Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Marius Pothmann

Vergleich und Bewertung von Clientarchitekturen und -technologien

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Marius Pothmann

Vergleich und Bewertung von Clientarchitekturen und -technologien

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Olaf Zukunft

Marius Pothmann

Thema der Bachelorarbeit

Vergleich und Bewertung von Clientarchitekturen und -technologien

Stichworte

Windows Forms, Windows Presentation Foundation, NetBeans, Java

Kurzzusammenfassung

Die Bachelorarbeit beschreibt den Aufbau von modernen Clientarchitekturen, die mit Hilfe erprobter Entwurfsmuster die Benutzerschnittstelle von der darunterliegenden Geschäftslogik trennen. Diese Trennung ermöglicht es, die Oberfläche einfach auszutauschen, ohne das am Modell Änderungen vorgenommen werden müssen.

Anhand von drei weitverbreiteten Technologien wird untersucht, wie diese Konzepte umgesetzt wurden und mit Hilfe eines zuvor erarbeiteten Kriterienkatalogs wird die Leistungsfähigkeit der Frameworks bewertet.

Marius Pothmann

Title of the paper

Comparison and evaluation of Client architectures and –technologies

Keywords

Windows Forms, Windows Presentation Foundation, NetBeans, Java

Abstract

This bachelor thesis describes the design of modern client architectures which separate the graphical user interface from the underlying business logic with the aid of approved design patterns. The separation permits to easily exchange the view without the need for modification on the model.

Three widely-used technologies are examined and compared in order to show how these concepts are implemented. The capabilities of the framework are evaluated based on a criteria catalogue.

Inhaltsverzeichnis

1 Einführung

Die Erwartungen an moderne Softwaresysteme steigen kontinuierlich. Während zu den Basisanforderungen seit etlicher Zeit Sicherheit, Skalierbarkeit und Erweiterbarkeit zählen, wird es immer wichtiger, dass Software dem Benutzer auch durch ein ansprechendes Äußeres (engl. user experience) entgegenkommt. Trotz gestiegener Erwartungen werden in der Softwareentwicklung jedoch immer kürzere Produkteinführungszeiten (engl. „time to market“) festgelegt.

Es ist daher erforderlich, dass bereits zu Beginn der Entwicklung entsprechende Entscheidungen getroffen werden, damit im weiteren Verlauf keine unerwarteten Überraschungen eintreten. Eine dieser wichtigen Entscheidungen ist die Wahl einer passenden Clienttechnologie, die die Entwicklung zeitgemäßer Benutzeroberflächen ermöglicht und gleichzeitig erprobte Entwicklungsmuster umsetzt.

Nicht immer wird diese Entscheidung durch bereits vorhandene Plattformen vorweggenommen. Oft ist das Projektteam gefragt, eine geeignete Technologie zu identifizieren und dabei unter einer Vielzahl von Optionen die Technologie auszuwählen, die projektspezifisch am besten passt.

Auf dem Markt existieren etliche Technologien, die es sich zum Ziel gesetzt haben die Entwicklung von Benutzerschnittstellen zu vereinfachen oder sogar zu revolutionieren und die sich teilweise stark voneinander unterscheiden. In dieser Arbeit werden daher zunächst einmal bewährte Methoden und Muster erläutert, die im Zusammenhang mit der Programmierung von Benutzerschnittstellen in den letzten Jahrzehnten entwickelt wurden.

Im Anschluss wird ein Kriterienkatalog aufgestellt, um die Leistungsfähigkeit der Rahmenwerke objektiv evaluieren zu können.

Bei den untersuchten Technologien, handelt es sich um die zwei Rahmenwerke *„Windows Forms“* und *„Windows Presentation Foundation*“ für die .Net Plattform, sowie die quelloffene, betriebssystemunabhängige *„NetBeans Plattform“*, die in der Java-Laufzeitumgebung läuft.

Die drei Technologien werden dahingehend untersucht, ob und wie sie die im Kriterienkatalog ermittelten Anforderungen erfüllen. Dazu wird oftmals mit Codebeispielen die Funktionsweise erläutert.

Abschließend werden Entscheidungshilfen gegeben, unter welchen Bedingungen der Einsatz welcher Technologie sinnvoll erscheint.

2 Grundlagen

In diesem Kapitel wird der Begriff Softwarearchitektur definiert und erläutert, wie der Softwareentwicklungsprozess Nutzen aus einer guten Architektur ziehen kann.

Anhand der weit verbreiteten Dreischichtenarchitektur wird gezeigt, wie eine logische Trennung des Systems in die Schichten Darstellung, Geschäftslogik und Datenquelle aussehen kann.

Des Weiteren werden Entwurfsmuster beschrieben, die die Entwicklung von modularen Benutzerschnittstellen ermöglichen. Dazu zählen die objektorientierten Verhaltensmuster, Kommando, Mediator und Beobachter sowie das Model-View-Controller (MVC) und Model-View-Presenter (MVP) Muster.

Mit dem MVP Muster wird beispielhaft ein einfacher Login Bildschirm realisiert.

2.1 Definition Softwarearchitektur

Die Forschung auf dem Gebiet der Softwarearchitektur entstand aus der Notwendigkeit heraus immer größere und komplexere Softwaresysteme beherrschen zu müssen. Bereits in den 60er und 70er Jahren beschäftigten sich die Veröffentlichungen der Computerwissenschaftler Parnas [1], Brooks [2] und Dijkstra [3] mit der Partitionierung und Strukturierung von Softwaresystemen.

Der Begriff Softwarearchitektur wurde jedoch erst in den 90er Jahren geprägt, als große Softwaresysteme die Regel wurden und das Thema weitreichende Anerkennung und Aufmerksamkeit in Forschung und Industrie fand.

Len Bass definiert in [4] „Softwarearchitektur“ wie folgt:

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

Gemäß dieser Definition[1] umfasst der Architekturentwicklungsprozess die Zerlegung des Systems in seine Hauptbestandteile auf oberster Ebene. Es werden die Architekturbausteine, deren Verantwortlichkeiten, Instanzen, Schnittstellen und die Interaktion untereinander definiert. Softwarearchitektur manifestiert somit die frühsten und wichtigsten Designentscheidungen für das Softwaresystem [5].

[1] Etliche weitere Definitionen sind unter http://www.sei.cmu.edu/architecture/start/definitions.cfm zu finden. (letzter Abruf 10.11.2010)

Nach Christiane Hofmeister [6] bildet Softwarearchitektur die Brücke zwischen Anforderungsanalyse und Implementierung. Als solche kommt sie nach der Definition der Anforderungen und vor dem Feindesign, der Implementierung, der Integration, und dem Test.

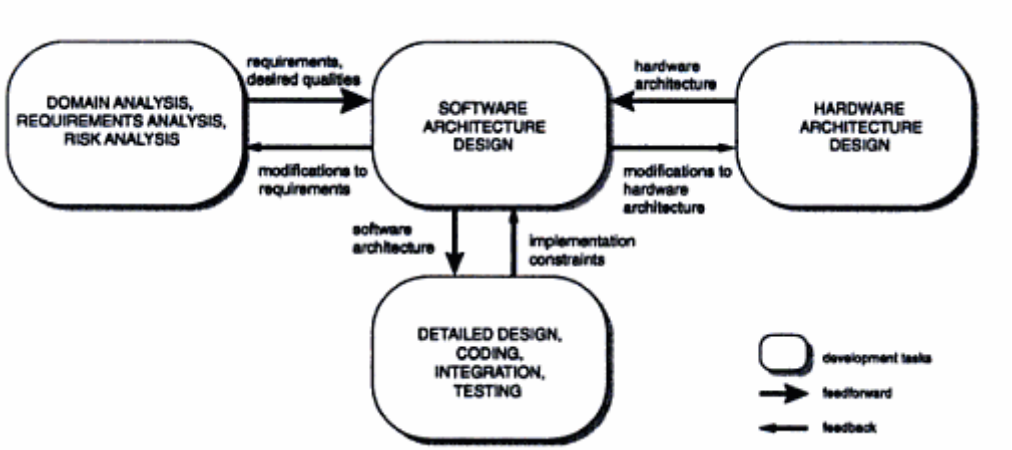


Abbildung 1: Softwarearchitektur – Brücke zwischen Anforderungsanalyse und Implementierung. Quelle: [6]

Len Bass [4] merkt des Weiteren an, dass es keine generell richtige Architektur gibt. Zur Bewertung muss sie immer im Kontext konkreter Ziele betrachtet werden. Eine gute Architektur erlaubt es dem System über einen langen Zeitraum hinweg, die Verhaltens-, Qualitäts- und Lebenszyklusanforderungen zu erfüllen.

2.2 Ziele einer Softwarearchitektur

Softwarearchitektur trägt zur Effizienz des Entwicklungsprozesses bei, indem Teilaufgaben voneinander entkoppelt und so Arbeitsteilung und eine flexible Projektorganisation ermöglicht wird. Sie hilft Risiken zu minimieren und hat somit großen Einfluss auf die erfolgreiche Durchführung des Softwareentwicklungsvorhabens. Bei guter Dokumentation dient sie als wiederverwendbarer Baustein für Folgeprojekte mit ähnlicher Problemstellung.

2.2.1 Effiziente Entwicklung

Die Erstellung von Software ist oftmals durch einen iterativ inkrementellen Entwicklungsprozess definiert. Würde im Vorfeld keine durchdachte Softwarearchitektur vorliegen, die dem System einen Rahmen gibt, so müsste für jede neue Anforderung, die inkrementell eingebaut wird, die Architektur zurechtgebogen werden. Dies impliziert, dass ab einer gewissen Größe allein für die Überarbeitung der bestehenden Software bereits beträchtlicher Aufwand betrieben werden muss.

Softwarearchitektur bildet daher das Fundament, um effektives, unabhängiges, verteiltes Arbeiten der Entwicklungsteams zu ermöglichen. Schließlich ist die Architektur ein Artefakt, das alle Informationen enthält, damit die Teams unabhängig voneinander in ihren speziellen Bereichen arbeiten können [7].

2.2.2 Risiken minimieren

Ein Softwareentwicklungsprojekt ist von einer Vielzahl von Risiken bedroht. Der Chaos Report 2009[2] kommt zu folgendem erschreckenden Ergebnis:

- *32% of all projects succeeding which are delivered on time, on budget, with required features and functions*
- *44% were challenged which are late, over budget, and/or with less than the required features and functions*
- *24% failed which are cancelled prior to completion or delivered and never used.*

Aufgabe von Softwarearchitektur ist es, Risiken in der Entwicklung frühzeitig zu erkennen und zu minimieren. Dies gelingt, indem Einflussfaktoren wie Qualitätsanforderungen (z.B. Sicherheit, Performanz) oder Managementanforderungen (z.B. Kosten- und Zeitvorgaben), die den Architekturdesignprozess beeinflussen, frühzeitig berücksichtigt werden.

2.2.3 Verständnis schaffen

Die Architektur als Softwareentwicklungsartefakt dient als Kommunikationsmedium zum einen zwischen Interessenvertretern aber auch zwischen allen Mitarbeitern des Projekts untereinander.

Die verschiedenen Sichten auf die Architektur ermöglichen es, dass viele Beteiligte mitreden können. Wäre die einzige Sicht auf Programmcodeebene, so wäre ein großer Personenkreis von der Diskussion ausgeschlossen. Findet eine solche Diskussion nicht statt, so läuft das Projekt Gefahr, dass wichtige Aspekte des Systems übersehen werden.

2.2.4 Kernwissen festhalten

Eine Architektur kann eine enorme Hebelwirkung für Systeme mit ähnlichen Anforderungen entwickeln. Unter gewissen Voraussetzungen lässt sich Architektur als Modell übertragen und wiederverwenden. Die Architektur wird damit zu einem wertvollen geistigen Eigentum des Unternehmens [7]. Würde bei der Entwicklung nur Programmcode aber keine dokumentierte Architektur entstehen, so wäre der Wissenstransfer auf zukünftige Projekte nur dann möglich, wenn die gleichen Personen, die über das Wissen verfügen, wieder an dem Projekt beteiligt sind.

Auch die Einarbeitung neuer Mitarbeiter gestaltet sich ohne dokumentierte Architektur weitaus schwieriger, wenn diese sich nur auf Programmcodeebene mit den Strukturen und Beziehungen des Systems befassen können. Die Lernkurve verläuft dann weitaus steiler. [7]

[2] Im Chaos Report [31] der Standish Group werden jährlich ca. 10.000 IT Projekte analysiert.

2.3 Aufbau einer mehrschichtigen Clientarchitektur

Die Schichtenbildung ist eine gebräuchliche Technik [5] in der Softwareentwicklung, um komplexe Systeme in logische Subkomponenten zu unterteilen. Dieses Kapitel befasst sich mit der weitverbreiteten Dreischichtenarchitektur, wie sie oft in Unternehmens-anwendungen angetroffen werden kann.

Es ist von entscheidender Bedeutung, dass einzelne Schichten nur Dienste darunterliegender Schichten in Anspruch nehmen und nicht umgekehrt, da sonst zyklische Abhängigkeiten entstehen. Zyklische Abhängigkeiten sind problematisch, da sie die Änderbarkeit und Wartung erschweren.

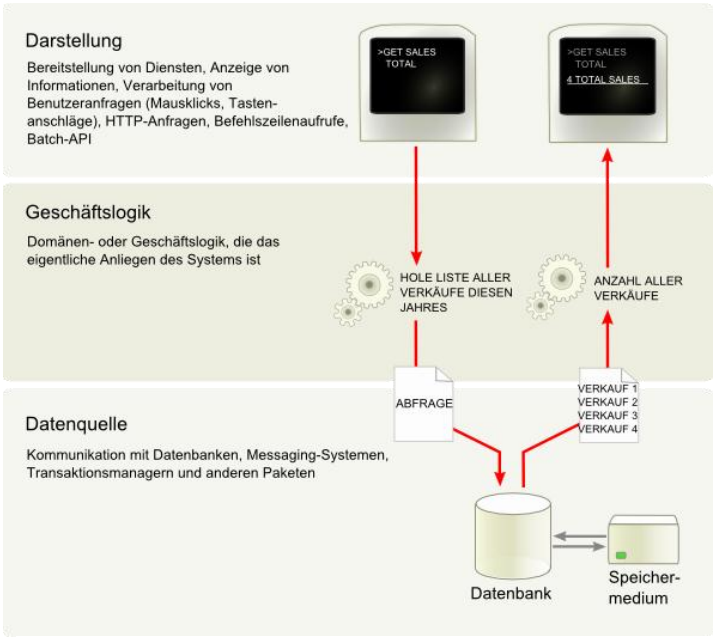


Abbildung 2: Drei-Schichten Architektur, Quelle: Wikipedia

2.3.1 Darstellung

Die Darstellungsschicht verarbeitet die Interaktion des Benutzers mit der Software über eine Benutzerschnittstelle. Konzeptionell ist es unerheblich, ob es sich um eine einfache Befehlszeile oder um eine graphische Benutzeroberfläche handelt.

2.3.2 Geschäftslogik

Diese Schicht stellt den Anwendungskern dar und umfasst Aufgaben, die von der Anwendung geleistet werden müssen. Dazu zählen Berechnungen, die auf Eingaben und gespeicherten Daten beruhen, die Prüfung von Daten, die von der Darstellungsschicht geliefert werden, sowie die Ansteuerung der Datenquelle, aufgrund von Befehlen, die aus der Präsentationsschicht stammen. In dieser Schicht liegt die eigentliche Fachlichkeit der Anwendung.

2.3.3 Datenquelle

Die Datenquelle behandelt die Kommunikation mit anderen Systemen, die für die laufende Anwendung spezielle Aufgaben ausführen. Dabei kann es sich um Transaktionsmonitore, andere Anwendungen, Messagingsysteme usw. handeln. Bei den meisten Unternehmensanwendungen besteht die größte Komponente der Datenquelle aus einer Datenbank, die hauptsächlich für die Speicherung persistierter Daten zuständig ist [5].

Entscheidend ist, dass weder Geschäftslogik noch Datenquelle Kenntnis von der Darstellungsschicht haben, damit sich diese leicht austauschen lässt, ohne dass am zu Grunde liegenden System etwas verändert werden muss.

2.4 Programmierparadigmen

2.4.1 Separation of concerns – SoC

Der Begriff Separation of concerns wurde 1974 von Edsger W. Dijkstra [8] geprägt und sagt aus, dass Software modular aufgebaut sein sollte. Dies hat den Vorteil, dass jedes Modul nur eine kleine klar definierte Aufgabe innerhalb des Gesamtsystems übernimmt und somit leichter zu verstehen und zu warten ist.

2.4.2 Don‘t repeat yourself – DRY

Das DRY Paradigma [9] bezieht sich auf die Vermeidung von dupliziertem Code, also Codefragmenten, die an mehreren Stellen gleich oder sehr ähnlich umgesetzt sind.

Redundant vorhandener Code ist schwierig zu pflegen, da die Konsistenz zwischen den einzelnen Duplikaten gewährleistet sein muss. Bei Systemen, die dem DRY-Prinzip treu bleiben, müssen hingegen Änderungen nur an einer Stelle vorgenommen werden.

2.4.3 Keep it simple and stupid – KISS

Das KISS Prinzip sagt aus, dass Einfachheit ein Hauptziel im Design eines Softwaresystems sein sollte, um unnötige Komplexität zu vermeiden.

2.4.4 You ain’t gonna need it – YAGNI

YAGNI ist ein Prinzip des eXtreme Programmings[3] und verlangt von dem Entwickler, dass Funktionen erst implementiert werden, wenn sie auch wirklich benötigt werden.

Ron Jeffries schreibt, *“Always implement things when you actually need them, never when you just foresee that you need them.”*[4]

---

[3] Softwareentwicklungsmethode, die von Kent Beck im Zuge des „Chrysler Comprehensive Compensation System“ (C3) Abrechnungsprojekts entwickelt wurde.

[4] http://www.xprogramming.com/Practices/PracNotNeed.html (letzter Abruf 10.11.2010)

2.5 Allgemeine Entwurfsmuster

Der Architekt Christoph Alexander beschreibt Muster wie folgt:

„Jedes Muster beschreibt ein in unserer Umwelt beständig wiederkehrendes Problem und erläutert den Kern der Lösung für dieses Problem, so daß Sie diese Lösung beliebig oft anwenden können, ohne sie jemals ein zweites Mal gleich auszuführen.“ [10]

Auch wenn Christoph Alexander von dem Bau von Häusern und Städten spricht, lässt sich der Vorsatz wiederverwendbarer Muster auch auf die Softwareentwicklung übertragen. Erich Gamma et. al. haben mit dem Buch „Design Patterns – Elements of Reusable Object-Oriented Software“ [11] diese Arbeit bereits erledigt und eben solche wiederkehrende Muster für die objektorientierte Softwareentwicklung niedergeschrieben. Die Autorengruppe wird oft auch als Gang of Four (GoF) bezeichnet.

