# Entwickeln für Windows Phone 7

Am Beispiel einer OpenStreetMap Anwendung

Martin Rauscher

18. Mai 2011

## 1 Einführung

Mit Windows Phone 7 (WP7) hat Microsoft sein Betriebssystem für Mobilsysteme komplett überarbeitet, um der Herausforderung durch Android und iOS gewachsen zu sein.

Mit dem neuen Betriebssystem wird auch ein neues Entwicklungsmodell eingeführt, das hauptsächlich auf Microsofts Silverlight-Technologie basiert.

Im Rahmen dieser Arbeit soll WP7 - und die Entwicklung dafür - vorgestellt werden.

### 1.1 Vorstellung der Beispielanwendung

Im Rahmen dieser Arbeit wurde eine Anwendung erstellt, die das Betrachten von Karten von OpenStreetMap (und anderen Anbietern) ermöglicht. Sie erlaubt die Steuerung durch die üblichen Multi-Touch-Gesten, das suchen nach Orten und das Verwalten Favoriten. Des Weiteren können Routen geplant werden.

Alle Beispiele in dieser Arbeit sind aus dem Quelltext dieser Anwendung. Für nähere Informationen zur “OpenStreetApp” wird auf [1] verwiesen.

### 1.2 Rahmen dieser Arbeit

Es gibt zwei Arten von WP7 Apps[1]:

1. Apps auf Basis von Silverlight

[1] Für spezielle Microsoft Partner ist es auch noch möglich halb-native Anwendungen zu schreiben.

2. oder auf Basis des XNA Frameworks - eine gekapselte DirectX-Schnittstille, welche hauptsächlich für die Entwicklung von Spielen verwendet wird.

Diese Arbeit beschäftigt sich ausschließlich mit der Entwicklung mit SILVERLIGHT, da das Entwicklungsmodell mit XNA gänzlich anders ist und es nur in seltenen Fällen Sinn macht mit XNA nicht-Spiele zu entwickeln.

# 2 Windows Phone 7

Windows Phone 7 ist hinsichtlich der Benutzeroberfläche als auch der Anwendungsplattform ein vollständige Neuentwicklung. Der Kern des Betriebssystems hingegen ist eine veränderte Version von Windows EC 7.0, dem Nachfolger von Windows CE 6, worauf Windows Mobile 6.5 beruht.[2]

Gründe nicht auf das bestehende Windows Mobile aufzusetzen waren

- UI-Konzept zu stark Desktoporientiert
- Fehlende Multi-Touch Unterstützung
- Entwicklung moderner Anwendungen schwierig

WP7 begegnet diesen Problemen mit einer starken Multi-Touch-Unterstützung in SILVERLIGHT und vordefinierten Kontrollelementen, die es leicht machen sollen moderne Anwendungen zu schreiben, die sich in das UI Konzept einfügen.

# 3 SILVERLIGHT und die WINDOWS PRESENTATION FOUNDATION

Das .NET Framework auf WP7 Handys basiert auf dem - vom Desktop bekannten - SILVERLIGHT, welches wiederum auf der WINDOWS PRESENTATION FOUNDATION basiert. In den folgen Abschnitten soll kurz aufgezeigt werden, wie die beiden Frameworks entstanden sind und was die groben Unterschiede sind.[3]

## 3.1 WINDOWS PRESENTATION FOUNDATION (WPF)

Ende 2006 stellte Microsoft die finale Version 3.0 des .NET Framwork vor. Mit diesem wurde zum ersten mal die, bis dahin unter dem Codenamen Avalon entwickelte, WINDOWS PRESENTATION FOUNDATION (WPF) der Öffentlichkeit zugänglich gemacht. Microsoft hoffte das Entwickler, insbesondere für das nur zwei Monate

später erscheinende Windows Vista, nun hauptsächlich mit .NET und WPF entwickeln würden. Bis heute ist allerdings die Akzeptanz von WPF im Consumerbereich eher gering.

Mit WPF führte Microsoft erstmals ein deklaratives Modell zur Gestaltung von Benutzeroberflächen ein. Dabei wird die GUI und die Datenbindungen mit einem auf XML basierenden Format beschrieben. Dies erhöht die Wartbarkeit und Wiederverwendbarkeit verglichen mit den bisher verwendeten, zum Teil riesigen, automatisch generierten Funktionen enorm.

