objekt per Default den Text „newTextX“ hat, wobei X für eine Zahl steht, wollen wir diesen nun in „Hallo“ ändern. Dazu ziehen wir das Keyboard-Symbol  , welches am rechten unteren Bildschirmrand zu finden ist, auf das *Textarea*-Objekt, welches wir ändern wollen. Indem wir das Keyboard auf das *TextArea*-Objekt „droppen“, verbinden wir es mit dem Keyboard und können nun den Text in „Hallo“ umbenennen. Sobald wir fertig sind, können wir das Keyboard wieder schließen. Abbildung 8 zeigt den Vorgang, bei dem der Text geändert wird. Da wir nun unsere Szene fertiggestellt haben, können wir den dazugehörigen Quellcode mit dem Button  generieren lassen. Die generierten Klassen werden im vorher definierten Projektpfad erstellt.

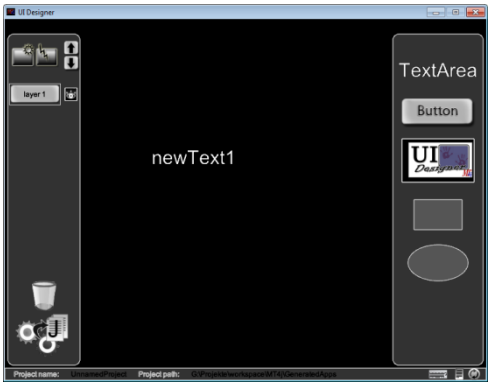


Abb. 9: Einfügen einer TextArea

Abb. 8: TextArea umbenennen

Um die generierten Klassen bzw. die generierte MT-Applikation direkt zu testen, ist es nützlich den Projektpfad direkt in die Entwicklungsumgebung, in diesem Fall war es die IDE Eclipse, einzubinden. Somit kann man die Klassen direkt kompilieren lassen und die generierte Applikation ausführen, um diese auszuprobieren. Zudem kann man an dieser Stelle die Arbeit mit der Szene bzw. die Entwicklung der Applikation fortsetzen und z.B. das gewünschte Verhalten der Applikation implementieren.

Nachdem der Quellcode der Applikation aus der vorher getätigten Oberflächenkonfiguration generiert wurde, sollten sich in diesem Beispiel zwei Java-Klassen in dem Projektpfad (hier: *GeneratedApps\UnnamedProject*) befinden. Eine Java-Klasse für das *MTApplication*-Objekt und eine Java-Klasse, die die Szene repräsentiert. Die nachfolgende Abbildung soll dies verdeutlichen.