Als nächstes werden drei Muster vorgestellt, die in der Entwicklung von Benutzerschnittstellen Verwendung finden.

2.5.1 Beobachter

Das Beobachtermuster ist ein objektbasiertes Verhaltensmuster, das eine 1-zu-n Abhängigkeit zwischen Objekten definiert. Eine Änderung an einem Objekt führt dazu, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden.

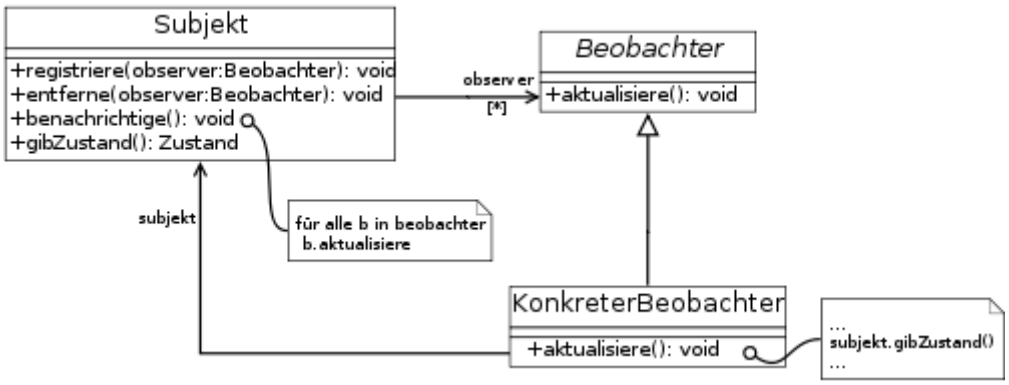


Abbildung 3: Beobachter Muster, Quelle: Wikipedia

Das beobachtete Subjekt bietet einen Mechanismus an, um Beobachter an- und abzumelden und diese über Änderungen zu informieren. Es kennt alle seine Beobachter nur über die (überschaubare) Schnittstelle Beobachter. Änderungen können so völlig unspezifisch an jeden angemeldeten Beobachter publiziert werden, ohne dass die weitere Struktur dieser Komponenten bekannt sein muss.

Die Beobachter implementieren ihrerseits die Methode aus der Beobachter Schnittstelle, um auf die Änderungen zu reagieren. In der Regel werden die für eine Komponente relevanten Teile des Zustands abgefragt. Es können aber auch die Änderungen proaktiv als Parameter an die aktualisiere-Methode weitergereicht werden.

In der Programmierung von Benutzerschnittstellen lässt sich dieses Muster nutzen, um mehrere Darstellungsformen mit einem zu Grunde liegenden Model zu verknüpfen. Das Modell kann dann bei Änderung seiner Daten, die angemeldeten Views benachrichtigen.

2.5.2 Mediator

Auch das Mediatormuster ist ein objektbasiertes Verhaltensmuster. Es steuert das kooperative Verhalten von Objekten, wobei die Objekte nicht direkt, sondern über einen Vermittler kooperieren. Die Objekte, die miteinander kooperieren möchten, werden als Kollegen bezeichnet.

Durch den Vermittler wird eine Schnittstelle zur Kommunikation mit Kollegen definiert. Der Konkrete Vermittler implementiert das kooperative Verhalten durch Koordination der beteiligten Kollegen. Er kennt und verwaltet beteiligte Kollegen. Die einzelnen Kollegen kennen ihren Vermittler und kommunizieren nur mit ihm statt mit anderen Kollegen. Dies reduziert die Anzahl ihrer Beziehungen untereinander.

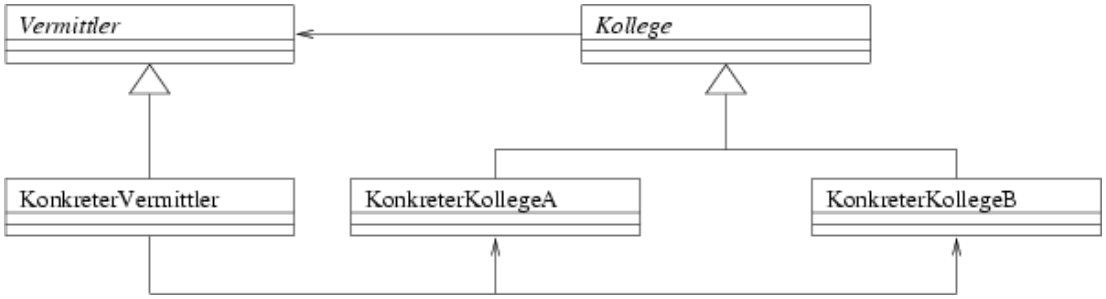


Abbildung 4: Mediator-Muster, Quelle. Wikipedia

In Benutzerschnittstellen lässt sich ein Vermittler nutzen, um in einem Dialog mit mehreren Kontrollelementen eine lose gekoppelte nachrichtenbasierte Kommunikation zwischen den Elementen zu etablieren.

2.5.3 Kommando

Entsprechend der „Gang of Four“ Klassifizierung ist das Kommandomuster ein objektbasiertes Verhaltensmuster. Es dient dazu, einen Befehl in einem Objekt zu kapseln. Dieses Objekt erhält alle Informationen, die für den Aufruf einer Methode nötig sind.

Die Implementierung des Musters sieht vor, dass eine abstrakte Klasse „Befehl“ eine abstrakte „führeAus-Methode“ definiert. Konkrete Befehle bilden Unterklassen dieser abstrakten Klasse und enthalten den Empfänger der Operation als Zustand. Die überschriebene „führeAus-Methode“ delegiert die Ausführung der Operation an das Empfängerobjekt. Der Aufrufer besitzt einen oder mehrere Verweise auf Befehle und fordert diese bei Bedarf auf, ihre Aktion auszuführen.

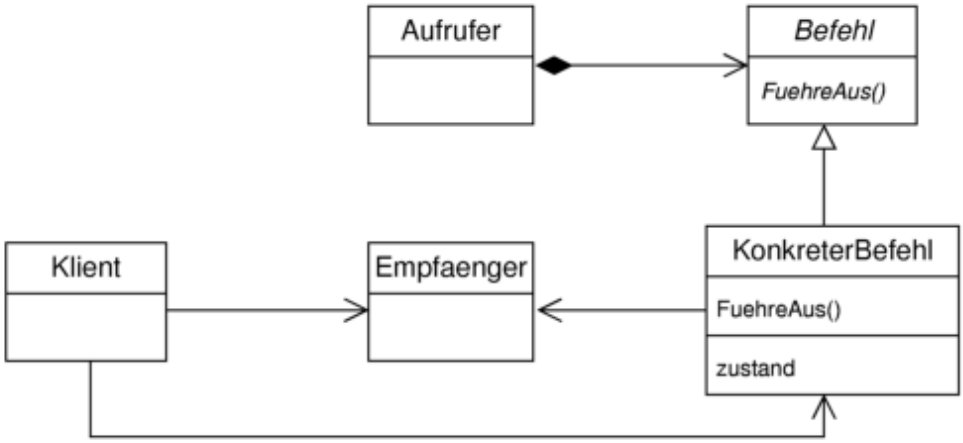


Abbildung 5: Befehl Muster, Quelle: Wikipedia

Dieses Muster ist insbesondere bei der Programmierung von Benutzungsschnittstellen hilfreich. Eine Klassenbibliothek für Benutzerschnittstellen stellt beispielsweise einen Menüeintrag als generisches Steuerungselement zur Verfügung. Die Bibliothek kann jedoch nicht wissen was bei einem Klick auf den Eintrag geschehen soll. Der Menüeintrag wird dann mit dem Exemplar einer konkreten Befehlsunterklasse verknüpft. Soll zusätzlich zu dem Menüeintrag auch ein Button die Operation ausführen, so nutzt er einfach dasselbe Befehlsexemplar wie der Menüeintrag. Der eigentliche Code, der ausgeführt werden soll, muss also nicht für jedes Steuerelement dupliziert werden. Dies entspricht dem DRY-Prinzip.

Die große Flexibilität des Befehlsmusters entsteht dadurch, dass das Objekt, das die Aktion auslöst, vollkommen von dem Objekt losgelöst ist, welches weiß wie der Befehl ausgeführt werden soll. Durch diese Entkopplung ist es ebenso möglich, kontextsensitiv Menüeinträge ein- und auch wieder auszublenden, indem Befehlsobjekte dynamisch gesetzt werden.

Werden die Befehle, die ein Benutzer aufruft mitprotokolliert, lässt sich zudem eine Undofunktion realisieren. Dafür muss jedoch im Befehlsobjekt ebenfalls der relevante Zustand für die Umkehr des Befehls enthalten sein.

2.6 Benutzerschnittstellen-Muster

Lange Zeit wurden Anwendungen mit sogenannten RAD (Rapid Application Development) Werkzeugen wie VB, Powerbuilder oder Delphi entwickelt. Mit diesen Werkzeugen war es besonders leicht, datenintensive Anwendungen zu erstellen, da sie über SQL-fähige Steuerelemente verfügen [5]. Die Funktionalität durfte jedoch nicht über die Anzeige und einfache Aktualisierung relationaler Daten hinausgehen. Die Probleme entstanden mit komplexerer Anwendungslogik (Geschäftsregeln, Datenprüfungen, Berechnungen, etc.), die normalerweise direkt in die Formulare eingefügt wurde. Sie ließ sich in der Regel nicht mehrfach verwenden, was zur Folge hatte, dass der Code dupliziert wurde. Selbst einfache Änderungen bedeuteten die Suche nach Codesegmenten in vielen Bildschirmformularen.

Mit Hilfe von Entwurfsmustern für Benutzerschnittstellen wird versucht, diese enge Kopplung zwischen Darstellung und Logik zu entzerren. Dies hat den Vorteil, dass sich die Logik wieder verwenden lässt und die Form der Darstellung leichter ausgetauscht werden kann.

2.6.1 Model-View-Controller

Der norwegische Computerwissenschaftler Trygve Reenskaug formulierte 1979 das Model-View-Controller Architekturmuster für die Strukturierung von Anwendungen mit graphischen Benutzerschnittstellen [12]. Die ursprüngliche Implementierung erfolgte in Smalltalk-80.

Reenskaug teilt eine Anwendung demnach in drei Bestandteile auf, um eine modulare Architektur der Anwendung zu ermöglichen.

Das Model:

Das Model ist die domänenspezifische Repräsentation der Daten, mit denen eine Anwendung arbeitet. Die Geschäftslogik versieht die Daten mit einer Bedeutung, in dem sie Operationen zur Manipulation der Daten zur Verfügung stellt. Des Weiteren verwaltet das Model Beobachter von Daten und informiert diese, sobald sich Änderungen an den Daten ergeben. Das Model darf keine direkte Kenntnis und somit Abhängigkeit von möglichen Views haben, sondern ausschließlich über das Beobachtermuster Veränderung nach außen propagieren. Auf diese Weise ist es möglich mit verschiedenen Views unterschiedliche Sichten auf zugrundeliegende Daten zu ermöglichen.

Die Views:

Der View stellt einen der zuvor angesprochenen Beobachter dar. Seine Aufgabe ist es, Inhalte des Models so zu visualisieren, dass ein Benutzer mit ihnen in Interaktion treten kann.

Der Controller:

Der Controller verwaltet einen oder auch mehrere Views, nimmt die Benutzerinteraktion entgegen und ruft entsprechende Dienste des Models zur Manipulation der Daten auf. Es ist jedoch nicht Aufgabe des Controllers Daten direkt zu manipulieren.

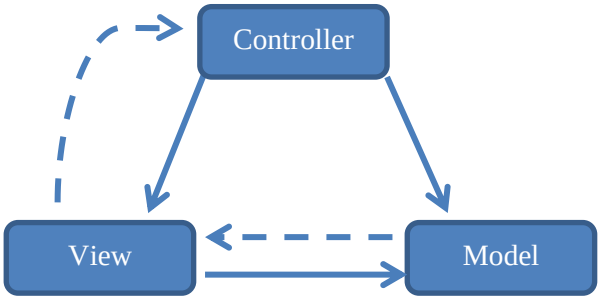


Abbildung 6: Das Model-View-Controller Muster

Das Diagramm zeigt die Beziehung zwischen Model, View und Controller. Durchgezogene Linien stellen eine direkte Assoziation dar. Die gestrichelten Linien bedeuten eine indirekte Kommunikation über das Beobachtermuster.

Dieses Muster erhöht massiv die Wiederverwendbarkeit einzelner Komponenten. Beispielsweise ist es möglich, eine Anwendung zu schreiben, die das gleiche Modell benutzt, aber einerseits eine Windows/Linux-Oberfläche realisiert, andererseits aber auch eine Weboberfläche beinhaltet. Beides basiert auf dem gleichen Modell, nur Controller und View müssen dabei jeweils neu konzipiert werden.

2.6.2 Model-View-Presenter

Das Model-View-Presenter Muster ist eine modifizierte Form des zuvor beschriebenen MVC-Musters und wurde in den frühen 90er Jahren von der Firma Taligent entwickelt. 2006 wurde dieses Muster von Microsoft durch ausführliche Dokumentation und Beispiele als Herangehensweise zur Programmierung von Benutzerschnittstellen in der .Net Umgebung mit Windows Forms favorisiert [13].
Ein großer Nachteil des MVC-Musters ist die schlechte Testbarkeit der Implementierung, da der Controller, der die Verbindung zwischen Model und View herstellt, Kenntnis sowohl über das Model als auch über den View haben muss. Durch diese enge Kopplung lässt er sich ohne View nicht testen. Diese Unzulänglichkeit wird durch das MVP Muster beseitigt, sodass automatisierte Tests auch das Bindeglied (Presenter) zwischen Model und View auf ordnungsgemäße Funktion hin überprüfen können.
Ein weiterer Vorteil der Entkopplung liegt in der Möglichkeit, die Logik des Presenters wiederzuverwenden. Da der View zuvor durch ein Interface spezifiziert wurde, kann der Presenter jede Form von Ausgabe technologieunabhängig ansteuern.
Im Folgenden wird Martin Fowlers Interpretation des MVP Musters mit einem Passive View [14] näher erläutert.
Ein Passive View reduziert das Verhalten der Steuerelemente auf ein absolutes Minimum. Im Gegenzug übernimmt der Presenter allein die Aufgabe auf Userevents zu antworten und den View entsprechend zu aktualisieren. Auf diese Weise liegt die gesamte Logik des View im Presenter, so dass dieser gut zu testen ist. Der passive View fungiert als einfacher Konsument ohne eigenes Verhalten und hat keinerlei Abhängigkeiten vom Model.
Interaktion mit dem Model sowie Aktualisierung des Views wird exklusiv durch den Presenter realisiert.
Das Ergebnis ist, dass die Ansicht nur einfachste Logik zur Darstellung beinhaltet und keine Logik zur Synchronisation von Daten. Dadurch wird der Quelltext der Darstellungsschicht sehr schlank und maximal portabel. Die fehlerfreie Arbeit des Presenters, lässt sich auf herkömmlichem Wege mit Unittests verifizieren.

Zusammenfassend lässt sich das MVP-Muster wie folgt charakterisieren:

- Der View enthält ein Presenter Exemplar (view „kennt“ presenter)
- Der Presenter ist die einzige Klasse, die weiß wie Daten aus dem Model extrahiert werden können
- Der Presenter kommuniziert mit dem View durch ein View Interface (abstrakte Repräsentation des View ohne UI-Technologie spezifische Eigenschaften)
- Der View weiß nichts von der Existenz des Model
- Das Model kennt weder View noch Presenter

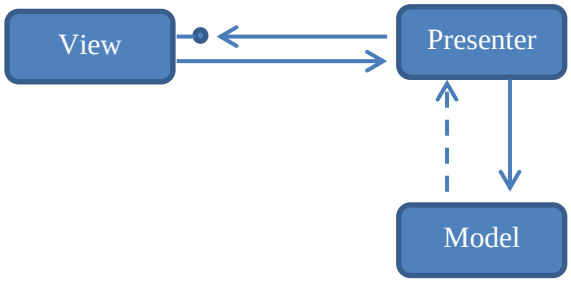


Abbildung 7: Das Model-View-Presenter Muster

Beispielhafte Implementierung
Um das Muster zu verdeutlichen wird in diesem Kapitel ein einfacher Login Dialog implementiert. Durch einen Klick auf den Button wird die eingegebene Benutzerkennung mit der im Modell hinterlegten Kennung abgeglichen. Ist die Kennung falsch oder war eines der Textfelder leer, so wird eine Fehlermeldung angezeigt. Auf der beigefügten CD befindet sich die vollständige lauffähige Visual Studio Projektmappe.

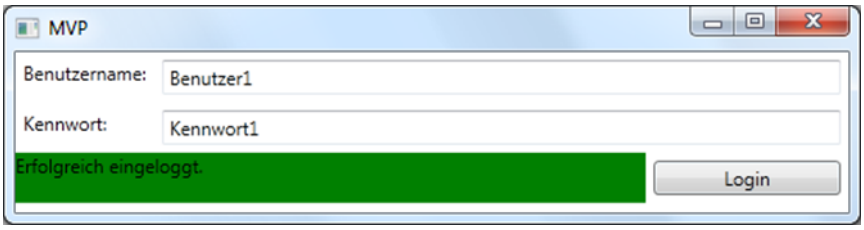


Abbildung 8: Login Bildschirm mit erfolgreicher Anmeldung

Das View Interface stellt die Abstraktion des Dialogs dar. Die Eigenschaften `Benutzername` und `Kennwort` stellen, den Wert der beiden Textfelder dar. Mit den Methoden Erfolg und Fehler wird in der Oberfläche signalisiert, ob der Loginvorgang erfolgreich war oder nicht.

C# Code

```
public interface IView
{
    string Benutzername { set; get; }

    string Kennwort { set; get; }

    void Fehler(string fehlermeldung);

    void Erfolg(string erfolgsmeldung);
}
```

Listing 1: MVP Muster – View Interface

Die Klasse MainWindow stellt die konkrete Implementierung des Viewinterface dar. Der View kennt seinen Presenter, damit Benutzerereignisse weitergeleitet werden können.