Eine weitere große Neuerung ist die Abwendung von GDI(+) für das GUI-Rendering. Mit WPF wird erstmals voll auf (hardwarebeschleunigtes) DirectX gesetzt. Das hat den Vorteil, dass komplexe Oberflächen gerendert, transformiert und mit Effekten versehen werden können, ohne dass die CPU zusätzlich belastet wird.

## 3.2 Silverlight (WPF/E)

2007 stellte Microsoft Silverlight vor, welches, plakativ gesprochen, eine stark reduziert Version des .NET Framework und eine Untermenge[2] der WPF ist.

Auch wenn Silverlight oft als Konkurenz zu Adobes Flash Player gesehen wird, wurde es hauptsächlich mit dem Ziel entwickelt so genannte Rich Internet Applications (RIAs) zu ermöglichen, die hauptsächlich im Geschäftsbereich ihren Einsatzzweck haben. Der einzige Bereich in dem Silverlight momentan tatsächlich in direkter Konkurrenz zu Flash steht ist die Videowiedergabe. Sowohl Smooth Streaming[3] als auch hardwarebeschleunigte Videowiedergabe waren zuerst in Silverlight möglich. Deshalb hat Microsoft einige prominente Unterstützter gefunden: U.a. wurden die Olympischen Winterspiele von 2008 und einige amerikanische Sportgroßereignisse exklusiv via Silverlight gestreamt; außerdem setzt das deutsche Videoportal “Maxdome” auf Silverlight.

Silverlight wurde vom Grund auf Plattformunabhängig konzipiert. Microsoft stellt es für die meisten Windows Versionen sowie für Mac OS X zur Verfügung. Desweiteren ist ein Großteil der Spezifikationen und einige Teile des Codes öffentlich zugänglich. Darauf aufbauend entwickelt das Mono Projekt unter Leitung von Novell einen Linux Port. “Moonlight” ist momentan Kompatibel mit Silverlight 3.[4]

[2] Aufgrund von unterschiedlichen Releasezyklen hat die aktuelle WPF einige Klassen nicht, die in Silverlight enthalten sind. Dies wurden aber über ein Silverlight Toolkit nachgereicht.

[3] Bei Smooth Streaming handelt es sich um eine Technik die Videomaterial dynamisch neu kodiert, um sich der zur Verfügung stehenden Bandbreite anzupassen.

### 3.3 Unterschiede

nochmal

Mit dem Ziel der Plattformunabhängigkeit gingen natürlich auch höhere Ansprüche an einen geringen Ressourcenverbrauch einher. Das hatte zur Folge, dass einige Funktionen - z.B. Hardwarebeschleunigung und einige Controls - erst in späteren Versionen zur Verfügung standen. Bis jetzt sind noch einige Konzepte, wie z.B. Trigger und implizite Styles, noch immer nur in WPF verfügbar.[4]

## 4 Entwicklen mit SILVERLIGHT

### 4.1 Struktur einer SILVERLIGHT Anwendung

Eine SILVERLIGHT Anwendung wird immer in Form einer in "*.XAP" umbenannten ZIP-Datei verteilt. In dieser Datei sind die vier Bestandteile einer App untergebracht:

- Das Manifest (WMAppManifest.xml), welches die Fähigkeiten der App festlegt.
- Das Anwendungsobjekt (App.cs und App.xaml), welches die zentrale Steuereinheit bildet
- Die verschiedenen "Seiten" der App
- Die Ressourcen (Bilder, Töne, etc.)

Wenn das Betriebssystem eine App startet, wird zuerst das Manifest eingelesen und daraus ein xxxxxxx

navigation

### 4.2 XAML

XAML ist eine deklarative Sprache zur Beschreibung von Datenstrukturen die insbesondere für SILVERLIGHT- und WPF-Benutzeroberflächen verwendet wird. Sie ist ein XML Dialekt, der um verschiedene Konzepte erweitert wurde:

[4] Mit der kürzlich vorgestellten Version 5 werden einige dieser Konzepte auch in SILVERLIGHT nachgezogen.

### 4.2.1 Komplexe Attributdefinitionen

In XML können Attributen nur Strings, Zahlen und Referenzen zugewiesen werden, während der Inhalt von Element beliebig komplex sein kann. In XAML können Attribute durch eine simple Erweiterung der Semantik von Elementnamen, ebenfalls beschrieben werden. Im Beispiel 1 wird so auf das Attribut `Background` über `Grid.Background` zugegriffen.