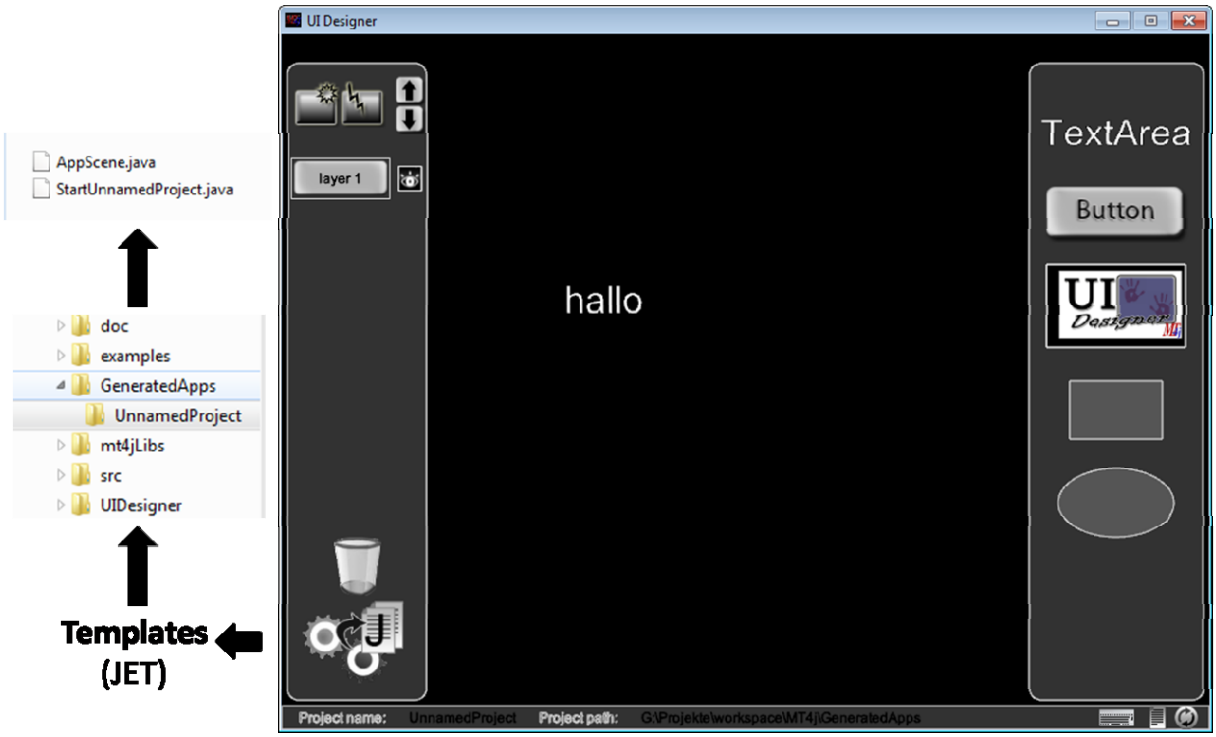


Abb. 10: Ergebnis nach der Code-Generierung

Analyse

Da das Projekt bereits in Form der Aufgabenstellung (Projektvorschlag) größtenteils definiert war, erfolgte nach der Einarbeitung in das Framework MT4j eine Analyse des Problems bzw. dieser Aufgabenstellung. Diese sollte mir sicherstellen, dass ich die Aufgabenstellung verstanden habe und nicht das Projektziel verfehlen würde. Dazu entstand ein Dokument mit einem groben Entwurf der GUI (siehe Abbildung) sowie einer Darstellung der Code-Generierung, so wie ich sie bis dahin verstanden hatte, und offenen Fragen, die beim ersten Projekttreffen mit Herrn Uwe Laufs besprochen und geklärt wurden. Auch erhielt ich bei diesem Projekttreffen bereits wertvolle Tipps für die Umsetzung der Code-Generierung, nämlich die Verwendung von JET (Java Emitter Templates), die einfacher einzusetzen waren, als die Technologie Apache Velocity.

Für die Entwicklung des Multi-Touch UI-Designers hätten theoretisch unterschiedliche Technologien zur Verfügung gestanden. Die nächstliegende Idee, welche durch Herrn Laufs vorgegeben wurde, war es, das Framework MT4j selbst als Basis für den Multi-Touch UI-Designer zu verwenden. Als IDE kam Eclipse zum Einsatz und als Programmiersprache Java, da auch das Framework MT4j mit dieser DIE und Java entwickelt wird.

Nach diesem ersten Projekttreffen wurden unterschiedliche Test-Applikationen erstellt, um auch die Möglichkeiten des Framework zu ermitteln. Dadurch, dass ich das Framework vorher nicht gekannt hatte, war diese Phase für mich schwer. Zwar hatte ich eine Vorstellung davon, was entwickelt

werden sollte, jedoch nicht direkt, wie ich dies mit dem Framework lösen würde. Hierbei halfen mir die Test-Applikationen bzw. Wegwerf-Prototypen, mit denen ich herausfinden konnte, wie ich bestimmte Anforderungen umsetzen kann. Die Entwicklung dieser Prototypen war daher noch Bestandteil der Einarbeitungsphase in das Framework. Der prinzipielle und endgültige Aufbau des Multi-Touch UI-Designers wird im Kapitel „Entwurf“ vorgestellt.

Zusätzlich wurde das Projekt, nach dem erstem Projekttreffen, in zwei Haupt-Arbeitspakete mit folgenden Arbeitseinheiten unterteilt:

- UIDesigner-Szenen-Objekt
  - Erstellung der GUI
    - ToolBar, ToolBox, ProjInfoBar
    - Entwerfen und Erstellen von Grafiken (Buttons etc.)
  - Layer-Management
    - Neue Layer, Auswahl der aktuellen Layer, Layer löschen
  - Projekt-Settings
- Code-Generierung
  - Erstellung von Templates (JET)
  - Generierung der Java-Klassen mit Hilfe der Oberflächenkonfiguration und der Templates (JET)