C# Code

```
public partial class MainWindow : Window, IView
{
    private ViewPresenter _presenter;

    public MainWindow()
    {
        InitializeComponent();
        _presenter = new ViewPresenter(this);
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        _presenter.Anmelden();
    }

    string Benutzername
    {
        get { return benutzername.Text; }
        set { benutzername.Text = value; }
    }

    string Kennwort
    {
        get { return kennwort.Text; }
        set { kennwort.Text = value; }
    }

    public void Fehler(string fehler)
    {
        status.Text = fehler;
        status.Background = Brushes.Red;
    }
```

```
    public void Erfolg(string erfolg)
    {
        status.Text = erfolg;
        status.Background = Brushes.Green;
    }
}
```

Listing 2: MVP Muster – View

Der Presenter ist dafür zuständig, auf Anfrage die Eingaben aus dem View abzurufen und per Delegation mit Hilfe des Models zu verifizieren. Da er den View nur unter der technologieunabhängigen Schnittstelle `IView` kennt, lässt er sich auch mit anderen Darstellungsformen wiederverwenden.

C# Code

```
class ViewPresenter
{
    IView _view;
    IModel _model;

    public ViewPresenter(IView view)
    {
        _view = view;
        _model = new Model();
    }

    internal void Anmelden()
    {
        if (String.IsNullOrEmpty(_view.Benutzername)
            || String.IsNullOrEmpty(_view.Kennwort))
        {
            _view.Fehler("Die Textfelder Benutzername und Kennwort müssen
                                ausgefüllt sein.");
            return;
        }
        if (_model.überprüfeBenutzerkennung(_view.Benutzername,
                                                _view.Kennwort))
        {
            _view.Erfolg("Erfolgreich eingeloggt.");
        }
        else
        {
            _view.Fehler("Login nicht erfolgreich.");
        }
    }
}
```

Listing 3: MVP Muster – Presenter

Das Model stellt die Anwendungslogik dar und kennt weder den Presenter noch den View.

C# Code

```
interface IModel
{
    bool überprüfeBenutzerkennung(string name, string kennwort);
}

class Model : IModel
{
    Dictionary<string, string> benutzer =
        new Dictionary<string, string>();

    public Model()
    {
        benutzer.Add("Benutzer1", "Kennwort1");
        benutzer.Add("Benutzer2", "Kennwort2");
    }


    bool überprüfeBenutzerkennung(string name, string kennwort)
    {
        string korrektesKennwort;
        benutzer.TryGetValue(name, out korrektesKennwort);

        return korrektesKennwort == kennwort;
    }
}
```

Listing 4: MVP Muster – Model

Fehlermeldung bei falscher Benutzerkennung

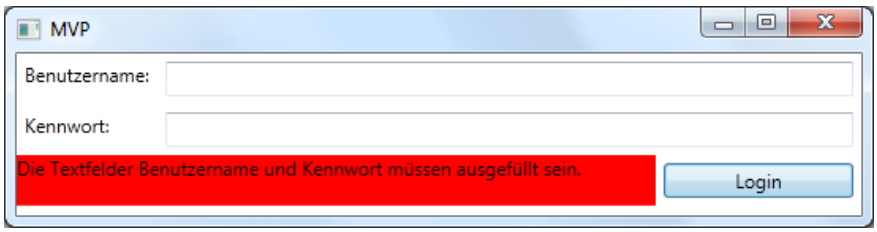


Abbildung 9: Login Bildschirm mit fehlgeschlagener Anmeldung 1

Fehlermeldung bei nicht ausgefüllten Textfeldern

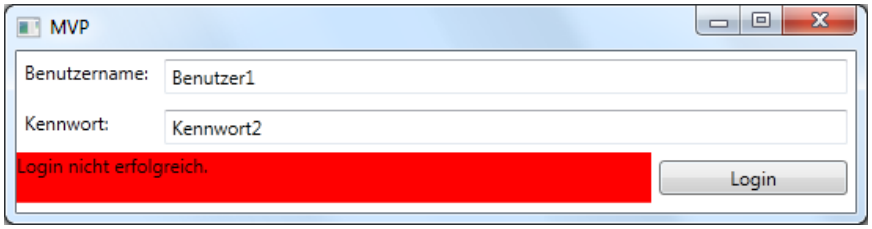


Abbildung 10: Login Bildschirm mit fehlgeschlagener Anmeldung 2

3 Kriterienkatalog zur Bewertung von Clienttechnologien

In diesem Kapitel wird ein Kriterienkatalog erarbeitet, um die drei Technologien objektiv evaluieren zu können und somit eine direkte Vergleichbarkeit herzustellen.

3.1 Editorunterstützung in der Entwicklungsumgebung

Die drei Technologien sind alle eng mit einer integrierten Entwicklungsumgebung verknüpft. Mit diesem Kriterium wird bewertet, wie der Entwickler durch die Werkzeuge der Entwicklungsumgebung unterstützt wird. Dabei wird ausschließlich die Rolle des visuellen Editors für Benutzerschnittstellen in die Bewertung mit aufgenommen und kein Augenmerk auf die Möglichkeiten des Codeeditors gelegt. Für die Evaluierung sind folgende Fragestellungen relevant:

- Können Komponenten bequem per Drag & Drop platziert werden? Die Komponenten müssen des Weiteren so konfiguriert werden können, dass sie sich an anderen Komponenten ausrichten können, um ein dynamisches Verhalten beim Maximieren und Minimieren des Fensters zu erhalten.
- Ist die Bedienung intuitiv genug, um eine schnelle Einarbeitung zu Beginn als auch eine rasche Abwicklung immer wiederkehrender Aktionen zu gewähren?
- Ist es möglich die Eigenschaften (Beschriftung Größe, Farbe, Events) von graphischen Komponenten in übersichtlicher Form tabellarisch einzusehen und verändern zu können?
- Lassen sich Komponenten von Drittanbietern nahtlos in die Entwicklungsumgebung integrieren?
- Gerade im kommerziellen Bereich gibt es oftmals mehrere Versionen mit einem variierenden Funktionsumfang, die sich entsprechend im Preis unterscheiden. Dieser Punkt bewertet das Preis/Leistungsverhältnis sowie die Lizensierungsmöglichen. Opensourcelösungen sind von diesem Punkt nicht betroffen.

3.2 Trennung von Darstellung und Logik

Wie bereits in Kapitel 2.3 erläutert, ist es erstrebenswert die Darstellung von der Logik zu trennen. Für die verwendete Technologie bedeutet dies, dass Mechanismen und Entwurfsmuster umgesetzt sein müssen, um die Kopplung von Benutzerschnittstelle und Anwendungslogik lose zu gestalten. Insbesondere bei großen komplexen Anwendungen erfordert dies mitunter einen erheblichen Aufwand. Unter diesem Punkt wird ermittelt, welche Bordmittel zur Verfügung stehen, um die Anwendung diesbezüglich zu strukturieren.

3.3 Arbeitsteilung von Designern und Entwicklern

In den letzten Jahren ist zu beobachten, dass Benutzer an Software immer höhere Anforderungen stellen. Waren früher Merkmale wie Performanz und Bedienbarkeit von besonderer Bedeutung, kommt heutzutage noch das Bedürfnis für eine gesteigerte Nutzererfahrung (engl. User Experience, UX) hinzu. Dabei kommt es auch auf die Eleganz des Produkts an, so dass es den Nutzern eine Freude ist, das Produkt zu besitzen oder zu benutzen. Auch die International Organization for Standardization hat den UX-Begriff bereits standardisiert.[5]

Vorreiter auf diesem Gebiet ist die Firma Apple Inc. die beginnend mit dem iPod eine Hard- und Software umfassende Designrevolution einleitete und derzeit mit weiteren Produkten wie iPhone und iPad anführt. Bei all diesen Produkten ist zu bemerken, dass dort eine erhebliche Menge an Zeit in die Gestaltung des Bedienkonzeptes sowie der Optik investiert wurde. Diese Bewegung hat sogar einen neuen Arbeitsmarkt hervorgebracht, der sich gezielt mit dem Erstellen solcher Bedienerlebnisse beschäftigt[6].

Um eine benutzerzentrierte Gestaltung zu ermöglichen, muss es möglich sein, dass sich UX-Designer mit spezieller Software in den Entwicklungsprozess einklinken, ohne das Entwickler, die für die reine Anwendungslogik verantwortlich sind, in ihren Arbeitsabläufen behindert werden.

---

[5] ISO 9241-210: Ergonomics of human system interaction — Part 210: Human-centered design for interactive systems. ISO, Genf, 2010.

[6] http://www.uxjobs.org/ (letzter Abruf 10.11.2010)

3.4 Modularität

Die Größe und Komplexität moderner Anwendung erhöht sich mit der Zeit stetig. Gleichzeitig wird von professionellen Anwendungen verlangt, dass sie so flexibel sind, dass Erweiterungen schnell und einfach realisiert werden können.

Aus diesem Grund ist es wünschenswert, Anwendungen in verschiedene logische Einheiten zu zerlegen, sodass eine modulare Architektur entsteht. Module erzwingen Separation of Concerns und vereinfachen die Trennung von Benutzerschnittstelle und Geschäftslogik.

Um die modulare Entwicklung zu unterstützen sollten Frameworks folgende Möglichkeiten bieten:

- Ordnungsgemäß definierte Module haben eine hohe interne Kohäsion und eine lose Kopplung untereinander. Die Kopplung untereinander geschieht durch wohldefinierte Schnittstellen im Gegensatz zu direkten Referenzen.
- Module können unabhängig voneinander entwickelt, getestet und verteilt werden.
- Module sollen sich zur Laufzeit dynamisch hinzufügen und auch wieder entfernen lassen.
- Module können aus unterschiedlichen Ressourcen geladen werden. Dabei spielt es keine Rolle, ob es sich bei der Ressource um das Web, das Dateisystem oder eine Datenbank handelt.
- Sofern eine Anwendung nicht lokal auf dem System des Benutzers installiert ist, sondern über das Netz nachgeladen werden muss, verringern Module die Übertragungszeit der Anwendung, da nur die für den Start benötigten Module geladen werden. Die übrigen Module werden im Hintergrund geladen und initialisiert, sobald sie benötigt werden. Diesen Vorgang nennt man „lazy loading“.
- Um die Startzeit der Anwendung zu verkürzen, werden nur die für den Start zwingend erforderlichen Module geladen und initialisiert.

Ein Clientframework muss in der Lage sein, die Abhängigkeiten der einzelnen Module aufzulösen und so die Anwendung aus den vielen Modulen dynamisch zusammenzubauen.

3.5 Testbarkeit

Das Testen von graphischen Benutzeroberflächen stellt den Entwickler vor zweierlei schwerwiegende Probleme [15].

- Die Anzahl möglicher Operationen einer Anwendung mit graphischer Benutzeroberfläche ist in der Regel sehr hoch. Beispielsweise hat die simple Anwendung Microsoft WordPad bereits 325 mögliche GUI Operationen [16].
- Des Weiteren setzt die Ausführbarkeit mancher Operationen eine komplexe Abfolge von Benutzerschnittstellenereignissen voraus.
  Beispiel: Um eine Datei zu öffnen, muss der Anwender auf das Dateimenu klicken, den Menüeintrag „Öffnen“ auswählen und im sich anschließend öffnenden Dialogfeld muss ein Dateiname angeben werden. Zuletzt muss der Fokus der Anwendung auf das neue Fenster wechseln.
  Solche komplexen Abhängigkeiten lassen sich oftmals nur mit endlichen Zustandsautomaten modellieren [17].

Unter diesem Kriterium soll ermittelt werden, welche Möglichkeiten für programmatische Tests zu Verfügung stehen. Voraussetzung ist dabei wieder einmal die saubere Trennung von Darstellung und Logik.

Es werden keine Werkzeuge betrachtet, die durch das Aufzeichnen von Makros das Verhalten eines Benutzers simulieren, sondern ausschließlich automatisierte programmatische Testverfahren, wie sie von Modultest durch die Unit-Test Rahmenwerke bekannt sind.

3.6 Technische Funktionen

Die Technologien sollten die folgenden Funktionen leicht nutzbar zur Verfügung stellen, um ein schnelle und moderne Entwicklung zu ermöglichen.

3.6.1 Generische Steuerelemente

Als Steuerelemente bezeichnet man Komponenten, die dem Anwender auf einer Benutzeroberfläche Interaktion erlauben oder Informationen anzeigen. Dies reicht von einfachen Komponenten wie Textfeldern oder Buttons bis hin zu komplexen Strukturen wie Datagrids, die in tabellarischer Form Daten anzeigen und mit beliebigen Datenquellen verknüpft werden können.

Des Weiteren gehören zu den Steuerelementen auch Layout Manager, die mehrere Komponenten enthalten und anordnen. Dies ist weitaus komfortabler und flexibler als das Positionieren mittels fester Koordinaten.

Ein Framework muss über eine Vielzahl solcher Steuerelemente verfügen, damit es sich in diversen Kontexten und Anwendungsszenarien einsetzen lässt und darüber hinaus eine Grundlage für das Erstellen eigener Steuerelemente bieten.

3.6.2 Look & Feel

Wie bereits in Kapitel 3.3 erwähnt, wird von modernen Anwendungen auch ein ansprechendes Äußeres erwartet. Mit austauschbarem Look & Feel ist gemeint, dass im Handumdrehen das Erscheinungsbild der gesamten Anwendung verändert werden kann, ohne das an der Gesamtstruktur der Anwendung Änderungen vorgenommen werden müssen. Dies ist vergleichbar mit Cascading Stylesheets[7], mit denen HTML Elemente gestaltet werden können.

3.6.3 Ereignisse/Befehle

Ereignisse werden von Elementen der Bedienoberfläche ausgelöst, wenn beispielsweise ein Button geklickt wird oder in einem Textfeld eine Eingabe erfolgt. Solche Ereignisse stellen auf einem sehr niedrigen Niveau einen Benachrichtigungsmechanismus ähnlich dem Beobachtermuster zur Verfügung und gehören zur Standardausrüstung von GUI-Bibliotheken.

In komplexeren Anwendungen sollte die Funktionalität in Befehlen organisiert sein, die eine höhere Abstraktion bieten. Einmal zentral definierte Befehle können durch viele verschiedene Aktionen und Benutzerschnittstellenelemente wie Menüs, Kontextmenüs, Tastaturkürzel, etc. ausgelöst werden, ohne das redundanter Code zur Ereignisbehandlung geschrieben werden muss[8]. Des Weiteren verwaltet ein Befehl auch den Zustand der Benutzerschnittstelle, indem er kontextabhängig aktiviert oder deaktiviert wird. Die zentrale Definition von Befehlen erleichtert zudem die Lokalisierung.

3.6.4 Datenbindung & Validierung

Mit Datenbindung bezeichnet man den Vorgang Informationen aus Datenbanken oder Objekten zu extrahieren, um sie auf der Benutzeroberfläche anzuzeigen. Das Schreiben solchen Codes ist oftmals mühselig und fehleranfällig und sollte daher durch die Benutzerschnittstellenbibliothek vereinfacht werden.

Validierung ist eine Funktionalität, die oftmals im Zusammenhang mit Datenbindung realisiert wird, um fehlerhafte Eingaben zu entdecken und zu kennzeichnen oder gar nicht erst zuzulassen.

3.6.5 Unterstützung moderner Bedienkonzepte

Während Tastatur und Maus seit Jahrzehnten zur Standardinteraktion mit graphischen Benutzerschnittstellen zählen, hat die Forschung in den letzten Jahren weitere Bedienkonzepte hervorgebracht. Dazu zählen Drag & Drop (Ziehen und Fallenlassen von

---

[7] http://de.wikipedia.org/wiki/Cascading_Style_Sheets (letzter Abruf 10.11.2010)

[8] Dieses Vorgehen folgt dem DRY-Prinzip. Siehe [9].

Elementen auf dem Bildschirm), Unterstützung von Touchscreens[9] und der dazugehörigen Gestenerkennung sowie neuartige Steuerelemente, wie sie beispielsweise Microsoft mit der Ribbonoberfläche in Office 2007 eingeführt hat.

3.6.6 Barrierefreiheit

Laut UN Berichten[10] gibt es weltweit ca. 650 Millionen Menschen mit Behinderungen. Eine moderne Clienttechnologie, muss auch diesen Menschen gerecht werden und daher Konzepte der Barrierefreiheit unterstützen.

3.6.7 Internationalisierung

Anwendungen sollen sich problemlos in andere Sprachen übersetzen lassen. Damit das möglich ist, müssen sich die in der Anwendung befindlichen Zeichenketten auslagern lassen, damit sie übersetzt werden können. Dies umfasst auch die länderspezifische Formatierung von Währungs-, Datums- oder Zeitangaben.

3.6.8 Softwareverteilung

Die Technologie muss eine Infrastruktur vorweisen, um die Arbeitsabläufe der Softwareverteilung zu vereinfachen. Die Softwareverteilung umfasst die Erstellung von Installationspaketen, sowie die Versorgung mit Produktaktualisierungen.

3.7 Angebote von Drittanbietern

Unter diesem Punkt wird eine Marktübersicht über spezialisierte Drittanbieter gegeben, die den Funktionsumfang der Technologie mit eigenen Produkten erweitern.

3.8 Hilfe und Support

Unter diesem Punkt werden die Hilfestellungen ermittelt die Entwickler bei der Nutzung der konkreten Technologie unterstützen. Dazu zählen die Qualität der Dokumentation, die Größe der Entwicklergemeinde sowie die Betreuung durch den Hersteller.

Des Weiteren wird versucht die Komplexität zu bewerten und den ungefähren Einarbeitungsaufwand abzuschätzen. Da dieser aber von diversen individuellen Faktoren, wie Erfahrung, verfügbarer Zeit und Auffassungsgabe des Entwicklers abhängig ist, sollte man diesen Punkt mit Vorsicht genießen.

---

[9] Genau genommen ist Multitouch die Aufgabe des Betriebssystems, dass die Eingabe erkennen und an die Anwendung weiterleiten muss. Nichtsdestotrotz, muss das Framework auf eine Multitouchbearbeitung vorbereitet sein.

[10] http://www.un.org/disabilities/ (letzter Abruf 10.11.2010)

4 Evaluierung

4.1 Windows Forms

Hersteller:	Microsoft Corporation
Aktuelle Version:	.NET Framework 4 (12. April 2010)
Plattform	Windows
Link:	www.windowsclient.net/

4.1.1 Allgemeines

Mit Windows Forms bezeichnet Microsoft seit 2002 mit Veröffentlichung des .Net Frameworks 1.0 eine API zur Erstellung von graphischen Benutzeroberflächen. Windows Forms ersetzt die Microsoft Foundation Class Library, die in C++ geschrieben, einen objektorientierten Aufsatz auf die Windows API darstellt.