**Algorithmus 1** Beispiel für Komplexe Attributdefinitionen

```
<Grid Background="Transparent"/>
<!-- oder -->
<Grid>
    <Grid.Background>
        <SolidColorBrush Color="Transparent"/>
    </Grid.Background>
</Grid>
```

### 4.2.2 Attached Properties

Insbesondere bei der Beschreibung von Benutzeroberflächen steht man oft vor dem Problem, dass Elemente eine Eigenschaft besitzen, die nur innerhalb eines bestimmten Kontext sinnvoll sind.

Z.B. hat ein Button der in einem Canvas angeordnet ist eine X und eine Y Koordinate. Wird er aber außerhalb eines Canvas eingesetzt sind diese Eigenschaften u.U. nicht sinnvoll.

**Algorithmus 2** Beispiel für ein Attached Property

```
<Canvas>
    <Button Canvas.Top="100">
        <Canvas.Left>25</Canvas.Left>
        <TextBlock Text="Ein Knopf"/>
    </Button>
</Canvas>
```

XAML bietet für dieses Problem eine einfache Lösung: Genau wie bei den komplexen Attributdefinitionen wird der Name des Attributbesitzers verwendet um die Herkunft des Attributs zu qualifizieren. Der Wert des Attributs wird dabei in einer statischen Eigenschaft der entsprechenden Klasse gespeichert. [5]

Attached Properties sind eine Spezialform von Depency Properties (siehe 4.3.1).

### 4.2.3 Markup Extensions

Markup Extensions sind ein Konzept, dass es ermöglicht den XAML Parser in begrenztem Umfang zu erweitern, so dass die Verwendung komplexer Attributdefinitionen entfallen kann.

**Algorithmus 3** Beispiel für eine Markup Extension

```
<TextBlock Text="{StaticResource AppName}" />
<!-- oder -->
<TextBlock Text="{StaticResource ResourceKey=AppName}" />
<!-- vs. -->
<TextBlock>
    <TextBlock.Text>
        <StaticResource ResourceKey="AppName" />
    </TextBlock.Text>
</TextBlock>
```

Markup Extensions werden durch führende und endende geschweifte Klammern gekennzeichnet. Das erste Wort in den Klammern gibt die Klasse der zu verwendenden Extension an. Parameter werden nach dem Klassennamen gelistet und werden in der Form "Parameter=Wert" angegeben. Einzige Ausnahme ist der Standardparameter, der ohne Angabe seines Namens verwendet werden kann. Im Beispiel 3 ist der Standardparameter "ResourceKey".

SILVERLIGHT bietet - in der aktuellen Version - keine Möglichkeit eigene Markup Extensions zu definieren[5]. Es stehen nur die folgenden drei Vordefinierten zur verfügung: *Binding*, *StaticResource*, *DynamicResource*. Auf deren Bedeutung wird im folgenden Abschnitt eingegangen.

## 4.3 Datenbindung

Eine der größten Stärken von SILVERLIGHT ist die Möglichkeit Eigenschaften von Elementen auf einfache Weise an Attribute von Datenobjekten oder an Eigenschaften anderer GUI Elemente zu binden. Somit muss keine Code geschrieben werden um die Benutzeroberfläche bei Änderungen an den Daten aktuell zu halten.

[5] Auch diese Funktion ist für SL5 angekündigt.

**Algorithmus 4** Beispiel für eine Datenbindung

```
<TextBox Text="{Binding Path=Vorname,
                        Mode=TwoWay,
                        Converter={StaticResource conv1}}"/>
```

In Beispiel 4 kann man sehen wie die Definition einer Datenbindung mit Hilfe der {Binding} Markup Extension aussieht. Dabei bezieht sich die Datenbindung auf das Objekt das der aktuelle Datenkontext ist, da nichts anderes angegeben wurde. Dieser Kontext ergibt sich aus dem Element innerhalb dessen das Binding definiert wurde. Jedes GUI-Element[6] hat eine DataContext Eigenschaft, die es an seine Kind-Elemente weitervererbt, von ihnen aber auch überschrieben werden kann.

### 4.3.1 Dependency Properties

Ein Problem vieler Datenbindungs-Ansätze ist, dass sie Reflection[7] verwenden, da die Eigenschaften als String angegeben werden. Außerdem muss jede Klasse typischerweise ein Interface implementieren, dass die eventuellen Ziele einer Datenbindung benachrichtigt.