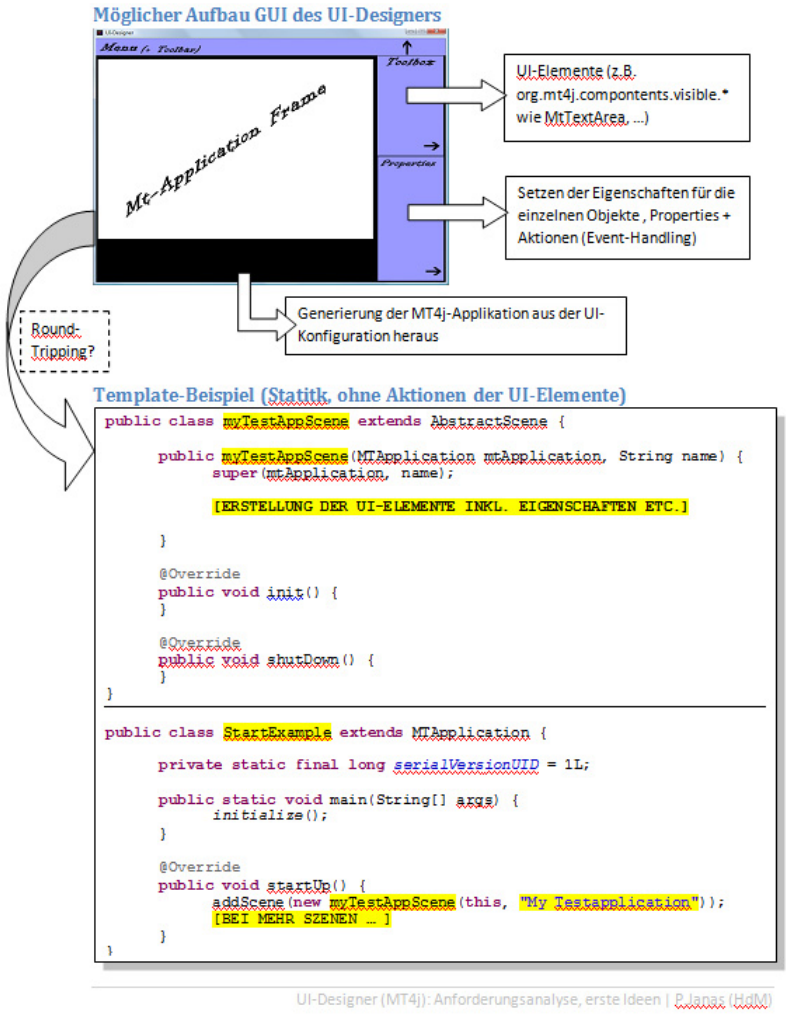


Abb. 11: Ausschnitt aus dem Dokument „Erste Ideen“

Entwurf und Implementierung

Nachdem einige Wegwerf-Prototypen entwickelt wurden, wurde mit dem groben Entwurf der Klassen für den Multi-Touch UI-Designer begonnen. Dieser Entwurf wurde immer wieder überarbeitet, die Klassen wurden refaktorisiert und die Änderungen anschließend implementiert, sodass sich der Entwurf dem endgültigen Entwurf annäherte. Auf eine detailierte Darstellung der endgültigen Klassen soll an dieser Stelle verzichtet werden, dazu wird auf den Source-Code des UI-Designers verwiesen. Stattdessen wird an dieser Stelle ein schematischer Aufbau dargestellt, der übersichtlicher ist:

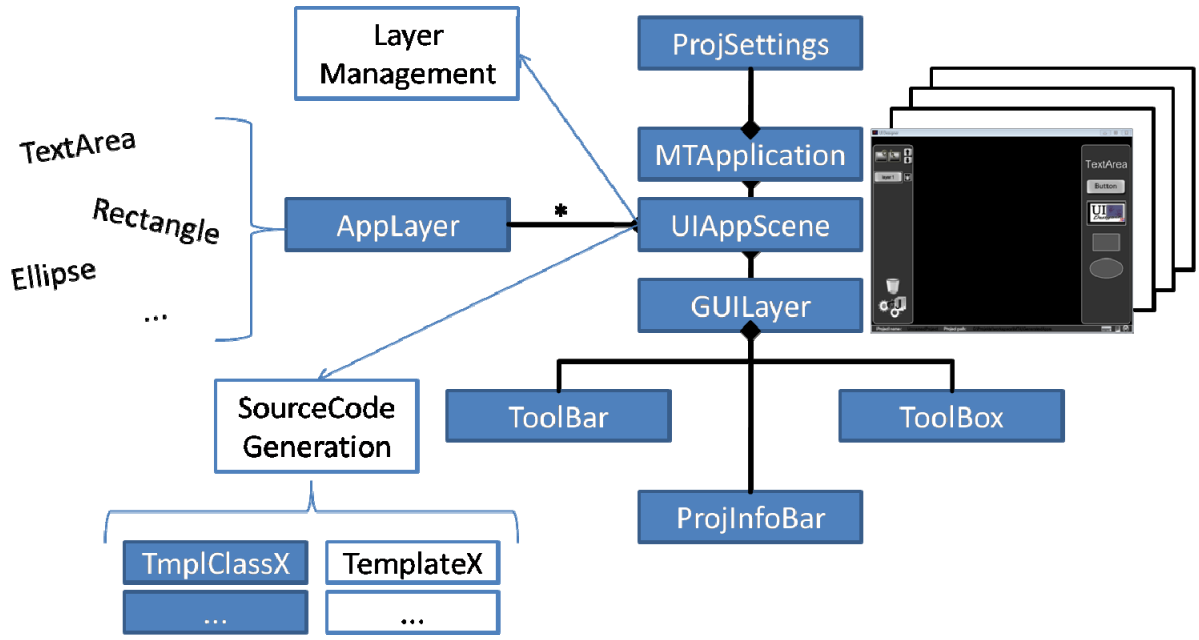


Abb. 12: Schematischer Aufbau des Multi-Touch UI-Designers (Software-Architektur)

Der UI-Designer besteht zunächst aus einem *MTApplication*-Klasse, welche Einstiegspunkt ist, zum Initialisieren der Anwendung und zum Laden der Projekt-Einstellungen (*ProjSettings*) benötigt wird, und einer Szenen-Klasse (*UIAppScene*). Die Szenen-Klasse repräsentiert gewissermaßen den UI-Designer und ist in zwei Arten von Layern (*MTComponent*-Objekte) unterteilt: einerseits eine GUI-Layer, die Bedienoberfläche der Anwendung darstellt und u.a. die Bereiche ToolBar, ToolBox und ProjInfoBar beinhaltet, sowie mindestens eine Applikation-Layer (i.d.R. sind es mehrere). Auf den Applikation-Layern stellt der Benutzer des UI-Designers seine Szene zusammen, wie er sie haben möchte. Mit Hilfe der mehrere Layer lassen sich so Objekte logisch gruppieren und so z.B. während des Entwerfens der Szene gemeinsam ein- und ausblenden. In einer Applikation-Layer können theoretisch alle vom MT4j-Framework vorgegebenen UI-Komponenten (bzw. alle von *MTComponent* abgeleitete Klassen) verwendet werden. Bisher werden jedoch nur einige wenige Komponenten u.a. TextArea, Rectangle sowie Ellipse exemplarisch unterstützt.

Neben diesen zwei Arten von Layern ist die *UIAppScene*-Klasse für das Layer-Management zuständig. Dazu gehört u.a. das Anlegen neuer oder das Löschen von Layern. Zusätzlich steuert diese Klasse die Generierung des Source-Code für die durch den Benutzer erstellten Szene.