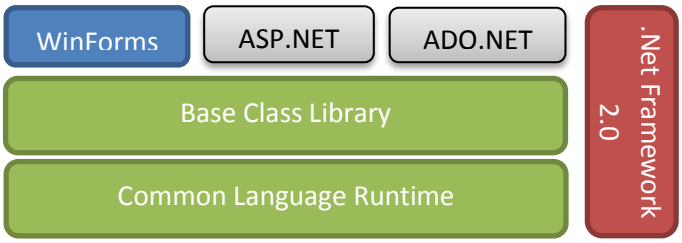


Abbildung 11: Komponenten des .Net Framework 2.0

Die Windows Forms API ist eine in managed code geschriebene auf der .Net Plattform lauffähige Programmbibliothek. Sie wird überwiegend mit den Programmiersprachen C# und Visual Basic verwendet, ist aber prinzipiell mit jeder auf der .Net Plattform verfügbaren Sprache verwendbar.

Windows Forms Anwendungen sind auch ohne Neukompilierung unter unixoiden Betriebssystemen lauffähig, sofern das Mono-Projekt installiert ist. Das Mono-Projekt ist eine Portierung der .Net Laufzeitumgebung und unterstützt die Windows Forms API bis .Net 2.0 [11].

[11] http://www.mono-project.com/WinForms (letzter Abruf 10.11.2010)

4.1.2 Bewertung nach Kriterienkatalog

Editorunterstützung in der Entwicklungsumgebung

Für die Entwicklung von Windows Forms Anwendungen kommt die Visual Studio IDE zum Einsatz. Wird der Projekttyp Windows-Anwendung gewählt, öffnet sich ein visueller Designer auf dessen Formoberfläche Steuerelemente per Drag & Drop positioniert werden können. Der Programmcode zur Positionierung wird automatisch generiert und kann vom Entwickler nicht verändert werden. Zu jedem Formfenster gibt es eine korrespondierende Code-Behind Datei, die als partielle Klasse die Definition von Eventhandlern enthält.

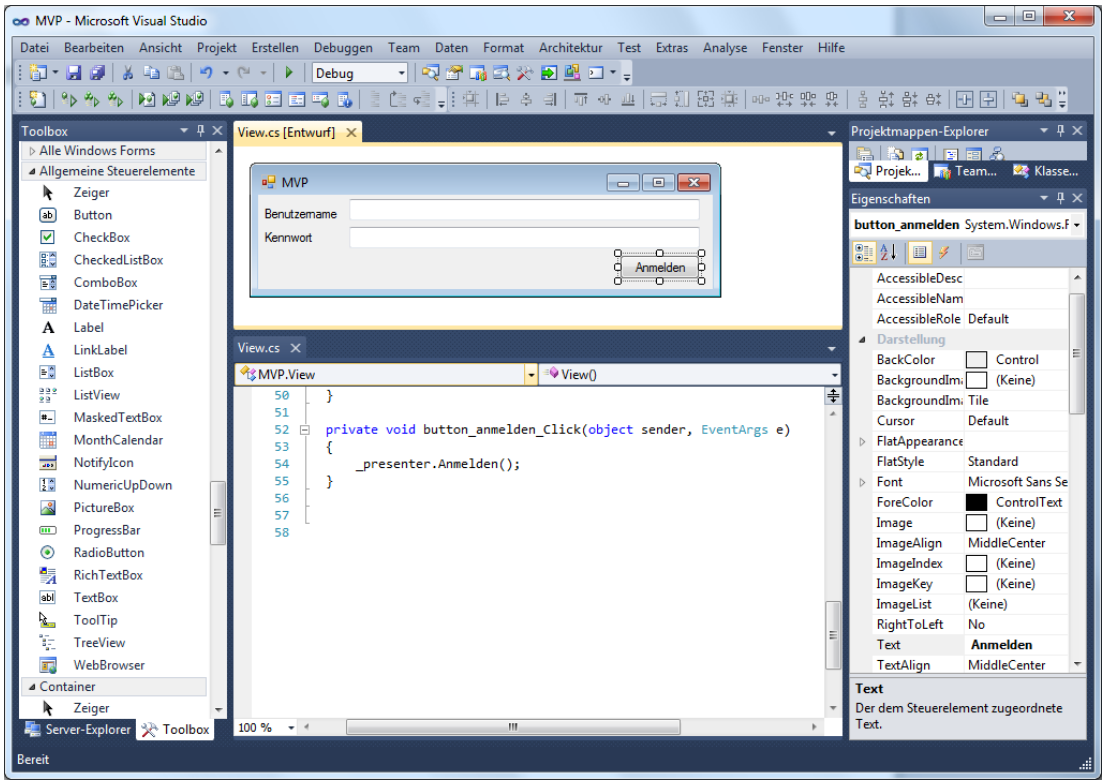


Abbildung 12: Windows FormsDesigner von Visual Studio 2010

Die Eigenschaften der Steuerelemente können in einem Eigenschaftenfenster eingesehen und bearbeitet werden.

Visual Studio bietet Drittanbietern die Möglichkeit mit Plug-Ins den Funktionsumfang der Entwicklungsumgebung zu erweitern. Die Integration gelingt dabei so nahtlos, dass kein Unterschied festzustellen ist, ob es sich um eine Erweiterung oder um Kernfunktionalität der IDE handelt.

Die Bedienung ist sehr ausgereift und mit Hilfe zahlreicher Assistenten, wird die Entwicklung beschleunigt.

Microsoft bietet Visual Studio als kostenlose Express Edition, sowie in den kostenpflichten Versionen Professional, Premium oder Ultimate an. Die Preispanne der kostenpflichtigen Editionen reicht von 650 EUR bis 12.000 EUR.

Trennung von Darstellung und Logik
Wie bereits im vorhergehenden Abschnitt erwähnt, wird die Ereignisbehandlung in der code behind Datei abgewickelt. Theoretisch ist es möglich die gesamte Logik direkt im Eventhandler zu implementieren. Dieses Vorgehen eignet sich aber nur für kleinere Projekte in denen die Trennung von Darstellung und Logik auf Grund der überschaubaren Größe nicht so eine große Bedeutung spielt. In größeren Projekten sollte ein Entwurfsmuster wie MVP (siehe Kapitel 2.6.2) verwendet werden, um eine Trennung von Darstellung und Logik zu realisieren. Domänenlogik in der code-behind Datei verstößt zum einen gegen das DRY Prinzip und ist zum anderen ohne View nicht zu testen. Durch umfangreiche Dokumentation in der MSDN Library [13] gelingt die Anwendung des MVP Musters in Windows Forms Anwendung recht mühelos.
Für eine Beispielimplementierung des MVP Musters in einer Windows Forms Anwendung, sei auf das Projekt „MVP“ auf der DVD verwiesen.

Arbeitsteilung von Designern und Entwicklern
Die Optik von Windows Forms Steuerelementen kann nur programmatisch beeinflusst werden und bietet auch dann nur begrenzte Möglichkeiten der Gestaltung wie zum Beispiel das Setzen von Schriftgröße/Farbe/Art. Um mehr Gestaltungsmöglichkeiten zu erhalten, muss anstelle der Standardsteuerelemente, die im Windows Forms Lieferumfang erhalten sind, auf die Angebote von Drittanbietern zurückgegriffen werden. In den externen Bibliotheken ist es oftmals möglich, zur Laufzeit das Look & Feel auszutauschen.
Von Haus aus bietet Windows Forms, den Designern keine Möglichkeit mit spezieller Software das optische Erscheinungsbild der Anwendung zu gestalten.

Modularität
Für die Entwicklung modularer Windows Forms Anwendungen wird der „Composite UI Application Block“ von Microsoft patterns & practices angeboten. In der Bibliothek sind bereits etliche Entwurfsmuster umgesetzt, auf die der Entwickler zurückgreifen kann. Dazu zählen Dependency Injection, um einzelne Module mit ihren Abhängigkeiten zu laden, MVC und MVP Muster zur Trennung von Darstellung und Logik, sowie das Kommando Muster um, Anwendungsbefehle wiederverwendbar zu kapseln. Die Module werden in einem Shellcontainer gehostet, der den Lebenszyklus der Module verwaltet.

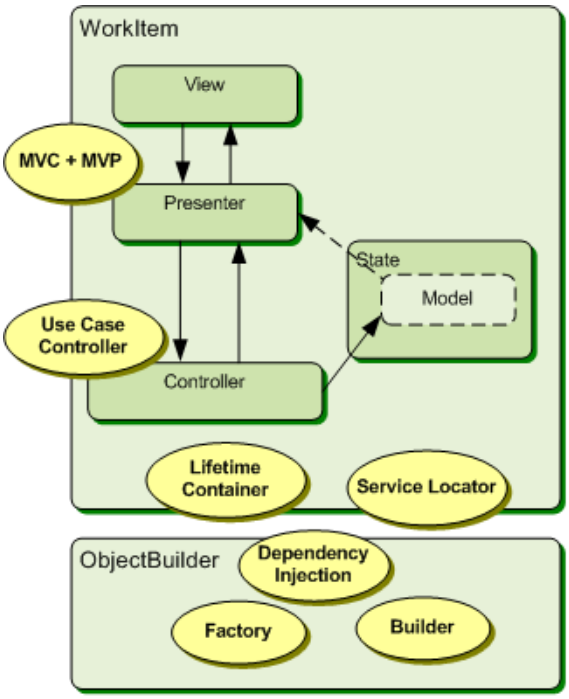


Abbildung 13: Lose Kopplung durch den Composite UI Application Block, Quelle: http://msdn.microsoft.com/en-us/library/ff709809.aspx (letzter Abruf 15.10.2010)

Testbarkeit
Um automatisierten Test-Frameworks programmatischen Zugriff zu gewähren, bietet .Net im System.Windows.Automation-Namespace (UI Automation (UIA)) eine API zur Kontrolle und Steuerung von Benutzeroberflächen. Programmatischer Zugriff bedeutet, dass durch Code jede Interaktion mit traditionellen Benutzerschnittstellen imitiert werden kann.
UI Automation bietet folgende Funktionalität:

- Ein UIA Baum erleichtert die Navigation durch die logische Struktur der Anwendung
- UI Automation Elements sind individuelle Komponenten der Oberfläche.
- UI Automation Properties stellen spezifische Information über UI Elemente zur Verfügung.
- UI Automation Events können mit Triggern auf Veränderung und Benachrichtigung von/durch Interaktion reagieren.

Auf der DVD befindet sich eine Testanwendung die den im Kapitel 2.6.2 entwickelten Login Screen testet.

Generische Steuerelemente
Windows Forms besitzt standardmäßig eine Vielzahl von Steuerelementen, die für die meisten Anwendungsfälle ausreichend sind.
Sollte der Umfang nicht ausreichen, können auf Basis der Klasse `UserControl` mit wenig Aufwand, benutzerdefinierte Steuerelemente erstellt werden, die den speziellen Anforderungen gerecht werden.
Die Anordnung der Steuerelemente kann vollständig über das Eigenschaftenfenster mit den Optionen Dock und Anchor konfiguriert werden. Intern setzt ein Layoutmanager die Einstellung entsprechend um, damit das Fenster in den korrekten Proportionen skaliert und die Komponenten an ihrem richtigen Platz bleiben.

Look & Feel
Das Erscheinungsbild von Windows Forms Steuerelementen lässt sich nur geringfügig verändern. Zu den veränderlichen Eigenschaften zählen Schriftart und Größe, sowie die Farbe von Hinter- sowie Vordergrund. Manche Steuerelemente erlauben es außerdem ein Hintergrundbild anzugeben.
Sollten umfangreichere Änderungsmöglichkeiten gewünscht sein, müssen benutzerdefinierte Steuerelemente geschrieben werden, die über die gewünschte Funktionalität verfügen. Dies macht es sehr umständlich das Look & Feel anwendungsübergreifend auszutauschen. Abhilfe schaffen hier nur die Komponenten von Drittanbietern.

Ereignisse und Befehle
Windows Forms Steuerelemente erzeugen bei Benutzerinteraktion Ereignisse, auf die in der code-behind Datei eine Reaktion erfolgen kann. Eine Ereignisabstraktion in Form von Befehlen, wie mit dem Kommandomuster ist nicht gegeben. Dieses Defizit wird nur durch die Verwendung der „Composite UI Application Block“ Bibliothek wieder wettgemacht.

Datenbindung und Validierung
Steuerelemente lassen sich über Visual Studio mit Hilfe eines Assistenten an verschiedene Datenquellen binden. Zu den unterstützen Quellen zählen Datenbanken, Dienste, gewöhnliche Objekte sowie Sharepointdienste. Durch die gute Integration in Visual Studio ist kein manuelles Programmieren erforderlich.
Für Datenvalidierung bieten alle Steuerelemente, die Benutzereingaben erlauben, eine ereignisgesteuerte Validierung an. Wenn ein solches Validierungsereignis ausgelöst wird, kann eine Methode die Eingaben überprüfen und nur bei korrekten Daten eine Weiterverarbeitung zulassen.

Unterstützung moderner Bedienkonzepte
Windows Forms Anwendungen unterstützen Drag & Drop Operationen, indem eine Ereignisbehandlung auf die Ereignisse `DragEnter`, `DragLeave`, and `DragDrop` erfolgt.
Multitouchbedienung ist unter Windows 7 auch für Windows Forms Anwendung möglich, da die Gesten durch das Betriebssystem in gewöhnliche Ereignisse konvertiert werden.

Barrierefreiheit

Der `System.Windows.Automation` Namespace, welcher bereits im Abschnitt über das programmatische Steuern der Benutzerschnittstelle zu Testzwecken verwendet wurde dient ebenfalls als Grundlage, um Menschen mit Einschränkungen durch Werkzeuge wie Screenreader[12], die Interaktion mit der Oberfläche zu ermöglichen. Microsoft hat zu diesem Thema ein umfassendes Whitepaper [18] veröffentlicht.

Internationalisierung
Visual Studio unterstützt die Lokalisierung der Benutzerschnittstelle indem die betroffenen Zeichenketten aus dem Form extrahiert und in separaten Ressourcendateien abgespeichert werden. Beim Start der Anwendung wird entsprechend der Ländereinstellung des Betriebssystems die korrekte `Resource` in das Form eingebunden.
Das Übersetzen der Oberfläche geht somit mühelos von der Hand, da alle Einstellungen über Menüs von Visual Studio vorgenommen werden können.

[12] Ein Screenreader ist eine Software, die Blinden und Sehbehinderten eine alternative Benutzerschnittstelle anstelle einer grafischen Benutzeroberfläche bietet. Es werden die Informationen, die gewöhnlich auf dem Bildschirm ausgegeben werden, mithilfe nicht-visueller Ausgabegeräte vermittelt.

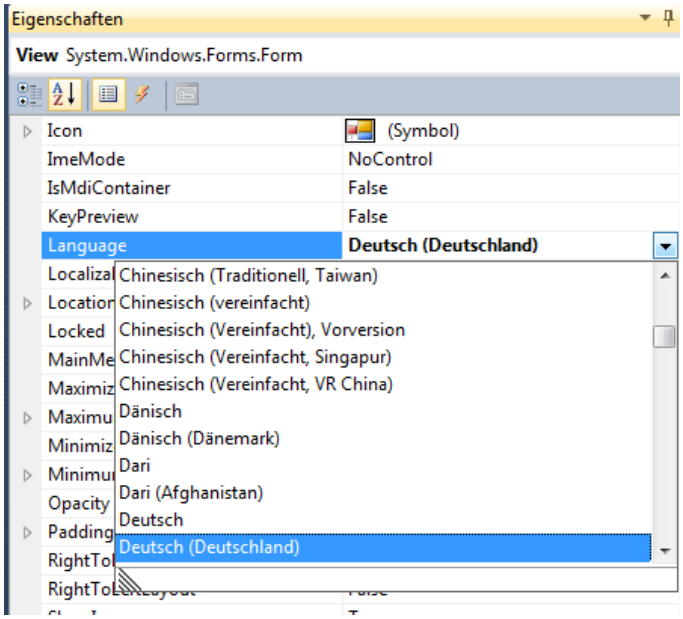


Abbildung 14: Ändern der Sprache zur Lokalisierung des Forms

Die Spracheinstellungen können auch programmatisch für die Anwendung gesetzt werden.

C# Code

```
Thread.CurrentThread.CurrentUICulture = new CultureInfo("fr-FR");
```

Listing 5: Spracheinstellungen programmatisch auf französisch stellen

Verteilung

Wenn die Anwendung einen auslieferungswürdigen Zustand erreicht hat, bietet Visual Studio zwei Optionen zur Verteilung an.

ClickOnce

ClickOnce ist eine ab .Net 2.0 zur Verfügung stehende Technologie, um Windowsanwendungen mit einer stark vereinfachten Installationsroutine installieren zu können. Aus Visual Studio heraus kann über das Menü *Erstellen -> „Anwendung veröffentlichen“* ein ClickOnce Installer erzeugt werden.

Über ClickOnce verbreitete Programme werden als „wenig beeinflussend“ (low impact) eingestuft, weil sie nur für den einzelnen Nutzer und nicht für den gesamten Rechner installiert werden. Zum Installieren werden keine Administratorrechte benötigt. Jedes ClickOnce-Programm ist vom anderen separiert. Das bedeutet, dass ClickOnce-Anwendungen untereinander nicht in Konflikte geraten können. ClickOnce-Programme sind selbstaktualisierend und können automatisch beim Start auf die Verfügbarkeit einer neueren Version prüfen und alle aktualisierten Dateien ersetzen.

Für einfache Installationsszenarien bietet ClickOnce bei nur wenig Aufwand einen erheblichen Nutzen.

Windows Installer

Windows Installer ist eine Laufzeitumgebung für Setuproutinen zur Installation, Wartung, und Entfernung von Software, die fest in das Windows Betriebssystem integriert ist. Während es das Ziel von ClickOnce war, eine möglichst einfache Installation zu ermöglichen, bietet Windows Installer viel mehr Möglichkeiten das Setup zu konfigurieren. Dies umfasst das Setzen von Dateitypverknüpfung, das Reparieren von Installationen, die Generierung einer Deinstallationsroutine, die Auflösung von Abhängigkeiten zu anderen Softwareprodukten (Datenbanken, Browser, etc.), sowie die Aufteilung eines großen Programms in mehrere Teile (Features), die vom Anwender bei Bedarf nachinstalliert werden können. Für weitere Informationen sei auf [19] verwiesen.

Angebote von Drittanbietern

Es gibt etliche Hersteller, die den Funktionsumfang von Windows Forms mit eigenen Steuerelementen erweitern. Im Folgenden werden drei Anbieter mit einem Auszug aus ihren Angeboten genannt.