Der gezeigte Aufbau ist wie oben schon erwähnt schematisch. Die Aufteilung in Klassen ist granularer, wie im Source-Code zu sehen ist.

Implementiert wurde der Multi-Touch UI-Designer in Java unter der Verwendung von Eclipse basierend auf dem Framework MT4j. Für die Code-Generierung wurden Java Emitter Templates (JET) eingesetzt. Dazu war es nötig mehrere Templates zu schreiben, aus denen anschließend Klassen generiert wurden (jeweils eine Klasse pro Template), mit deren Hilfe man parametrisiert Source-Code (d.h. Java-Klassen) schreiben konnte. JET hat gegenüber Apache Velocity[1] den Vorteil, dass direkt Java-Klassen kreiert werden. Zudem wurde der Source-Code mit Javadoc-Kommentaren versehen, aus denen sich eine Dokumentation der vorhandenen Klassen generieren lässt. Für das Einlesen und Schreiben der Projekt-Einstellung im XML-Format kam die Library XStream[2] in der Version 1.3.1 zum Einsatz.

Da ich leider viel Zeit in der Einarbeitung in das Framework MT4j verloren hatte, fehlte mir zum Schluss die Zeit, alle Anforderungen des Projektes umzusetzen. So lässt sich das Verhalten der UI-Komponenten, die ein User auf seine Szene platziert hat, noch nicht zuweisen. Auch sind Farbinformationen der einzelnen Komponenten nicht über den UI-Designer einstellbar. Zwar bestanden hier grundsätzliche Ideen, wie diese Funktionalität umgesetzt werden könnte, jedoch fehlte am Ende leider die Zeit dafür. Dass während der Umsetzung dieser Ideen keine Probleme, die das Verständnis des Frameworks etc. betroffen hätten, entstanden wären, ist natürlich nicht ausgeschlossen.

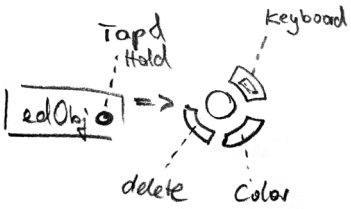


Abb. 13: Tap & Hold Menu

Auch ist es bisher nötig, den generierten Source-Code zu formatieren bzw. die Importe zu organisieren. Dies hätte z.B. mit dem CodeFormatter der Eclipse-Entwicklungsumgebung gelöst werden können. Auf der anderen Seite nimmt diese Formatierung des Quellcodes nicht viel Zeit in Anspruch und lässt sich in Eclipse mit zwei Hotkeys erledigen ([Strg]+[Shift]+F für Format und [Strg]+[Shift]+O für Organize Imports). Ein Implementierung hätte daher die zum Projektschluss kostbare Zeit unnötig verschwendet.

Weiterhin wäre es nötig die Generierung des Source-Codes für die spätere Nachbearbeitung durch den Benutzer zu optimieren, so ist manche Bearbeitung umständlich bzw. komplizierter (z.B. das Setzen der Position eines Objektes z.Z. über eine Matrix).

UI-Design

Für die Benutzungsschnittstelle war es erforderlich, Grafiken zu erstellen. Zu Beginn des Projektes wurden zunächst nur einfache Dummy-Grafiken erstellen, da hier noch nicht klar war, inwieweit sich die GUI des Designers noch ändern wird. Daher sollte vermieden werden, dass unnötig viel Zeit in die Erstellung von Icons bzw. Grafiken fließt, die später womöglich wieder verworfen werden.

Im Laufe des Projekts, nachdem die die Grafik betreffende Funktionalität des UI-Designers stand, wurden die vorläufigen Grafiken durch die endgültigen Grafiken ersetzt. Für das Erstellen bzw. Bearbeiten der Grafiken wurden Paint.NET (Version 3.5.4) sowie Photoshop CS (Version 8.0.1) verwendet.

[1] velocity.apache.org
[2] Eine einfache Bibliothek um Java-Objekte in XML zu (de-)serialisieren, xstream.codehaus.org

Projektverlauf

Projektstart

Da für das Projekt ein maximaler Workload von 240 Stunden seitens der Hochschule vorgesehen war, dies entspricht 30 Tagen zu je 8 Stunden, wurde zu Beginn des Projektes folgende Zeitplanung bzw. Aufteilung des Projektes vorgenommen:

Projektphase	Geschätzte Dauer
Definition-Phase bzw. Analyse-Phase	**5-9 Tage**
- Einarbeitung in die Thematik	1-2 Tage
- Einarbeitung in das MT4J-Framework inkl. Beispiele	2-3 Tage
- Anforderungsanalyse [1-2 Tage] + Pflichtenheft [1-2 Tage]	2-4 Tage
Projektplanung	**2-4 Tage**
- Arbeitspakete, Zeitplanung etc.	
Entwurf	**3-5 Tage**
- Architektur- und Klassenentwurf	
Implementierung	**4-6 Tage**
Testen	**2-3 Tage**
Präsentation	**2-3 Tage**
- Erstellung und Vorbereitung	

Da ich in diesem Projekt der einzige Entwickler war, sah ich es nicht für erforderlich einen Netzplan zu erstellen, bei dem parallel durchführbare Aufgaben ermittelt und eingeplant worden wären.

Vorgehensweise/Vorgehensmodell

In diesem Abschnitt soll nochmals die tatsächliche Vorgehensweise bzw. der Projektablauf verdeutlicht werden. Dieser verlief nämlich nicht geradlinig und ohne Rücksprung zu vorherigen Phasen sondern bestand insbesondere aus vielen Iterationen zwischen den Phasen Design (Entwurf) und Implementierung. Die anschließende Abbildung soll die nachfolgende Erläuterung unterstützen:

Nachdem die Aufgabenstellung analysiert wurde, war es für mich wichtig, diese auch wirklich richtig verstanden zu haben. Dazu wurde von mir zunächst die grundsätzliche Idee des Multi-Touch UI-Designers in einem Dokument festgehalten und beim ersten Projekttreffen (nach der Anmeldung des Projekts an der Hochschule der Medien) mit Herrn Uwe Laufs besprochen.

Der nächste Schritt bestand darin, mich in das Framework MT4j einzuarbeiten und die vorhandenen Beispiele zu verstehen. Dieser Schritt hatte mehr Zeit in Anspruch genommen als gedacht und wurde auch nicht direkt abgeschlossen, sondern begleitete das gesamte Projekt, da ich im Laufe des Projektes immer wieder Bestandteile des Frameworks erlernen und verstehen musste. Zusätzlich zum Framework MT4j war es nötig die Technologie JET (Java Emitter Templates) zu verstehen. Hierzu hatte ich mich Tutorials und Beispielen der Internetseiten www.eclipse.org[3] und www.vogella.de[4] bedient.

[3] http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html
[4] http://www.vogella.de/articles/EclipseJET/article.html

In dieser Einarbeitungsphase entstanden bereits Prototypen des Multi-Touch UI-Designers, mit Hilfe derer ich die Möglichkeiten des Frameworks ausprobierte. Da die Prototypen nur zu Testzwecken dienten („Quick&Dirty“), wurde ein Großteil des dort entstanden Source-Codes wieder verworfen.

Nach dieser Einarbeitung folgten mehrere Entwürfe der benötigten Klassen. Die entworfenen Klassen wurden in Java implementiert und anschließend getestet. Hierbei wurden bei Bedarf, trotz der vorher erstellten Prototypen, erneut Prototypen bzw. Test-Beispiele erstellt, um einen Entwurf erst separat zu testen und nicht direkt in die Anwendung zu integrieren. Im Gegensatz zu den Wegwerf-Prototypen aus der Analyse-Phase, wurden diese in den Multi-Touch UI-Designer übernommen. Während der Iteration von Design und Implementierung fanden bei Bedarf Projekttreffen mit Herrn Uwe Laufs statt, bei denen der derzeitige Stand der Projekts vorgestellt wurde und Fragen bzw. Probleme angesprochen und geklärt wurden.