Infragistics[13]: Office Ribbon Looks, Kalender, Diagramme, Rechtschreibprüfung, Layout Manager; 995 USD

Telerik[14]: Outlook Status Bar, Farbauswahldialog, Kontextmenüs; 799 USD

DevExpress[15]: Pivot, Reports, Editoren; 799 USD

Die Komponenten lassen sich alle in Visual Studio integrieren und wie herkömmliche Standardsteuerelemente konfigurieren.

Hilfe und Support

Die deutschsprachige Referenzdokumentation findet sich in der MSDN Library[16] und ist die erste Anlaufstelle, um Fragen zu Windows Forms zu beantworten. Neben einer vollständigen Beschreibung der API wird in zahlreichen Schritt-für-Schritt Anleitungen auch die Bedienung der vielen Assistenten von Visual Studio erläutert.
Im Forum des Windows Developer Center[17] können Fragen zur Windows Forms Technologie und Verteilungsstrategie gestellt werden. Innerhalb von wenigen Stunden erhält man dort eine Antwort. Oftmals sogar gleich eine Lösung, da in dem Forum etliche MVPs[18] vertreten sind.
Seit Veröffentlichung von Windows Forms im Jahr 2002 sind viele umfassende Bücher erschienen. Insbesondere die Bücher von Matthew MacDonald [20] vermitteln ein

---

[13] http://www.infragistics.com/dotnet/netadvantage/wpf.aspx (letzter Abruf 10.11.2010)

[14] http://www.telerik.com/products/wpf.aspx (letzter Abruf 10.11.2010)

[15] http://www.devexpress.com/Products/NET/Controls/WPF/ (letzter Abruf 10.11.2010)

[16] http://msdn.microsoft.com/en-us/library/dd30h2yb.aspx (letzter Abruf 10.11.2010)

[17] http://social.msdn.microsoft.com/forums/en-US/category/windowsforms/ (letzter Abruf 10.11.2010)

[18] Als Microsoft® Most Valuable Professional (MVP) werden die besten und versiertesten Mitglieder der Technologiecommunitys auf der ganzen Welt durch Microsoft ausgezeichnet.

tiefgreifendes Verständnis und geben viele Praxistipps, die sich unmittelbar in die Anwendung integrieren lassen. Die Einarbeitung in Windows Forms gelingt durch die vielen Bücher und die MSDN Dokumentation sehr schnell.
Die vermutlich größte Hilfe stellt allerdings die Bibliothek „Composite Application Block“ dar. Zum einen erspart sie dem Entwickler etliche Arbeit an der Infrastruktur der Anwendung, zum anderen sind zwei dokumentierte Referenzimplementierungen enthalten, die aufzeigen wie man den größtmöglichen Nutzen aus der Bibliothek ziehen kann.

4.2 Windows Presentation Foundation

Hersteller:	Microsoft Corporation
Aktuelle Version:	.NET Framework 4 (12. April 2010)
Plattform	Windows
Link:	www.windowsclient.net/

4.2.1 Allgemeines

Microsoft stellte im Jahr 2006 mit der Veröffentlichung des .NET Framework 3.0 erstmalig die Windows Presentation Foundation vor. Mit WPF hat Microsoft einen Paradigmenwechsel bezüglich der Entwicklung von Benutzerschnittstellen vollzogen. Während Windows Forms lediglich einen Wrapper um die Windows-API[19] gelegt hat und somit an teilweise zehn Jahre alte Designentscheidungen gebunden ist, erfolgte mit WPF eine Neuentwicklung von Grund auf.

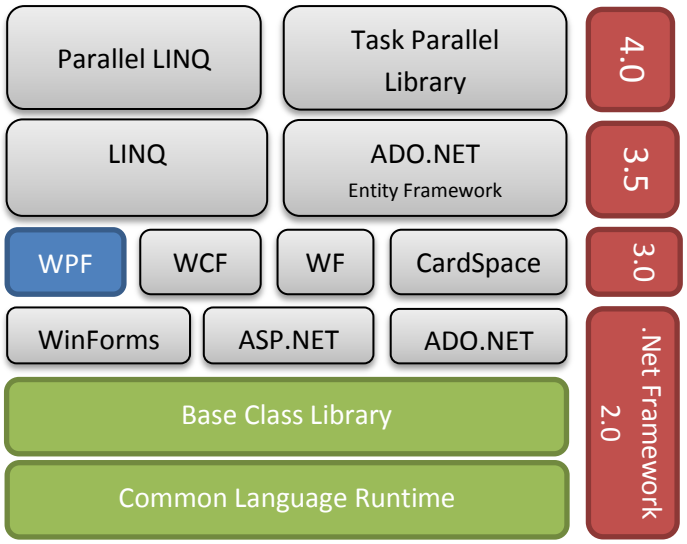


Abbildung 15: Komponenten der .Net Plattform 4.0

[19] Bei den APIs handelt es sich um User32 und GDI/GDI+.

Seit 2006 hat die Windows Presentation Foundation mehrere Aktualisierungen erhalten und ist derzeit in Version 4 erhältlich.

- WPF 3.0. ist die erste Version von WPF und wurde erstmalig zusammen mit Windows Workflow Foundation und Windows Communication Foundation im .Net Framework 3.0 veröffentlicht
- WPF 3.5. Ein Jahr später, wurde mit dem Framework 3.5 eine neue Version mit geringen Veränderungen, Optimierungen und Bugfixes veröffentlicht.
- WPF 3.5 SP1. Mit dem .NET Framework Service Pack 1 (SP1) wurde WPF um weitere graphische Effekte sowie ein fortschrittliches DataGrid erweitert
- WPF 4. Die aktuelle Version optimiert das Textrendering, ermöglicht natürlichere Animationen und unterstützt Funktionen von Windows 7, wie beispielsweise Multitouch und die neue Taskbar.

Deklarative Beschreibung der Benutzeroberfläche

In WPF wird die Benutzeroberfläche durch eine deklarative XML-basierte Auszeichnungssprache namens XAML (Abkürzung für: Extensible Application Markup Language, gesprochen: zammel) definiert.

XAML ist eine relativ einfache, universell einsetzbare Sprache, um .Net Objekte zu konstruieren und zu initialisieren. Das .Net Framework 3.0 enthält einen Compiler, einen Laufzeitparser, sowie ein Browserplugin um WPF basierte Seiten im Browser anzuzeigen.

Weil XAML lediglich eine Möglichkeit darstellt, die .Net API anzusteuern, sind Vergleiche mit HTML, Scalable Vector Graphics (SVG) oder domänenspezifischen Sprachen oftmals irreführend. XAML besteht aus Regeln, die dem Parser/Compiler mitteilen, wie der XML-Code samt Schlüsselwörtern zu verwenden ist.

Auch wenn XAML ursprünglich für WPF entworfen wurde, findet es auch in anderen Technologien, wie der Windows Workflow Foundation, Verwendung.

Die XAML Spezifikation definiert Regeln, um .NET Namensräume, Typen, Eigenschaften und Events auf XML Namensräume, Elemente und Attribute abzubilden. Voraussetzung ist, dass die Klassen über einen Standardkonstruktor[20] verfügen. An folgendem einfachen aber vollständigen Beispiel lässt sich der Zusammenhang zwischen deklarativen XAML Code und imperativen C# Code erläutern. Es wird im ersten Listing ein Button mit einigen Eigenschaften und einem Eventhandler in XAML definiert und im zweiten Listing der korrespondierende äquivalente C# Programmcode gezeigt.

XAML Code

```
<Button xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
 MinWidth=”120” Margin=”5” Content=”Login” Click=”Button_Click” />
```

Listing 6: Einen Button durch XAML Code erzeugen

[20] Ein Standardkonstruktor erwartet keine Argumente.

C# Code

```
System.Windows.Controls.Button b = new System.Windows.Controls.Button();
b.MinWidth = 120;
b.Margin = 5;
b.Content = “Login”;
b.Click += new System.Windows.RoutedEventHandler(Button_Click);
```

Listing 7: Einen Button durch C# Code erzeugen

Der XML Namensraum[21] *http://schemas.microsoft.com/winfx/2006/xaml/presentation* stellt die Abbildung auf den .Net Namensraum `System.Windows.Controls` dar. Mit dem <Button…> Tag wird eine Instanz der korrespondierenden Button Klasse aus dem .Net Framework mit ihrem Standardkonstruktor erzeugt. `MinWidth`, `Margin` und `Content` setzen jeweils Eigenschaften des frisch erzeugten Objekts. Mit dem Click Attribut wird das Ereignis, welches durch einen Mausklick auf den Button ausgelöst wird, mit einer Methode (`ButtonClick`) verknüpft. Bei Click handelt es sich um ein C# Event, dem eine Methode mit passender Signatur hinzugefügt wurde.

C# Code

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    // Tu was!
}
```

Listing 8: Methode zur Ereignisbehandlung

Der Methode zur Ereignisbehandlung wird die Ereignisquelle (`sender`), in unserem Fall die Buttoninstanz und ein Funktionsobjekt (`e`), welches die Interna der Ereignisbehandlung regelt, übergeben.

Dieses kurze Beispiel über die Verwendung von XAML soll verdeutlichen, dass man durchaus auch WPF Anwendungen ohne die Verwendung von XAML schreiben kann, da der XAML-Parser nichts weiter macht, als auf Basis von XML-Elementen Exemplare von .Net Klassen zu initialisieren und zu konfigurieren. Der große Vorteil in der Nutzung von XAML wird sich noch in einigen Punkten bei der Evaluierung zeigen.

Hardwarebeschleunigung

WPF verwendet für die Darstellung Direct3D[22]. Die Inhalte (2D, 3D, Graphiken, Text) einer Benutzeroberfläche werden in Direct3D Objekte (Dreiecke, Texturen) konvertiert, damit sie von der Grafikkarte verarbeitet werden können. Dadurch kann WPF im Gegensatz zu GDI-basierten Anwendungen die Hardwarebeschleunigung der Grafikkarte (GPU) voll ausnutzen.

[21] Namensräume stellen Eindeutigkeit von Namen her.

[22] Direct3D ist eine Programmierschnittstelle (API) von Microsoft für 3D-Computergrafik. Direct3D ist ein Bestandteil von DirectX.

Intelligente dynamische Layouts

Bei der Gestaltung von Benutzeroberflächen verfolgt WPF den Ansatz fließender und dynamisch skalierender Layouts, wie sie aus dem Web mit HTML bekannt sind. Steuerelemente werden nicht mit festen Koordinaten und Größen auf der Oberfläche fixiert, sondern mit Hilfe von Layoutmanagern positioniert, die eine eigene Logik zur Anordnung ihres Inhalts haben. Dadurch lassen sich Anwendungen entwerfen, die sich dynamisch an die Fenstergröße anpassen und auch auf großen Monitoren den zur Verfügung stehenden Raum optimal nutzen.

4.2.2 Bewertung nach Kriterienkatalog

Editorunterstützung in der Entwicklungsumgebung

Zur Entwicklung von WPF Anwendung stellt Microsoft die Visual Studio IDE zur Verfügung. Für die Untersuchung in dieser Arbeit wird die Version Visual Studio 2010 verwendet.

Für Entwickler, die zuvor schon mit Visual Studio gearbeitet haben, birgt der WPF-spezifische Teil der IDE keine großen Neuigkeiten. Es gibt den speziellen Projekttyp „WPF-Anwendung“, der Entwickler in die Lage versetzt, WPF mit einem WYSIWYG ähnlichen Editor (Codename: „Cider“) zu entwerfen. Dabei werden in einem geteilten (vertikalen oder horizontalen) Fenster die Anwendungsoberfläche und der dazugehörige XAML-Code angezeigt.

Aus einer Toolbox können theoretisch Steuerelemente per Drag & Drop auf der Oberfläche angeordnet werden. Der generierte XAML Code wird dadurch jedoch schnell aufgebläht, da etliche Eigenschaften automatisch gesetzt werden, ohne dass sie wirklich benötigt werden. Der Editor generiert des Weiteren durch das Setzen fester Außenabstände an den Steuerelementen ein vollkommen starres Layout, anstatt die vielen Layoutmanager zu nutzen, wie es eigentlich der zuvor erläuterten Philosophie von WPF entspräche.

In der Praxis[23] sieht es eher so aus, dass der Entwickler den XAML-Code überwiegend von Hand schreibt und lediglich einzelne Eigenschaften über den Eigenschaftendialog setzt. Im XAML-Editor steht der volle IntelliSense Funktionsumfang zur Verfügung, sodass nach einer gewissen Einarbeitungszeit das Schreiben des deklarativen Codes relativ schnell von der Hand geht.

Nichtsdestotrotz lässt der graphische Designer von Visual Studio sehr zu wünschen über, da die effektive Anordnung der Komponenten nur direkt im XAML Editor erfolgen kann.

---

[23] In der WPF Literatur [30] ist die einstimmige Meinung, XAML-Code in Visual Studio von Hand zu schreiben oder die speziell auf visuelle Gestaltung ausgerichtete Expression Suite zu verwenden.

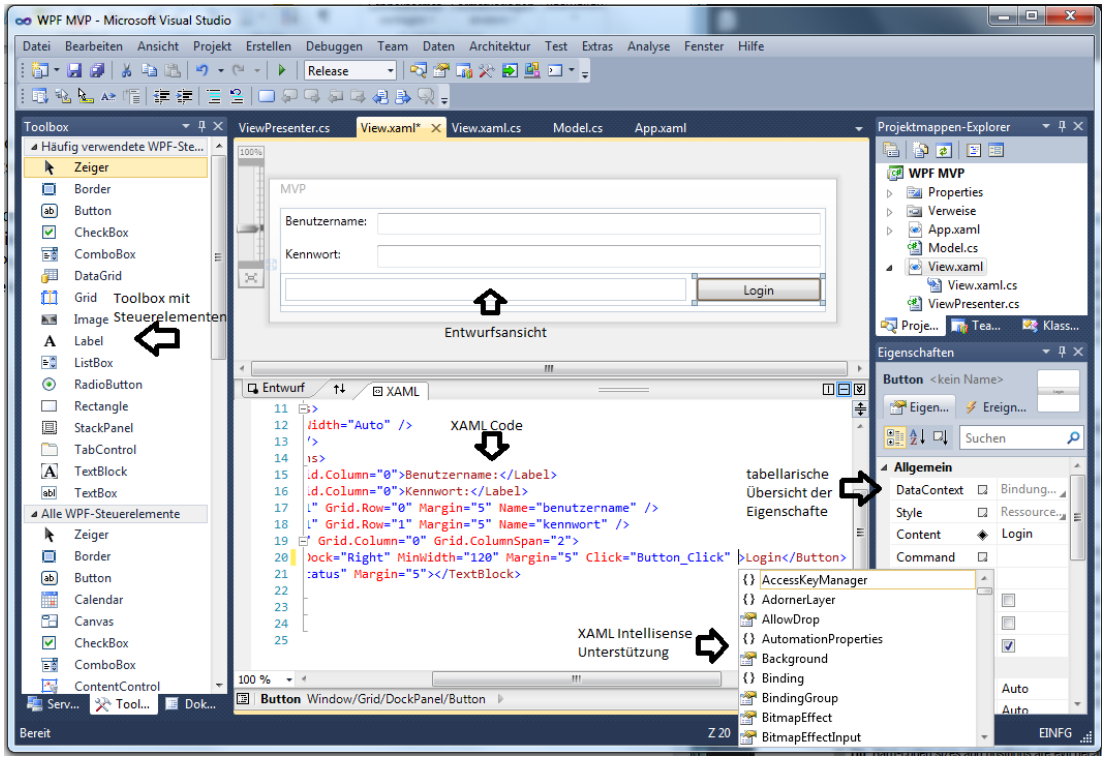


Abbildung 16: WPF Designer in Visual Studio 2010

Für die Integration von Drittanbieterplugins sowie Preise für Visual Studio sei auf das Windows Forms Kapitel 4.1.2 verwiesen.

Trennung von Darstellung und Logik

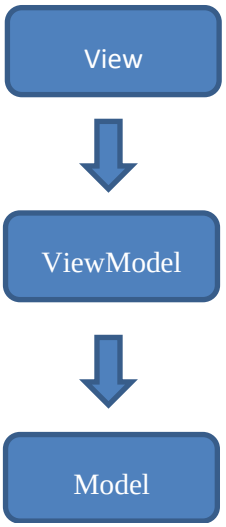


Abbildung 17: MVVM Muster

Bei der Entwicklung von WPF ist die Trennung von Darstellung und Logik ein Hauptanliegen von Microsoft gewesen. Dies zeigt sich daran, dass noch vor Veröffentlichung von WPF der Architekt John Gossmann im Jahr 2005 in seinem Blog [21] mit MVVM (Model-View-ViewModel) ein Entwurfsmuster vorstellte, dass unter Ausnutzung der von WPF angeboten Funktionen eine saubere Trennung der Verantwortlichkeiten anpeilt.
Das MVVM-Muster geht von einer Dreiteilung der Anwendung in ein Model (M), ein ViewModel (VM) und einen View (V), aus. Das Model ist das Herz der Anwendung und macht den wichtigsten Teil der Anwendung aus, da es die komplexen Geschäftsentitäten, ihre Verbindungen und Funktionalität enthält.
Oberhalb des Models befindet sich das ViewModel, welches zwei Ziele verfolgt. Zum einen soll sich das Model durch das ViewModel leichter visualisieren lassen. Dazu bietet das VM unter Ausnutzung von WPF-Sprachfeatures wie Databinding und Templates Funktionalität an, damit der View das Model leichter „konsumieren“ kann. Zum anderen wird der View vom Model abgekapselt.

Der View enthält ein ViewModel in seinem Datenkontext, um auf die Dienste des ViewModels zugreifen zu können.
Wie aus den Pfeilen des Diagramms ersichtlich wird, haben sowohl View als auch ViewModel keinerlei Kenntnis von der View. Diese Trennung wird im weiteren Verlauf der Evaluierung noch zum Tragen kommen, wenn die Punkte Testbarkeit und Modularität bewertet werden. Das MVVM Muster weist etliche Parallelen zu den in Kapitel 2.6 dargestellten Entwurfsmustern auf und hat auch nicht den Anspruch ein radikaler Neuentwurf zu sein.
Es wurde mit WPF vielmehr eine Technologie geschaffen, die viele Möglichkeiten zur losen Kopplung von Komponenten bietet. Das MVVM Muster schöpft diese Möglichkeiten konsequent aus. Daher ist es auch nicht auf andere Technologien zu adaptieren, da technologieabhängig auf viele WPF-spezifische Eigenschaften aufgebaut wird. Auf einige der Eigenschaften wie Datenbindung und Befehle wird noch weiter eingegangen.
In der Projektmappe „MVVM“ auf der DVD ist der aus Kapitel 2.6.2 bekannte Logindialog mit dem MVVM Muster realisiert.

Arbeitsteilung von Designer und Entwicklern
Wir bereits im vorangehenden Absatz erwähnt, versteht sich WPF auf die Trennung von Darstellung und Logik. Um diesen Arbeitsablauf weiter zu optimieren, hat Microsoft mit der Expression Suite ein Werkzeug geschaffen, mit dem sich Projektmappen aus Visual Studio öffnen lassen, damit der darin enthaltene XAML Code von Designern professionell bearbeitet werden kann. Expression geht dabei weit über die dürftigen gestalterischen Fähigkeiten von Visual Studio hinaus und richtet sich gezielt an Designer, die der Anwendung optischen Glanz verschaffen.

Modularität
Um eine hohe Modularität innerhalb der Anwendung zu erreichen, wird von Microsoft zur Verwendung des Prism[24] Rahmenwerkes geraten, das aus der Forschung des Patterns & Practices Team hervorgegangen ist.
Ein Modul innerhalb einer Prism Anwendung muss über eine Klasse verfügen, die das IModule Interface implementiert.

C# Code

```
public interface IModule
{
    void Initialize();
}
```

Listing 9: IModule Interface aus dem Prism Framework

Die Methode `Initialize()` wird im Initialisierungsvorgang des Moduls aufgerufen, um das Modul in einen gültigen Ausgangszustand zu versetzen. Da Module oftmals Abhängigkeiten mit anderen Komponenten des Systems haben, verwendet das Prism Rahmenwerk Unity als leichtgewichtigen Dependency Injection Container, um die Konfiguration der Module über Konstruktorinjektion vorzunehmen.

[24] http://compositewpf.codeplex.com/ (letzter Abruf 10.11.2010)

C# Code

```
class ZeiterfassungsModul : IModule
{
    public ZeiterfassungsModul(IUnityContainer container, IRegionManager
                                    regionManager)
    {
        _container = container;
        _regionManager = regionManager;
    }
    public void Initialize() {}
}
```

Listing 10: Über Konstruktorinjektion konfigurierbares Modul (gekürzt)

Im obigen Listing wird ein `ZeiterfassungsModul` konfiguriert, indem der DI Container sowie ein `RegionManager` übergeben werden.
Eine sogenannte Shell stellt den Rahmen für die Oberfläche dar. In ihr werden Regionen definiert, in denen die einzelnen Module Platz finden. Regionen müssen das `IRegion` Interface implementieren, um das Anzeigen, Aktivieren und Entfernen von Regionen aus der Shell zu ermöglichen.

C# Code

```
public interface IRegion
{
    IViewsCollection Views { get; }
    IViewsCollection ActiveViews { get; }
    IRegionManager Add(object view);
    IRegionManager Add(object view, string viewName);
    IRegionManager Add(object view, string viewName, bool
                        createRegionManagerScope);
    void Remove(object view);
    void Activate(object view);
    void Deactivate(object view);
    object GetView(string viewName);
    IRegionManager RegionManager { get; set; }
}
```

Listing 11: Das IRegion Interface

Über einen `RegionManager` ist die Anordnung und Verankerung der einzelnen Regionen in der Shell möglich.

C# Code

```
IRegion region = _regionManager.Regions["MainRegion"];

var ordersPresentationModel = _container.Resolve<IOrdersPresentationModel>();
var _ordersView = ordersPresentationModel.View;
region.Add(_ordersView, "OrdersView");
region.Activate(_ordersView);
InitializeComponent();
```

Listing 12: Einer region einen View hinzufügen

Um einer Region einen View hinzuzufügen, muss die Region vom `RegionManager` angefordert werden und über die Add-Methode einen View erhalten.

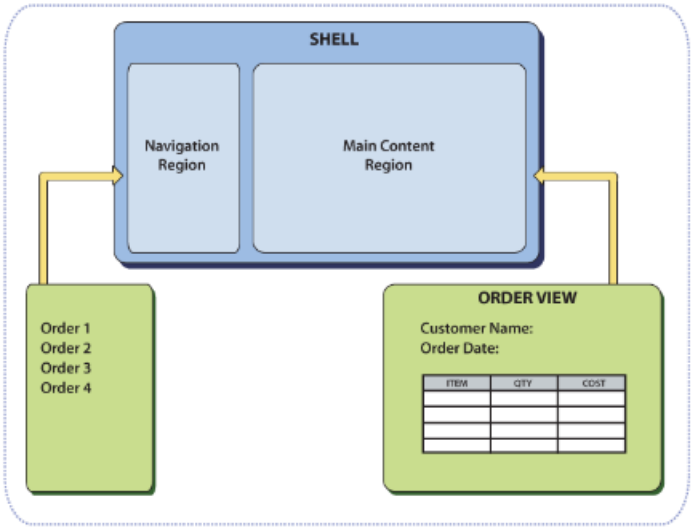


Abbildung 18: Regionen und Views in einer Prism Shell

Auf diese Weise können Module mit den dazugehörigen Views getrennt voneinander entwickelt werden.

Testbarkeit

Um automatisierten Test-Frameworks programmatischen Zugriff zu gewähren, bietet .Net im `System.Windows.Automation`-Namespace (UI Automation (UIA)) eine API zur Kontrolle und Steuerung von Benutzeroberflächen. Siehe die Erläuterungen im Windows Forms Kapitel 4.1.

Technische Funktionen

Generische Steuerelemente

WPF liefert von Haus eine Vielzahl von Steuerelementen mit, die in graphischen Benutzerschnittstellen Verwendung finden. Neben Standardelementen wie Textfeldern, Comboboxen und Buttons zählen auch neuartige Controls, wie die aus Office bekannte Ribbonoberfläche, zum Funktionsumfang. Da WPF eine verhältnismäßig neue Technologie ist, kann man erwarten dass in zukünftigen Versionen der Funktionsumfang weiter zunehmen wird. Mit .Net 4 wurde unter anderem ein leistungsfähigeres DataGrid (Steuerelement zur Anzeige tabellarischer Daten), sowie ein Datumsauswahldialog hinzugefügt.
Die Komponentenvielfalt reicht aber noch nicht an Windows Forms heran.

Um den Umstieg von Windows Forms auf WPF zu erleichtern, besitzt WPF einen Hosting-Modus für Windows Forms Steuerelemente.

XAML Code

```
<WindowsFormsHost>
    <wf:MaskedTextBox x:Name="mtbDate" Mask="00/00/0000" />
</WindowsFormsHost>
```

Listing 13: WPF Hosting Mode für Windows Forms Steuerelemente

Mit der Klasse `CustomControl` lassen sich Steuerelemente entwickeln, für die die WPF Bibliothek keine Lösung parat hat.

Look & Feel

Jedes WPF Steuerelement ist zunächst einmal „lookless". Das bedeutet, dass die visuelle Erscheinung (look) vollständig ausgetauscht werden kann, während die Funktionalität erhalten bleibt.
Durch Styles, die Steuerelementen zugewiesen werden können, ist es möglich, ein Element in seinem Aussehen stark zu verändern. Das Vorgehen erinnert dabei stark an Cascading Stylesheets, wie sie aus der Webentwicklung bekannt sind.
Solche Styles können als anwendungsübergreifende Ressourcen definiert werden, damit sie beliebig wiederverwendet werden können.

XAML Code

```
<Style x:Key="GroßeSchriftButtonStyle">
    <Setter Property="Control.FontFamily" Value="Wingdings" />
    <Setter Property="Control.FontSize" Value="40" />
    <Setter Property="Control.FontWeight" Value="Bold" />
</Style>
```

Listing 14: Definition eines WPF-Styles

XAML Code

```
<Button Style="{StaticResource GroßeSchriftButtonStyle}">Login</Button>
```

Listing 15: Mit Style gestalteter Button

Links ist der Button in seiner Standardoptik und rechts mit angewandtem Style.

Ereignisse und Befehle

Wpf unterstützt primitive Ereignisse (Mausklick, Tastendruck, Fokuswechsel) und auch Befehle.

Um einen Button mit der Fähigkeit auszustatten, auf Mausklicks zu reagieren, reicht es die Click-Eigenschaft im XAML Code zu setzen, damit das Ereignis mit einer Methode aus der Codebehind-Datei verknüpft wird.

XAML Code

```
<Button Click=”Button_Click”>Login</Button>
```

Listing 16: Button mit Eventhandler für das Click-Event

Die ButtonClick Methode erhält als Parameter die Ereignisquelle, sowie weitere für die Ereignisbehandlung relevante Information.

Des Weiteren erlaubt es WPF, Anwendungsaufgaben in Form von Befehlen (Commands) zu definieren und mit Steuerelementen zu verknüpfen. Dies hat den Vorteil, dass nicht wiederholt Ereignisbehandlungs-Code geschrieben werden muss, wenn ein Befehl von mehreren Steuerelementen aus erreichbar ist. Außerdem kapseln Commands die Logik über den Zustand des Befehls, indem er automatisch inaktiv wird, wenn er im Kontext der Anwendung nicht zur Verfügung steht.

Um ein Verständnis für Commands zu erhalten, hilft ein Blick in das `ICommand` interface.

C# Code

```
public interface ICommand
{
    void Execute(object parameter);
    bool CanExecute(object parameter);
    event EventHandler CanExecuteChanged;
}
```

Listing 17: ICommand Interface von WPF

In der Execute Methode wird die eigentliche Arbeit erledigt. Die `CanExecute` Methode überprüft, ob der Befehl im aktuellen Kontext zur Verfügung steht. Über das `CanExeceuteChanged` Event wird signalisiert, ob sich am Zustand des Befehls etwas verändert hat. WPF stellt über den die Klassen `RoutedCommand` und `RoutedUICommand` Implementierungen des Interface bereit, die die Entwicklung eigener Befehle erleichtern. Zur Vertiefung sei auf die „MVVM“ Projektmappe auf der CD verwiesen.

Datenbindung und Validierung

Mit WPF lassen sich Steuerelemente untereinander oder mit beliebigen Objekten verbinden.

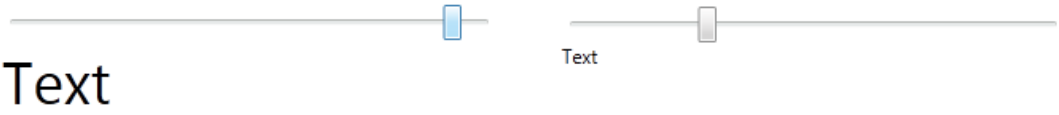
In folgendem Listing kann die Schriftgröße eines Textes mit einem Slider verändert werden. Dies gelingt vollständig in deklarativem XAML Code. Der Wert (Value) des Sliders erhält einen Bindungsausdruck auf die Schriftgröße (Path=FontSize) eines anderen Steuerelements (ElementName=textblock).

XAML Code

```
<Slider Value="{Binding ElementName=textblock, Path=FontSize}" />
<TextBlock Name="textblock" Text="Text" />
```

Listing 18: Der Wert des Sliders wird per Databinding an die Schriftgröße der Textbox gebunden.

Wird der Slider bewegt, ändert sich automatisch die Schriftgröße des Textblocks.



Text

Dieses einfache Beispiel soll zeigen, wie Steuerelemente mit Hilfe von Bindungsausdrücken verbunden werden können. Datenbindung ist in komplexeren Szenarien weitaus nützlicher. Wenn beispielsweise Daten in relationalen Datenbanken gespeichert sind, können über Visual Studio Datenquellen verwaltet werden und entsprechende Steuerelemente (Datagrid) diesen Quellen zugewiesen werden. Derart lassen sich schnell datenzentrierte Geschäftsanwendungen entwerfen, ohne dass viel Aufwand in die Synchronisierung von Model und Darstellung investiert werden muss.

In die Datenbindung von WPF ist die Datenvalidierung fest eingebaut. Wird der Bindungsausdruck um eines der Attribute „`ValidatesOnExceptions`“ oder „`ValidatesOnDataError`“ ergänzt, können in der Oberfläche Fehlerinformationen bezüglich der gemachten Eingabe abgerufen werden.

Standardmäßig werden Steuerelemente bei Fehlern mit einem roten Rand angezeigt. Dieses Verhalten lässt sich mit Styles modifizieren.

Bedienkonzepte

WPF bietet Drag & Drop Funktionalität[25], um Steuerelemente auf dem Bildschirm bewegen zu können. Dies erfordert aber noch sehr viel Programmieraufwand und ist von Steuerelement zu Steuerelement unterschiedlich.

Multitouch-Unterstützung von WPF im Zusammenspiel mit Windows7 ist gegeben. Für weitere Information sei auf [22] verwiesen.

Barrierefreiheit

Für die Barrierefreiheit von WPF Anwendungen gelten die gleichen Konzepte wie für Windows Forms Anwendungen.

Lokalisierung

Für die Lokalisierung von WPF Anwendungen werden satellite assemblies[26] verwendet. Beim Buildvorgang wird der XAML Code kompilliert und entsprechend des Länderkürzels in einer separaten Assembly untergebracht. Aus diesen Assemblies lassen sich mit dem Kommandozeilenttool `LocBAML` die zu lokalisierenden Zeichenketten als CSV Datei

[25] http://msdn.microsoft.com/en-us/library/ms749074.aspx (letzter Abruf 10.11.2010)

[26] Assemblies, die keinen Programmcode sondern lediglich Ressourcen beinhalten, werden als Satellite Assemblies bezeichnet.

extrahieren. Wenn die CSV Datei übersetzt ist, wird mit LocBAML eine satellite assembly erzeugt.

Beim Programmstart werden automatisch die korrekten Assemblies entsprechend der Windowsspracheinstellungen geladen.

Wie an dieser vereinfachten Beschreibung zu erkennen ist, ist das Lokalisieren von WPF Anwendung derzeit noch etwas umständlich.

Verteilung

Für die Verteilung von WPF Anwendungen gelten die gleichen Konzepte wie für WindowsForms Anwendungen.

Angebote von Drittanbietern

Die bereits im Windows Forms Kapitel 4.1 erwähnten Hersteller bieten alle ihre Steuerelemente auch als WPF Versionen an. Dadurch ist es dem Kunden sogar möglich, beim Umstieg von Windows Forms auf WPF, weiterhin die bekannte API des Herstellers zu nutzen und trotzdem die modernere WPF Technologie zu verwenden.

Hilfe und Support

Erste Anlaufstelle für Entwickler sind die MSDN Seiten[27] zur Windows Presentation Foundation, die direkt von Microsoft gepflegt werden und die offizielle Dokumentation darstellen. Auf diesen auch in Deutsch verfügbaren Seiten wird sehr ausführlich Architektur und Funktionsumfang der WPF erläutert.

Der Buchhandel zur WPF ist nach wie vor sehr überschaubar. Während es im englischsprachigen Raum bereits einige Titel in immer wieder aktualisierten Auflagen gibt, ist auf dem deutschen Markt nur der Autor Thomas Claudius Huber mit einem WPF Buch [23] vertreten. Alle WPF Bücher haben gemeinsam, dass sie zwar die Funktionsweise von WPF detailliert erläutern, aber nicht darüber hinaus Praxiswissen zur effektiven Anwendung in der Softwareentwicklung vermitteln. Mit dem Prism Framework versucht der Forschungsbereich Patterns & Practices von Microsoft die Lücke zu schließen, indem Entwicklern ein Leitfaden zur Strukturierung der Anwendung an die Hand gegeben wird. Der zusätzliche Einarbeitungsaufwand von Prism ist aber sehr hoch und für kleinere Projekte sicherlich nicht gerechtfertigt.

WPF wird vermutlich auch, weil es noch eine sehr junge Technologie mit wenigen Langzeiterfahrungen ist, in vielen Blogs thematisiert. Eine herausragende Rolle nimmt dabei der Blogger Josh Smith[28] ein, der auch zahlreiche MSDN Artikel über WPF und MVVM veröffentlicht. Kompetente Hilfestellung erfährt der Entwickler in der WPF Disciples Group[29].

Da WPF mit der strikt deklarativen Definition einen radikalen Neuanfang wagt, der sich vom bisherigen Entwicklungsvorgehen mit GUI-Buildern unterscheidet, ist der

---

[27] http://msdn.microsoft.com/en-us/library/ms754130.aspx (letzter Abruf 10.11.2010)

[28] http://joshsmithonwpf.wordpress.com/ (letzter Abruf 10.11.2010)

[29] http://groups.google.com/group/wpf-disciples/ (letzter Abruf 10.11.2010)

Einarbeitungsaufwand für Entwickler sehr hoch. Während es das Konzept von Windows Forms ist, per Drag & Drop möglichst schnell die benötigten Steuerelemente auf einem Form zu platzieren, erfordert WPF mehr Codierungsaufwand von Hand.

Im Internet wird auch eine emsige Debatte[30] darüber geführt, ob WPF schon bereit für Geschäftsanwendungen ist. Selbst mit der aktuellen Version 4 reicht WPF in Bezug auf Steuerelemente noch nicht an den Umfang des Vorgängers Windows Forms heran. Dieses Defizit wird jedoch von Drittanbietern wieder ausgeglichen.

[30] http://social.msdn.microsoft.com/forums/en-US/wpf/thread/af511166-002d-472c-b759-131d002adb43/ (letzter Abruf 10.11.2010)

4.3 NetBeans Plattform

Hersteller:	Sun Microsystems / Oracle Corporation
Aktuelle Version:	Version 6.9.1, August 4, 2010
Plattform	plattformunabhängig
Link:	www.netbeans.org/

4.3.1 Allgemeines

Die NetBeans Plattform ist ein generisches Rahmenwerk, um die Entwicklung von Swing-Desktopanwendungen in Java zu vereinfachen. Die NetBeans Plattform entstand aus einem Projekt tschechischer Studenten, die im Jahre 1996 eine integrierte Entwicklungsumgebung, für die damals noch junge Programmiersprache Java schaffen wollten.

Im Java Development Kit werden zwei Pakete für alle Benutzerschnittstellen-Komponenten bereitgestellt. Das Paket *java.awt* (Abstract Window Toolkit) enthält Komponenten, die auf plattformspezifische Elemente (peers) abgebildet werden. Diese Komponenten sind fehleranfällig für plattformspezifische Bugs, da der peer-basierte Ansatz stark auf dem darunterliegenden Betriebssystem aufbaut [24]. AWT eignet sich daher lediglich für die Entwicklung sehr einfacher Benutzerschnittstellen.

Seit Java2 wurden die *java.awt* Komponenten durch das robustere und vielfältigere Paket *javax.swing* ersetzt. Swing Komponenten werden direkt durch Javacode gezeichnet, sind somit weniger auf die darunterliegende Plattform angewiesen und werden daher als leichtgewichtige Komponenten bezeichnet.

Die NetBeans Plattform verwendet zur Visualisierung das Swing Paket. Auch wenn Swing prinzipiell nichts mit der NetBeans Plattform zu tun hat, wird im weiteren Verlauf NetBeans und Swing als eine Einheit behandelt.

Da die meisten Anwendungen immer wiederkehrende Anforderungen haben, wie zum Beispiel das Bereitstellen von Menüs, Dokumentenmanagement, Änderungen von Einstellungen und ähnlichem, stellt die NetBeans-Plattform entsprechende Funktionen zur Verfügung. Der Entwickler erzeugt den Anwendungscode als ein oder mehrere NetBeans-Module und fügt diese der Plattform hinzu.

Dies erleichtert die Arbeit des Entwicklers, da er sich nicht so sehr mit der Infrastruktur der Anwendung befassen muss, sondern sich mehr auf die Geschäftslogik der Anwendungsdomäne konzentrieren kann.

Zu den Nutzern, der NetBeans Plattform gehören große Firmen, wie Boeing, Chrysler, sowie das US-Verteidigungsministerium[31].

[31] http://netbeans.org/features/platform/showcase.html (letzter Abruf 10.11.2010)

4.3.2 Bewertung nach Kriterienkatalog

Editorunterstützung in der Entwicklungsumgebung

Die NetBeans Plattform umfasst einen Satz von Bibliotheken, die prinzipiell auch ohne spezielle Entwicklungsumgebung genutzt werden können. Sinnvoll wird ein Einsatz aber erst, wenn auch die NetBeans IDE verwendet wird.

Die NetBeans IDE ist eine vollwertige Entwicklungsumgebung, die neben Java auch noch eine Vielzahl anderer Sprachen unterstützt und selbst auf der NetBeans Plattform aufsetzt. Speziell zur Entwicklung von Anwendungen auf der NetBeans Plattform, bietet sie eine Projektvorlage, einen UI-Designer für Swing Oberflächen, sowie diverse Assistenten die Arbeitsabläufe durch Codegenerierung vereinfachen.

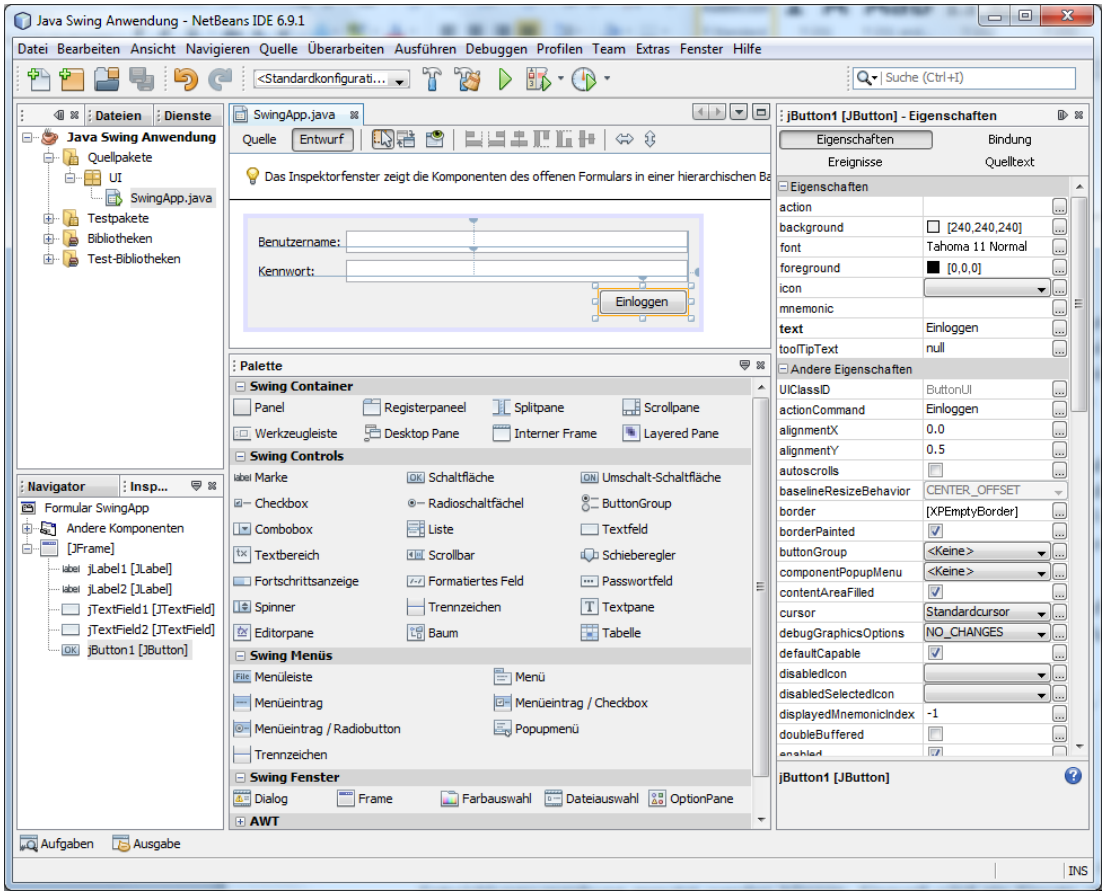


Abbildung 19: Swing Designer in NetBeans 6.9.1

Von der Palette aus können Steuerelemente auf einem Form frei positioniert werden, indem sie per Drag & Drop platziert werden. Im Hintergrund generiert der GUI-Editor entsprechenden XML-Code aus dem wiederum Javacode erzeugt wird. Es lassen sich keine Änderungen am generierten Programmcode vornehmen, da dieser schreibgeschützt ist und bei jeder Änderung des Formulars aus dem XML-Code heraus neu erzeugt wird.

Bei der Positionierung der Elemente wird der speziell für den GUI-Editor entwickelte `GroupLayout` Layoutmanager [25] verwendet, der es ermöglicht, Elemente so zu konfigurieren, dass sie im Fenster dynamisch skalieren, wenn sich die Größe der

Anzeigefläche verändert. Die Konfiguration gelingt dabei vollständig über die Menüs der IDE.
Steuerelementeigenschaften lassen sich tabellarisch einsehen und verändern. In einem Inspektor wird die gesamte Elementhierarchie der Benutzerschnittstelle angezeigt, sodass Elemente auch auf Grund ihrer logischen Position innerhalb der Oberfläche ausgewählt werden können.
Die Palette mit den Steuerelementen kann um benutzerdefinierte Komponenten erweitert werden, sofern diese der JavaBeans Spezifikation folgen. Dies bietet auch Drittanbietern die Möglichkeit den Funktionsumfang von Swing zu erweitern und unmittelbar in der NetBeans IDE nutzbar zu machen.
Über den Pluginassistenten lässt sich aus über 70 Plugins der Funktionsumfang der IDE erweitern.
Sowohl die NetBeans IDE als auch die NetBeans Plattform sind quelloffene Produkte und unter der CDDL und GPL Lizenz nutzbar.

Trennung von Darstellung und Logik
Die Swing Steuerelemente verwenden das MVC-Muster zur Trennung von Darstellung und Logik. In folgendem Listing visualisieren zwei Textareas das gleiche Model (`Document`). Änderungen an einer der beiden `TextAreas`, betreffen unmittelbar auch die andere.

Java Code