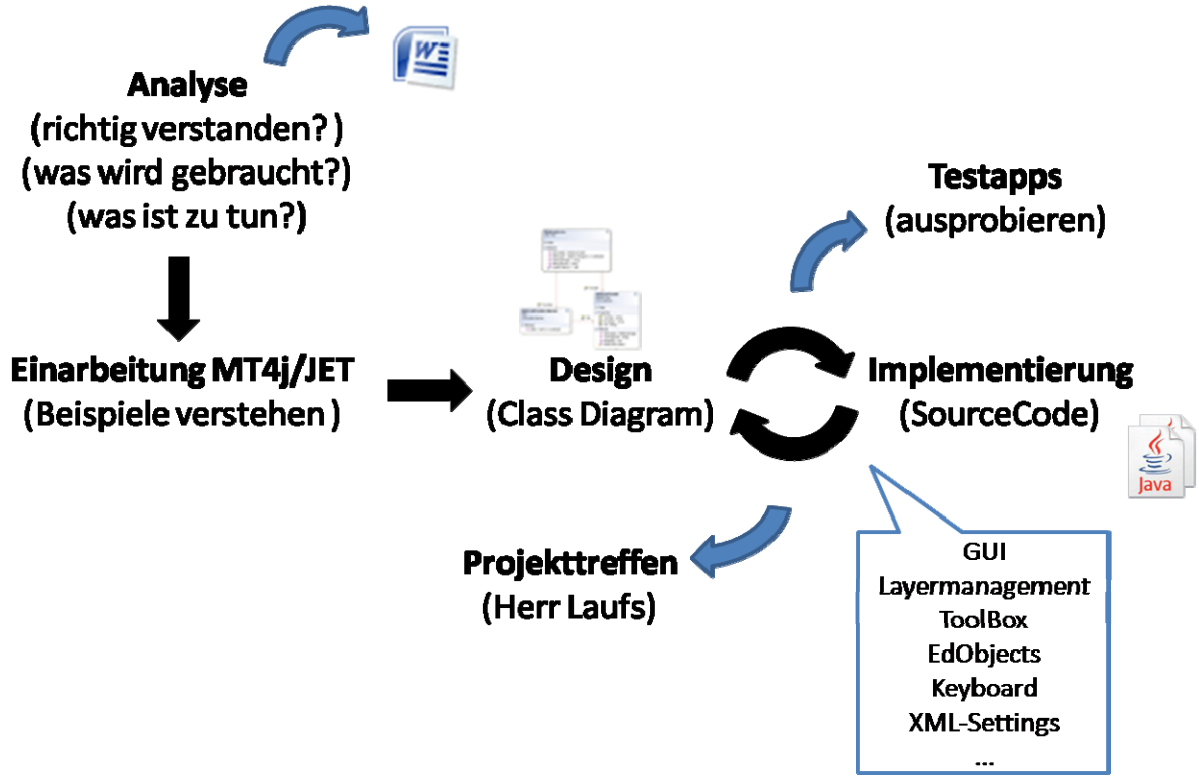


Abb. 14: Vorgehensweise bzw. Vorgehensmodell dieses Projektes

Neben dem Multi-Touch UI-Designer musste zusätzlich noch eine dazugehörige Präsentation für den MI-Präsentationstag (27. Januar 2010), Vorbereitungen für die MediaNight (28.Januar 2010) und die vorliegende Dokumentation mit eingeplant und erstellt werden.

Informationsquellen

Im Laufe des Projektes gab es diverse Fragen und Probleme die geklärt werden mussten. Als Informationsquelle stand überwiegend Herr Uwe Laufs zur Verfügung. Ein Großteil meiner Fragen betraf das Framework MT4j. Auch waren der Source-Code des Frameworks sowie die darin

enthaltenen Beispiele informationsgebend. Zusätzlich war die Hauptseite zum Framework www.mt4j.org für den Einstieg hilfreich, auch existieren zwei dazugehörige Internet-Foren[5].

Für Fragen die Java, die IDE Eclipse oder JET angingen, reichte es nach Informationen im Internet zu „googlen“ bzw. auf den betreffenden Web-Seiten der jeweiligen Technologie nachzuschauen.

Probleme

Es gab sowohl fachliche Probleme wie auch das Projektmanagement betreffend.

Die größte Schwierigkeit bestand für mich in der Einarbeitung in des Framework MT4j. Diese hatte nämlich deutlich mehr Zeit in Anspruch genommen, als ich es zu Beginn des Projektes mit meiner bisherigen Erfahrung, was andere Frameworks angeht, hätte einschätzen können. Diese Fehleinschätzung hatte meiner Meinung nach unterschiedliche Gründe. Einerseits lag es natürlich an meiner bisher mangelnden Erfahrung, was unterschiedliche Frameworks bzw. die Durchführung von Projekten betrifft. Andererseits lag es daran, dass das MT4j-Framework relativ jung und damit vergleichsweise nicht sehr weit verbreitet ist. Folglich ist die Community (noch) sehr klein und man findet im Allgemeinen durch eine einfache „Internet-Suche“ nicht die Lösung oder Lösungshinweise zu seinem Problem. Damit fallen wichtige Informationsquellen, die ich von anderen Frameworks wie etwa den MFC (Microsoft Foundation Classes) gewohnt bin weg. Zudem war ich hin und wieder in dem Dilemma, nicht einschätzen zu können, ob die Ursache bei einem Problem bei mir selbst liegt (also z.B. der falschen Verwendung bestimmter Klassen) oder einfach nur noch ein nicht gefundener Bug bzw. eine nicht abgeschlossene Implementierung darstellte. Bei einem ausgereiftem Framework wie etwa den MFC, mit denen ich längere Zeit gearbeitet hatte, wusste ich i.d.R. bei Fehlern, wo ich suche sollte. Eine weitere Konsequenz, die sich aus der Tatsache ergibt, dass das MT4j vergleichsweise jung ist, war der zwar gute aber für Anfänger relativ beschränkte Umfang der Dokumentation zum Projektstart, der mir persönlich damit den Einstieg erschwerte. Nichtsdestotrotz gab es auf der anderen Seite auch genügend Beispielcode, mithilfe dessen man das Framework ebenfalls erlernen konnte. Auch wurden kurz vor Ende des Projekts sehr nützliche Tutorials veröffentlicht, die speziell an Anfänger des Frameworks gerichtet sind und mir zu Beginn des Projektes das Verständnis des Frameworks aus jetziger Sicht erleichtert hätten.

Das fehlende Wissen über den Umgang mit dem Framework führte häufig dazu, dass eine Idee, die ich umsetzen wollte, darin endete, dass ich mich schnell „in den Tiefen“ des Frameworks verloren hatte – auf der Suche nach den benötigten Klassen bzw. Methoden oder nach dem Grund für das nicht gewünschte Verhalten meiner Implementierung. Zwar habe ich oft darauf geachtet, mich nicht zu „verrennen“, jedoch ist es stellenweise einfach nicht möglich, bestimmte Probleme einfach auszuklammern, und an derer Stelle weiterzumachen, um zu versuchen weiterzukommen, da man somit nur auf der Stelle tritt. Zudem sind solche Momente auch frustrierend und erschweren damit die weitere Arbeit am Projekt.

Die obige Problematik führte dazu, dass bei der Planung des Projektes zu wenig Zeit für die Einarbeitung bzw. das nötige Erlernen des Frameworks Mt4j eingerechnet wurden. So war es auch schwer abzuschätzen, wann genau der richtige Zeitpunkt, mit dem Entwurf der Klassen und deren Implementierung anzufangen gewesen wäre und die Einarbeitungsphase damit abzuschließen. Da ich auch nicht zu früh mit dem Entwickeln bzw. Programmieren der tatsächlichen Anwendung anfangen wollte, musste hier ein Mittelweg gefunden werden. Hilfreich war es dabei, nicht einen kompletten

[5] http://mt4j.org/forum sowie http://nuigroup.com/forums/viewforum/81/

Entwurf der Anwendung zu versuchen, sondern stattdessen durch Iterationen zwischen Entwurf und Implementierung, mich der endgültigen Anwendung zu nähern.

Weiterhin gab es auch fachliche Schwierigkeiten, die z.B. die Arbeit mit der Entwicklungsumgebung Eclipse angingen. Meine bisherige Arbeit mit dieser IDE beschränkte sich auf vergleichsweise kleine Anwendungen, die im Rahmen von Vorlesungen oder Übungen erstellt wurden und im Umfang mit diesem Projekt entsprechend nicht vergleichbar sind. So hatte ich z.B. bisher keine Erfahrung mit Projekten (in der Eclipse Entwicklungsumgebung), die aufeinander aufbauen (geeignete Projektstruktur wählen, build paths richtig setzen, referenzierende Libraries usw.; bei der Verwendung von JET als separatem Projekt nötig) oder diese Projektgröße hatte (Übersicht behalten, schnell Codestellen wiederfinden etc.). Selbst wenn diese Probleme banal klingen, haben sie dennoch den Fortschritt des Projektes nur unnötig verzögert.

Neben dem Mt4j-Framework war es zusätzlich erforderlich, die Technologie JET (Java Emitter Templates) zu erlernen. Dies stellte jedoch keine große Hürde dar, weil die Syntax den mir bekannten JSP (Java Server Pages) sehr ähnelte. So war der Zeitaufwand, der für das Einarbeiten nötig war, durch die Zeitersparnis, die mir diese Technologie bei der Code-Generierung eingebracht hatte, schnell wieder reingeholt.

Zu Beginn des Projekts habe ich mit der damaligen öffentlichen Version 0.81RC des Frameworks gearbeitet. Dies war für den Anfang praktisch und auch sinnvoll, da kein SVN-Account angelegt werden musste, und mir somit die Entwicklung zunächst vereinfachte – auch weil ich auf eine bestimmte Framework Version hin entwickelt habe. Im Laufe des Projekts zeigte sich jedoch, dass ich Stellenweise Features aus der aktuellen Entwicklung hätte gebrauchen könnten (z.B. MTList).

Bei der ursprünglichen Planung wurden zu wenig zusammenhängende Tage für diese Projekt eingeplant. Dies hätte sich schnell bemerkbar gemacht, insbesondere zu Beginn des Projektes in der Phase der Einarbeitung in der Mt4j-Framework. Hier wurde der Abstand zwischen den Tagen zu groß gewählt und effiziente Arbeit wurde damit schwer, da somit viel Zeit dafür verbraucht wurde, einen Überblick zu gewinnen, wo man beim letzten Mal stehen geblieben war, also um in das Projekt „wieder reinzukommen“. Diesem Manko wurde schnell damit Rechnung getragen, indem ich bei meiner Vorlesungsplanung Änderungen vorgenommen hatte und sich damit mehr zusammenhängende Tage für das Projekt ergaben.

Ein Problem ganz anderer Natur war und ist meine (teils schlechte) Charaktereigenschaft, mit Schwierigkeiten häufig erst selbst klarkommen bzw. Probleme zuerst selbst lösen zu wollen, was eventuell damit zusammenhängt, dass ich hin und wieder ungern bei anderen Personen nachfrage, in der Angst, das Gefühl zu haben, diese unnötig zu belästigen. So hatte ich bei konkreten Problemen, die z.B. das Framework angingen, zu Beginn die Scheu gehabt, Herrn Uwe Laufs direkt darauf anzusprechen und so eventuell zu einer schnellen Lösung zu kommen. Stattdessen habe ich somit unnötig Zeit damit verbracht, bestimmte Dinge erst selbst auszuprobieren. Dies hatte sich im Laufe des Projekts verbessert. Hier hätte ich vielleicht schon zu Beginn mehr verbindliche feste Termine für Projekttreffen einplanen sollen.

Fazit – „Lessons Learned“

Insgesamt war dieses Projekt sehr interessant wie auch lehrreich für mich gewesen. Auch wenn nicht alles im Projektverlauf so lief, wie es geplant wurde bzw. wie ich es mir erhofft hatte und ich nicht alle Anforderungen des Projekts umsetzen, daher das Projekt auch nicht fertig wurde, konnte ich dennoch einiges aus dem Projekt für mich mitnehmen. So habe ich ein interessante Framework kennengelernt und habe dadurch einen Einblick in die Entwicklung von Multi-Touch-Anwendungen bzw. in die Thematik selbst erhalten. Die Möglichkeiten, die sich mit der Multi-Touch-Technologie eröffnen, sind so vielfältig und innovativ, die Bedienung dieser Anwendungen kann so intuitiv sein, so dass ich gespannt bin, wie sich dieser Trend bzw. der Markt in den nächsten Monaten und Jahren weiterentwickeln wird.

Auch hatte ich durch dieses Projekt die Möglichkeit gehabt, einen Einblick in die Code-Generierung mit Hilfe von Templates zu bekommen, speziell habe ich also die Arbeit mit den JET (Java Emitter Templates) kennengelernt.

Durch die im Vergleich zu in Rahmen von Vorlesungen erstellten Übungen etc. intensivere Arbeit mit Eclipse konnte ich, was diese Entwicklungsumgebung angeht, mein Wissen vertiefen.