```
Document doc  = new DefaultStyledDocument();     // MODEL object
JTextArea ta1 = new JTextArea(doc);              // VIEW object 1
JtextArea ta2 = new JTextArea(doc);              // VIEW object 1
```

Listing 19: MVC in Swing.

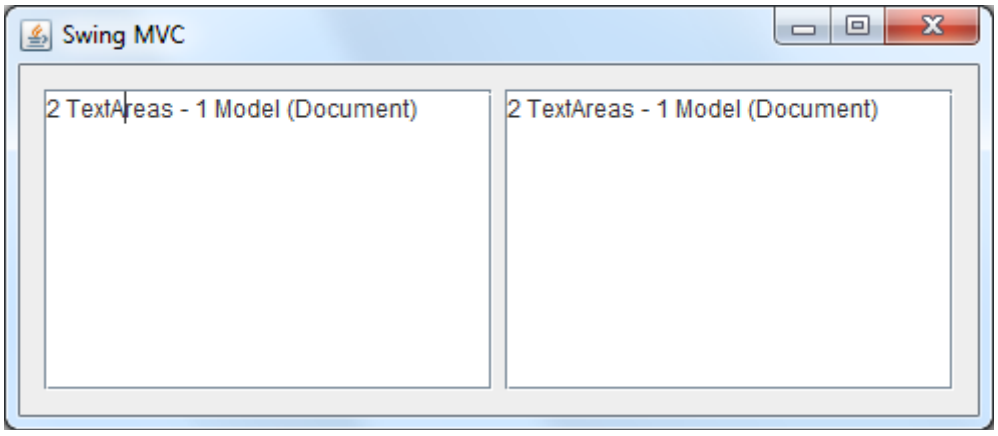


Abbildung 20: Beide TextAreas verwenden das gleiche Model

Swing bietet für alle Steuerelemente abstrakte Defaultmodels an, die als Grundlage für eigene Modelle zur Datenhaltung verwendet werden können.

Arbeitsteilung von Designern und Entwicklern
Die Oberfläche einer Swing Anwendung von Designern gestalten zu lassen, ist nicht möglich, da erhebliche Programmierkenntnisse erforderlich sind. Näheres dazu unter Look & Feel.

Modularität
Die NetBeans Plattform besitzt ein spezielles Modulsystem, das als zentrale Komponente einen Laufzeitcontainer zur Verfügung stellt, der für das Laden und Verwalten der Module innerhalb der Anwendung zuständig ist.
Das NetBeans Modulsystem verwendet bekannte Konzepte aus der Java-Entwicklung. Ein Modul enthält eine Manifest Datei (manifest.mf), eine deklarative Schnittstellenbeschreibung des Moduls (layers.xml), den Programmcode in Form von class Dateien, sowie benötigte Ressourcen wie Bilder Icons, etc. All das ist in einer jar-Datei verpackt.
Die Manifest Datei ist eine textuelle Beschreibung des Moduls und seiner Umgebung und wird beim Laden des Moduls vom NetBeans Modulsystem zuerst gelesen. Neben Abhängigkeiten des Moduls wird auch ein eindeutiger Identifier in der Manifestdatei vergeben, um ein Modul identifizieren und versionieren zu können.

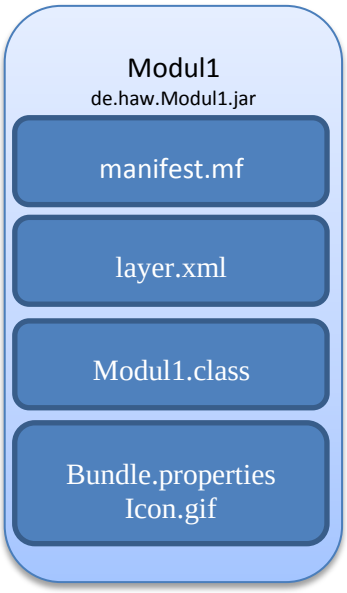


Abbildung 21: Ein NetBeans-konformes Modul

Die Layers-Datei dient der zentralen Konfiguration des Moduls und stellt die Schnittstelle des Moduls zur deklarativen Integration in der NetBeans Plattform dar. In ihr werden beispielsweise alle Aktionen definiert, die ein Modul ausführen kann, um sie an Menüeinträge binden zu können.
Das Modulsystem ist mit Hilfe der Konfigurationsdateien in der Lage, die Abhängigkeiten des Moduls aufzulösen und eine Installation in der Anwendung vorzunehmen. Des Weiteren können Module an Hand der Versionsinformationen aktualisiert oder deinstalliert werden.

Testbarkeit
Für automatisierte Tests von Swing Anwendungen bietet sich das auf JUnit basierende *UISpec4J* an. Steuerelemente können programmatisch aus dem Fenster extrahiert werden, um Aktionen (Klick, etc.) auszuführen oder ihren Zustand abzufragen.

Java Code
```
Window window = getMainWindow();
Button button = window.getButton("Einloggen");
TextBox input_benutzer = window.getInputTextBox("input_benutzer");
TextBox input_kennwort = window.getInputTextBox("input_kennwort");
TextBox status = window.getInputTextBox("status");
input_benutzer.setText("foo"); input_kennwort.setText("bar");
button.click();
assertEquals("Login nicht erfolgreich.", status.getText());
```
Listing 20: Test einer Swingoberfläche, siehe Swing MVP Projekt auf DVD

Generische Steuerlemente

Die Swing Bibliothek umfasst Steuerelemente, die den Standardanforderungen von Geschäftsanwendungen gerecht werden. Benutzerdefinierte Controls können über die Klasse `Component` erstellt werden.

Eine Sammlung weiterer Steuerelemente ist im Projekt „Java Desktop“[32] auf java.net zu finden.

Im Vergleich zu anderen Benutzerschnittstellenbibliotheken ist der Funktionsumfang an Steuerelementen jedoch nicht sehr umfangreich, sodass man schnell gezwungen ist, externe Bibliotheken einzubinden. Beispielsweise verfügt Swing von Haus aus über keine Kalenderkomponente (Datepicker).

Look & Feel

Die in Swing enthaltenen Steuerelemente greifen für die Visualisierung nicht auf Betriebssystemkomponenten zurück, sondern werden vollständig durch Javacode gerendert. In den JREs sind standardmäßig mehrere Look & Feel (L&F) Komponenten enthalten, die über einen `UIManager` ausgewählt werden können. Sogar zur Laufzeit der Anwendung ist es möglich, dass Aussehen der Anwendung durch einen anderen Look & Feel auszutauschen. Um plattformübergreifend ein einheitliches Layout sicherzustellen, bietet sich der Metal L&F an, der auch standardmäßig von Swing verwendet wird.

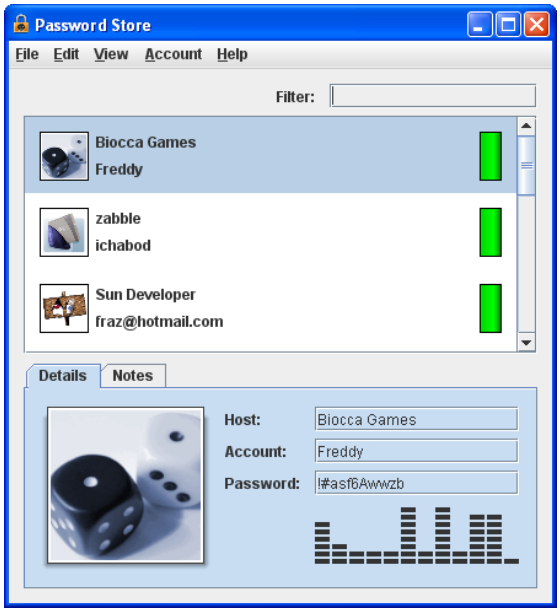


Abbildung 22: Metal L & F

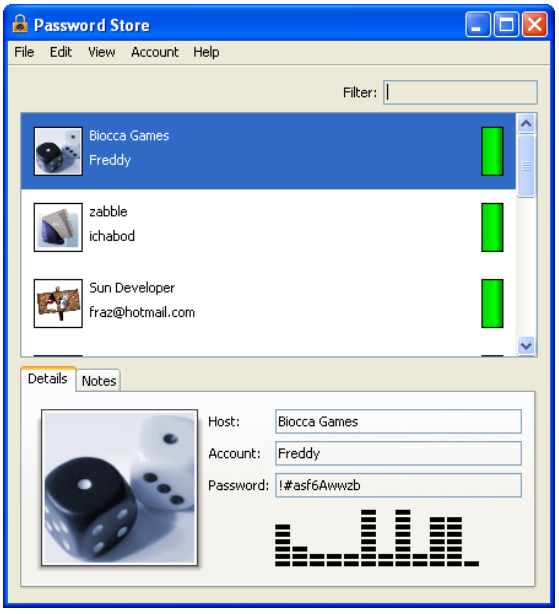


Abbildung 23: Windows L&F

Das Erstellen eigener Look & Feels ist mit Hilfe der Pluggable Look & Feel Api aus dem `javax.swing.plaf` Paket möglich aber auch sehr aufwändig.

[32] https://javadesktop.dev.java.net/ (letzter Abruf 10.11.2010)

Ereignisse und Befehle
Neben einfachen Ereignissen, die von Steuerelementen ausgelöst werden, bietet Swing mit dem `Action` Interface auch eine Abstraktion an, um anwendungsübergreifende Befehle definieren zu können. Das „SwingMVP“ Projekt verwendet das `Action` Interface.

Java Code