Neben diesem Wissen, konnte ich erneut – so wie es bisher bei jedem Projekt war – Erfahrungen bzgl. Projektmanagement sammeln. Allerdings musste ich auch wieder feststellen, wie schwer die Planung bzw. das Schätzen von Arbeitspaketen sein kann, wenn man noch wenig Erfahrung auf diesem Gebiet mitbringt. Obwohl ich bisher schon mit anderen Frameworks gearbeitet hatte und dieses Framework im Prinzip den mir bekannten ähnelte, habe ich dennoch den Zeitaufwand für die Einarbeitung in dieses mir bis dahin unbekannte Framework klar unterschätzt. In Zukunft werde ich bei unbekannten Frameworks bzw. Technologien genauer recherchieren, in wie weit es bestehende Dokumentationen gibt bzw. mir bestimmte Informationsquellen helfen können. Dadurch lässt sich der Zeitaufwand besser schätzen.

Einen Fehler, den ich nicht mehr machen werde ist, zu glauben, dass z.B. fünf Projekttage, die über zwei Wochen verteilt sind, genauso effektiv seien, wie fünf zusammenhängende Projekttage. Dies hätte ich bereits bei der anfänglichen Projektplanung berücksichtigen sollen. Auf der anderen Seite ist es aber auch wichtig, Abstand vom Projekt finden zu können, da man sich sonst schnell „fest fährt“. Auch ist es manchmal nicht anders möglich, da man mehrere Projekte parallel hat – in meinem Fall bestimmte feste Vorlesungszeiten, wenn man so will. Ein Vorteil die Projekttage nicht am Stück zu wählen ist aber auch, dass einem bzw. mir häufig interessante Ideen nicht beim Entwickeln, sondern wenn ich etwas komplett anderes mache (z.B. Auto fahren) kommen.

Weiterhin werde ich in Zukunft genauer feste Projekttreffen einplanen. In diesem Projekt wurden diese bei Bedarf gehalten, daher fehlte der Charakter der Verbindlichkeit, wie ich finde. Hier hätten besser feste Milestones durch mich gesetzt werden müssen.

Auch hätten mehr Projekttreffen nicht geschadet. Ein Grund für die relativ wenigen Projekttreffen (insbesondere in der ersten Hälfte des Projektes) ist ein Charakterzug von mir. So hatte ich anfangs die Scheu gehabt, zu oft bei meinem „Auftraggeber“ Herrn Uwe Laufs bei Problemen oder einfachen Fragen, die das Framework betreffen, direkt nachzufragen. Stattdessen wollte ich ihn nicht allzu oft „belästigen“ und sah mich auch der Position aus „Auftraggeber“, diese Schwierigkeiten selbst lösen zu müssen. Dadurch ergab sich das Problem, dass ich vieles im „Alleingang“ machen wollte („Lieber nicht nachfragen, sondern erst selbst probieren bzw. herausfinden“). Dass dieses Verhalten auf lange

Sicht nicht gut geht, zeigte sich in der zweiten Hälfte des Projektes, in der ich mir über solche Dinge keine Gedanken mehr gemacht hatte bzw. machen durfte und sozusagen über meinen eigenen Schatten gesprungen bin und mehr um Hilfe bat bzw. die Projekttreffen sich häuften. Ich denke, dass diese Problematik öfters in realen Projekten vorkommt als man glaubt. So ist es i.d.R. schwierig als Auftragnehmer, dem Auftraggeber (solange es nicht z.B. um Milestones etc. geht) zu gestehen, im Projekt Probleme zu haben, sofern man glaubt, diese bis zu einem bestimmten Zeitpunkt lösen zu können. Folglich ist oft eine fehlende, nicht ausreichende oder aber auch schlechte Kommunikation der Grund dafür, dass Projekte scheitern – Projekte scheitern nun mal mit den Menschen, die sie durchführen.

Genau unter diese Problematik – also der schlechten Kommunikation – fällt auch das Formulieren von Anforderungen. Es ist sehr schwer diese exakt! und vollständig! auf Papier zu bringen. Auch wenn diese (scheinbar) exakt formuliert sind, werde sie dennoch von Entwickler und Anwender oft anderes interpretiert. Somit ist es wichtig, möglichst früh den Anwender in das Projekt einzubeziehen und auch viele Iterationen im Projekt zu haben, um so den Anforderungen immer näher zu kommen.

Ein interessanter Aspekt, der mir im Laufe des Projekts – speziell bei der Einarbeitung in das Framework Mt4j - aufgefallen war bzw. von mir beobachtet wurde, war die damit verbundene Lernkurve. Zu Beginn war meine Effizienz das Framework betreffend sehr gering, daher auch oft frustrierend. So gab es auch nur hin und wieder „Aha“-Effekte, die meine Motivation steigern konnten. Zwar bestand jederzeit ein großes Interesse an der Arbeit, aufgrund des Themas Multi-Touch, jedoch war meist meine Motivation nicht von langer Dauer, da meist das nächste Problem schon auf mich zu warten schien. Gerade deshalb war es interessant in Laufe des Projektes zu beobachten, wie mir die Arbeit immer häufiger Spaß machte, da sich durch das bessere Verständnis des Frameworks immer mehr Erfolgserlebnisse ergaben. Anfangs hielt ich mich eher mit dem Framework auf, statt mit der eigentlichen Aufgabe bzw. meine Idee umzusetzen.

Abbildungsverzeichnis