```
void     addPropertyChangeListener(PropertyChangeListener listener)
Object   getValue(String key)
Boolean  isEnabled()
void     putValue(String key, Object value)
void     removePropertyChangeListener(PropertyChangeListener listener)
void     setEnabled(boolean b)
```

Listing 21: Das Action Interface in Swing

Die NetBeans Plattform unterstützt das Konzept der Actions durch Bereitstellung konkreter Implementierungen. Über einen Wizard innerhalb der IDE ist es möglich, Actions anzulegen, die automatisch in der layers.xml für das Modul registriert und somit publik gemacht werden. Aktionen werden kontextsensitiv aktiviert und deaktiviert und können über eine Lokalisierungsschnittstelle in andere Sprachen übersetzt werden.

Datenbindung und Validierung
Weder Swing noch die NetBeans Plattform unterstützen von sich aus die Möglichkeit Steuerelemente mit Objekten zu verknüpfen. Unter dem Namen Beans Binding wurde im JSR 296[33] begonnen, einen solchen Mechanismus auf Basis von Java Beans zu realisieren. Das Projekt hat aber in den letzten 3 Jahren keine neue Version veröffentlicht und kann daher als „inaktiv“ bezeichnet werden.
Auch vorgefertigte Validierungsmaßnahmen für Benutzereingaben sucht man vergeblich. Mit JGoodies Validation, sowie der Simple Validation API, stehen jedoch zwei Bibliotheken zur Verfügung, die noch aktiv weiterentwickelt werden und sich für den Produktiveinsatz lohnen.

Moderne Bedienkonzepte
Die Datatransfer API ermöglicht es, Cut/Copy/Paste sowie Drag & Drop Operationen durchzuführen. Multitouch wird nicht nativ unterstützt.
Am Fraunhofer Institut wird derzeit an einer quelloffenen Multitouch Bibliothek für Java gearbeitet[34].

Barrierefreiheit
Swing stellt mit der Accessibility API eine Möglichkeit dar, die Anwendung mit Hilfe von Screenreadern zu steuern. Swing Anwendungen müssen die Accessibility API implementieren, damit über die Java Access Bridge ein Zugriff auf Swingkomponenten erfolgen kann. Für weitere Information siehe [26]. Eine solche Access Bridge ist nur für Windows Betriebssysteme verfügbar.

[33] http://jcp.org/en/jsr/detail?id=295 (letzter Abruf 10.11.2010)

[34] http://www.mt4j.org/ (letzter Abruf 10.11.2010)

Internationalisierung
Die NetBeans IDE bietet einen komfortablen Lokalisierungsassistenten an, um Beschriftungen für Steuerelemente zu extrahieren und extern in einer Ressourcendatei zu lagern, damit diese je nach Bedarf mit der Klasse `ResourceBundle` geladen werden können.
Bei einer Ressourcendatei handelt es sich um eine einfache Textdatei mit Schlüssel/Wert Paaren, die vom Übersetzer mit einem beliebigen Texteditor bearbeitet werden können.

Verteilung
Java Anwendungen können über herkömmliche Setuproutinen oder mit der Java Web Start Technologie lokal auf dem Rechner des Benutzers installiert werden.
Um eine mit Java Web Start verteilte Anwendung zu starten, ruft der Benutzer mit einem Webbrowser eine Konfigurationsdatei mit der Endung *jnlp* auf. JNLP steht für *Java Network Launching Protocol* und ist ein XML Format, das festlegt, wie Anwendungen per Java Web Start aufgerufen werden. JNLP-Dateien enthalten Informationen wie den Ablageort von JAR-Dateien, den Namen der Hauptklasse einer Anwendung und zusätzliche Parameter für das aufzurufende Programm. Bei Bedarf kann auch eine Java Laufzeitumgebung installiert werden, sofern diese auf dem System noch nicht vorhanden ist.
Bei jedem Start einer Java-Web-Start-Anwendung kann überprüft werden, ob neuere Komponenten vorliegen. So kann der Anwender stets mit der aktuellen vom Autor des Programms zur Verfügung gestellten Version arbeiten. Eine einmal heruntergeladene Version einer Anwendung bleibt solange in einem Cache auf der Festplatte des Clients, bis bei der Prüfung festgestellt wird, dass eine neue Version vorliegt und diese geladen werden muss. Somit werden unnötige Downloads verhindert und trotzdem sichergestellt, dass immer die aktuelle Programmversion läuft [27].

Angebote von Drittanbietern
Das Projekt *SwingX* hat mehrere Steuerelemente hervorgebracht, die die Entwicklung von Swinganwendungen erheblich vereinfachen. Zu den Highlights zählen erweiterte Tabellenfunktionen, sowie der unter Swingentwicklern schmerzlich vermisste Datepicker. Das Projekt ist unter der Lesser General Public License veröffentlicht.
Die *Flamingo Swing component suite*, welche unter der BSD License erhältlich ist, ist eine Portierung der Office Ribbon Oberfläche auf Basis von Swing.
Zu den wenigen kommerziellen Anbietern zählt die Firma JIDE Software, welche schon seit 2002 am Markt etabliert ist und viele qualitativ hochwertige Swing-Komponenten vertreibt. Einige der Komponenten sind frei (JIDE Common Layer (Open Source Project)), weitere wie das Docking-Framework, Action Framework, (Pivot) Grids, Code Editor und weitere gehören zum kommerziellen Teil. Die Komponenten aus dem JIDE Common Layer stehen unter dual-license: GPL und free commercial license.

Hilfe und Support
Auf der NetBeans Homepage werden Dokumentation, Tutorials und Screencasts angeboten, um Entwicklern bei der Arbeit mit der NetBeans Plattform zu unterstützen. In dem dazugehörigen Forum im Communitybereich werden Fragen in der Regel innerhalb von wenigen Stunden kompetent beantwortet.

Der Autor Heiko Böck hat mit dem Buch „NetBeans Platform 6 – Rich-Client-Entwicklung mit Java“ [28] ein umfassendes, praxisnahes und gut zu lesendes Werk für die Anwendungsentwicklung mit der NetBeans Plattform verfasst.

5 Fazit

Um eine Entscheidung fällen zu können, welche der drei Technologien für eine Anwendung eingesetzt werden soll, ist als erstes zu berücksichtigen, auf welcher Zielplattform die Anwendung verwendet wird. Wird eine möglichst hohe Plattformunabhängigkeit angestrebt, kann die Entscheidung nur Java und die NetBeans Plattform lauten. Virtuelle Maschinen für Java gibt es für alle erdenklichen Systeme, sodass ohne Modifizierungen des Quellcodes oder Neukompilierung die Anwendung lauffähig ist. Da Java Anwendungen jedoch nur über die virtuelle Maschine mit dem Betriebssystem kommunizieren können, lassen sie sich nicht so tief in die Betriebssystemumgebung integrieren. Beispielsweise ist es nur mit erheblichem Aufwand möglich, die Windows 7 Sprunglisten oder die Taskbar von Linux Desktopumgebungen in Swing Anwendungen zu verwenden.
Die Entwicklung einer Swingoberfläche mit Hilfe des NetBeans GUI Editors geht sehr leicht von der Hand, da sowohl die Ausrichtung als auch die Konfiguration von Steuerelementen sehr intuitiv gelöst ist. Einen großen Nachteil stellen jedoch die fehlenden Datenbindungsmöglichkeiten dar, sodass viel manueller Aufwand betrieben werden muss, um Model und Darstellung synchron zu halten. Insbesondere die Entwicklung datenintensiver Geschäftsanwendungen mit vielen Masken und Tabellen wird dadurch sehr erschwert.
Das gesamte Swing Toolkit hat seit Java 1.4 aus dem Jahr 2002 so gut wie keine nennenswerten neuen Funktionen erhalten und auch die API verwendet keine der neuen Sprachfeatures wie Generics oder Annotations. Es ist auch eher unwahrscheinlich, dass ein Update des in die Jahre gekommenen Toolkits zu erwarten ist. Insbesondere seit der Übernahme von Sun Microsystems durch Oracle scheint sogar die Zukunft von Java mehr als ungewiss zu sein. Die Veröffentlichung von Java 7 war anfangs für 2008 geplant, wurde dann auf Frühjahr 2010 und noch einmal auf den Herbst 2010 hinausgeschoben. Seit Oktober 2010 steht nun jedoch fest, dass mit einer Veröffentlichung frühestens Mitte nächsten Jahres zu rechnen ist[35].
Die NetBeans Plattform macht alles in allem einen sehr durchdachten und ausgereiften Eindruck. Durch die ausführliche Dokumentation und gute Integration in die NetBeans IDE, gelingt die Einarbeitung sehr schnell. Im Gegensatz zum Swing Toolkit wird die NetBeans Plattform noch aktiv weiterentwickelt, sodass im halbjährlichen Rhythmus neue Versionen veröffentlicht werden.
Wie eingangs bereits erwähnt, ist Java die erste Wahl, wenn eine hohe Plattformunabhängigkeit gewünscht ist. Ist dies nicht der Fall und die Zielplattform ist eine reine Windowsumgebung, so verkürzt sich durch die bessere Integration die Entwicklungszeit erheblich, wenn auf Microsoft Technologien aufgebaut wird. Die zu

[35] http://blogs.sun.com/mr/entry/rethinking_jdk7 (letzter Abruf 10.11.2010)

beantwortende Frage ist dann, ob die ältere und etablierte Windows Forms API oder die neue Windows Presentation Foundation verwendet werden soll.
WPF ist erst ab Windows XP verfügbar, dies bedeutet, dass nur Windows Forms verwendet werden kann, wenn ältere Betriebssysteme unterstützt werden müssen.
Unterliegt das Projekt nicht solchen Einschränkungen, muss abgewogen werden, inwiefern die Anwendung von den WPF-Eigenschaften, wie Hardwarebeschleunigung und Unterstützung von Multimediainhalten, profitiert. Denn auch wenn WPF gegenüber der Windows Forms Bibliothek in den technischen Möglichkeiten eindeutig überlegen ist, darf der Einarbeitungsaufwand auf Grund des neuartigen Programmiermodells nicht unterschätzt werden [29]. Einsteiger in die WPF müssen sich zunächst intensiv mit der deklarativen Markupsprache XAML auseinandersetzen, ehe sie mit der eigentlichen Oberflächengestaltung beginnen können. Durch das Fehlen von RAD Designern, die das einfache Positionieren von Steuerelementen per Drag & Drop ermöglichen, ist bereits für kleinere Masken erheblicher manueller Aufwand von Nöten. Insbesondere in klassischen Geschäftsanwendungen, die nicht zwingend auf die modernen Konzepte von WPF angewiesen sind, ist der Mehraufwand sicherlich nicht gerechtfertigt. Des Weiteren kann WPF im Terminalserverbetrieb zu Leistungseinbußen durch das DirectX Rendering führen [23]. Auch wenn Microsoft derzeit an der Lösung des Problems arbeitet, muss dies bei der Verteilung der Anwendung berücksichtigt werden.
Die bisher aufgeführten Punkte raten eher zum Einsatz von Windows Forms, da insbesondere die Entwicklungsgeschwindigkeit zu Beginn rasanter ist. Auf Grund der weiten Verbreitung und der guten Unterstützung durch Dritthersteller ist davon auszugehen, dass auch in der Zukunft Windows Forms Anwendungen durch Microsoft unterstützt werden, wenn auch die API wahrscheinlich keine nennenswerten Erweiterungen mehr erwarten wird.
In der WPF liegt ganz klar die Zukunft der Oberflächenprogrammierung im .Net Framework. Die API und die Werkzeuge werden mit jeder neuen Version des Frameworks oder der Visual Studio IDE besser. Durch die zusätzlich zur Verfügung stehende Expression Suite ist es auch geschulten Designern möglich, ihren Beitrag zu einer gelungenen Benutzerschnittstelle zu liefern.
Mit Commands, Databinding sowie dem MVVM Muster haben Entwickler mächtige Werkzeuge zur Hand, um modulare und interaktive Benutzerschnittstellen zu erzeugen, die obendrein noch größtmöglichen Nutzen aus der Hardware ziehen.

A. Abkürzungsverzeichnis

API	Application programming interface
AWT	Abstract Window toolkit
C#	Eine objektorientierte Programmiersprache auf der .Net Plattform
OOP	Objektorientierte Programmierung
CSV	Comma Separated Value
GPL	GNU General Public License
IDE	Integrated Development Environment
ISO	International Organization for Standardization
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View ViewModel
RAD	Rapid Application Development
RCP	Rich Client Platform
SVG	Scalable Vector Graphics
SWT	Standard Widget Toolkit
UI	User interface
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

B. Glossar

Barrierefreie Software	Software, die von allen Nutzern unabhängig von körperlichen oder technischen Möglichkeiten uneingeschränkt (barrierefrei) genutzt werden kann.
C#	Eine objektorientierte streng typisierte Programmiersprache auf der .Net Plattform
Dependency Injection	Dependency Injection (DI) ist ein Entwurfsmuster und dient in einem objektorientierten System dazu, die Abhängigkeiten zwischen Komponenten oder Objekten zu minimieren.
Drag & Drop	Dt. „Ziehen und Fallenlassen“, ist eine Methode zur Bedienung grafischer Benutzeroberflächen von Rechnern durch das Bewegen grafischer Elemente mit der Maus
Java	Eine objektorientierte streng typisierte Programmiersprache, die auf der Java virtuellen Maschine läuft
Java Beans	JavaBeans sind Software-Komponenten für die Programmiersprache Java. JavaBeans entwickelten sich aus der Notwendigkeit heraus, GUI-Klassen (AWT, Swing) einfach instanziieren zu können. JavaBeans werden auch als Container zur Datenübertragung verwendet.
Java Virtuelle Maschine	Der Java Compiler übersetzt den Quellcode in Bytecode, der von der virtuellen Maschine ausgeführt wird. Virtuelle Maschine gibt es für eine Vielzahl von Betriebssystemen. Der Erzeugte Bytecode ist somit portabel und betreibsystemübergreifend nutzbar.
Look & Feel	(dt. Aussehen und Handhabung) Bezeichnet meist durch Hersteller oder Konsortien standardisierte Design-Aspekte einer Software, wie zum Beispiel Farben, Layout, Fontgröße, die Benutzung von grafischen Elementen (widgets), Bedienung über die Tastatur usw., in Software mit grafischer Benutzeroberfläche oder Webseiten.
Microsoft Corporation	Softwarehersteller mit Hauptsitz in Redmond, einem Vorort von Seattle (US-Bundesstaat Washington). Das Unternehmen ist bekannt für sein Betriebssystem Windows und seine Büro-Software Office.
NetBeans	In Java geschriebene Open Source Entwicklungsumgebung, die sich durch eine Vielzahl von Plugins erweitern lässt und auf der NetBeans Plattform läuft
Oracle Corporation	Einer der weltweit größten Softwarehersteller mit Hauptsitz Redwood Shores (Silicon Valley).
Partielle Klasse	Die Klassendefinition wird in mehrere physikalische Dateien aufgeteilt. Logisch gesehen macht das Aufteilen der Klasse auf mehrere Dateien für den Compiler keinen Unterschied, da zur Compiletime alle Definitionen zusammengeführt werden.
Sun Microsystems	Ein in Santa Clara ansässiger Hersteller von Computern und Software und seit dem 27. Januar 2010 eine hundertprozentige Tochter der Oracle Corporation.
Visual Studio	Entwicklungsumgebung aus dem Hause Microsoft mit Unterstützung für Visual Basic, C, C++, C++/CLI, C# und F#.

C. Literaturverzeichnis

[1]. **Parnas, D.L.** *On the Criteria To Be Used in Decomposing Systems into Modules.* [PDF] s.l. : Carnegie-Mellon University, 1972.

[2]. **Brooks, Jr., Frederick P.** *The Mythical Man-Month: Essays on Software Engineering.* s.l. : Addison-Wesley, 1982. 0-201-00650-2.

[3]. **Dijkstra, Edsger. W.** *The Humble Programmer.* [PDF] 1972.

[4]. **Bass, Len, Clements, Paul und Kazman, Rick.** *Software Architecture in Practice (2nd edition).* s.l. : Addison-Wesley, 2003.

[5]. **Fowler, Martin.** *Patterns für Enterprise Application-Architekturen.* s.l. : mitp, 2003.

[6]. **Hofmeister, Christine.** *Applied software architecture.* Seite 4 : s.n., 2000.

[7]. **Posch, Torsten.** *Basiswissen Softwarearchitektur.* s.l. : dpunkt, 2004. 3898642704.

[8]. **Dijkstra, Edsger W.** *On the role of scientific thought.* [PDF] 1982.

[9]. **Hunt, Thomas.** *The Pragmatic Programmer.* s.l. : Addison-Wesley, 1999. 020161622X.

[10]. **Alexander, Christoph.** *A Pattern Language.* s.l. : Oxford University Press, 1977.

[11]. **Gamma, Erich, Helm, Richard und E. Johnson, Ralph.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley Longman, 1994. 0201633612.

[12]. **Reenskaug, Trygve.** [Online] 1979. http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html. (letzter Abruf 10.11.2010)

[13]. **Boodhoo, Jean-Paul.** Model View Presenter. [Online] 2006. http://msdn.microsoft.com/en-us/magazine/cc188690.aspx. (letzter Abruf 10.11.2010)

[14]. **Fowler, Martin.** Passive View. [Online] 2006. http://www.martinfowler.com/eaaDev/PassiveScreen.html. (letzter Abruf 10.11.2010)

[15]. **Feathers, Michael.** *The Humble Dialog Box.* [PDF] s.l. : Object Mentor, Inc., 2002.

[16]. **Memon, Atif M., Pollack, Martha E. and Lou, Soffa Mary.** *Using a Goal-driven Approach to Generate Test Cases for GUIs.* [PDF] University of Pittsburgh : Dept. of Computer Scince, 1999.

[17]. **Clarke, James M.** *Automated test generation from a Behavioral Model.* [PDF] Naperville : Lucent Technologies, 1998.

[18]. **Microsoft.** WPF Application Quality Guide. [Online] 2009. http://windowsclient.net/wpf/white-papers/wpf-app-quality-guide.aspx. (letzter Abruf 10.11.2010)

[19]. **MSDN.** Visual Studio Installer Deployment. [Online] 2010. http://msdn.microsoft.com/en-us/library/2kt85ked.aspx. (letzter Abruf 10.11.2010)

[20]. **MacDonald, Matthew.** *Pro .Net 2.0 Windows Forms and Custom Controls in C#: From Professional to Expert.* s.l. : Apress, 2004. 1590594398. (letzter Abruf 10.11.2010)

[21]. **Gossman, John.** Introduction to Model/View/ViewModel pattern for building WPF apps. [Online] 2005. http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx. (letzter Abruf 10.11.2010)

[22]. **MSDN.** Windows Touch Api. [Online] 30. 6 2010. http://msdn.microsoft.com/en-us/library/dd371413(VS.85).aspx. (letzter Abruf 10.11.2010)

[23]. **Huber, Thomas-Claudius.** *Windows Presentation Foundation: Das umfassende Handbuch.* s.l. : Galileo Computing, 2010.

[24]. **Qian, Kai, Fu, Xiang und Tao, Lixin.** *Software Architecture and Design Illuminated* . s.l. : Jones & Bartlett, 2010.

[25]. **GroupLayout.** [Online] 2010. http://download-llnw.oracle.com/javase/6/docs/api/javax/swing/GroupLayout.html. (letzter Abruf 10.11.2010)

[26]. **Accessibility, Java SE Desktop.** [Online] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140174.html. (letzter Abruf 10.11.2010)

[27]. **Web Start, Java SE Desktop.** [Online] http://download.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/contents.html. (letzter Abruf 10.11.2010)

[28]. **Böck, Heiko.** *NetBeans Platform 6 - Rich-Client-Entwicklung mit Java.* s.l. : Galileo Computing, 2008.

[29]. **Januszewski, Karsten.** Hitting the Curve: On WPF and Productivity. [Online] 2006. http://blogs.msdn.com/b/karstenj/archive/2006/04/05/curve.aspx. (letzter Abruf 10.11.2010)

[30]. **MacDonald, Matthew.** *Pro WPF in C#.* 2010.

[31]. **Group, Standish.** [Online] 2010. http://www.standishgroup.com/. (letzter Abruf 10.11.2010)

D. Abbildungsverzeichnis

E. Listings

Inhalte der DVD

Model View Presenter

- Umsetzung des MVP Pattern mit Windows Forms
- Umsetzung des MVP Pattern mit Windows Presentation Foundation
- Umsetzung des MVP Pattern mit Java Swing

Model View ViewModel

- Umsetzung des MVVM Pattern mit Windows Presentation Foundation

Benutzerschnittstellentest

- Test der Swing Anwendung mit UISpec4J
- Test der Windows Forms Anwendung mit Windows.Automation.

Bachelorarbeit im PDF Format

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 15.11.2010

Ort, Datum	Verfasser