In Silverlight wird dieses Problem mit so genannten Depenency Properties (DPs) gelöst. Im Grunde sind DPs Dictionaries die als statische Eigenschaften in der Klasse definiert werden. Dieses weißt dann einem Objekt einen Wert zu.

Der große Vorteil von DPs ist, dass sich beliebige Objekt registrieren können, um benachrichtigt zu werden, wenn sich der Wert einer Eigenschaft eines bestimmten Objekts ändert; und zwar mit statisch getypten Vorher-Nachher-Werten.

Außerdem lassen sich bei der Definition von DPs Standardwerte festlegen und automatisches Vererben von Werten auf Kind-Objekte aktivieren.

### 4.3.2 Daten-Konverter

Es ist oft wünschenswert zwei Eigenschaften miteinander zu verbinden, die nicht den gleichen Datentyp haben. Das typische Beispiel hierfür ist die Text Eigenschaft eines Textblocks und ein Zahlwert eines Objekts. Im Gegensatz zu diesem einfachen Beispiel, bei dem die Konvertierung natürlich automatisch passiert, ist es oft komplizierter.

[6] Genauer, jede von FrameworkElement erbende Klasse.

[7] Reflection bezeichnet die Möglichkeit Objekte einer statisch getypten Sprache zur Laufzeit dynamisch zu untersuchen und zu verändern. Hierfür sind meistens aufwendige Aufrufe an die zu Grunde liegende Plattform zu richten.

**Algorithmus 5** Beispiel für einen Converter

```
public class VisibilityConverter : IValueConverter
{
   public Object Convert(Object value,
                  Type targetType,
                  Object parameter,
                  System.Globalization.CultureInfo culture)
   {
      return (bool)value == true ?
            System.Windows.Visibility.Visible
            : System.Windows.Visibility.Collapsed;
   }

   public Object ConvertBack(Object value,
                     Type targetType,
                     Object parameter,
                     System.Globalization.CultureInfo culture)
   {
      return (System.Windows.Visibility)value
         == System.Windows.Visibility.Visible;
   }
}
```

Die Lösung dieses Problems sind so genannte Konverter, die beliebige Typen in einander konvertieren können. Konverter sind Klassen, die das IValueConverter Interface implementieren. Im Beispiel 5 kann man exemplarisch sehen, wie ein Wahrheitswert in einen Aufzählungstyp - und umgekehrt - umgewandelt werden kann. Mit Hilfe der Convert und ConvertBack Methoden kann beliebig komplexer Code dazu verwendet werden Werte umzuwandeln.

### 4.3.3 Daten-Templates

Will man mehrere Objekte, z.B. in einer Liste, anzeigen, so reichen die Fähigkeiten von Konvertern normalerweise nicht aus. Für diese Anwendung bietet SILVERLIGHT Daten Templates an. Das sind Schablonen mit deren Hilfe beliebige Objekte durch (mehrere) UI Komponenten angezeigt werden können.

**Algorithmus 6** Beispiel für ein Daten Template

```
<DataTemplate>
    <StackPanel>
        <Image Source="{Binding Logo}"/>
        <TextBlock Text="{Binding Text}"/>
    </StackPanel>
</DataTemplate>
```

Wie im Beispiel 6 bereits angedeutet, können diese Templates beliebig komplex sein. Der Kontext von Datenbindungen innerhalb dieser Templates, ist dabei das Objekt für das das Template instanziiert wurde.

## 4.4 Resourcen

In jedem Projekt tauchen Teile auf, die an vielen Stellen wieder verwendet werden. Z.B. Bilder, Texte oder Vorlagen für bestimmte Stile. Ähnlich dem Konzept der Datenbindungen können in SILVERLIGHT Ressourcen einfach definiert und verwendet werden.

Jedes Steuerelement in SILVERLIGHT kann eine List von Ressourcen für sich und seine Kind-Elemente definieren. Außerdem können Anwendungsweite Ressourcen definiert werden - siehe Beispiel 7.

**Algorithmus 7** Beispiel Resourcendefinition

```
<Application.Resources>
    <System:String x:Key="AppTitle">
        Open Street App
    </System:String>
</Application.Resources>
```

In Beispiel 4 kann man sehen, wie auf eine Ressource innerhalb einer Datenbindungs-Definition verwiesen wird. Bei der Auflösung des Namens einer Ressource wird der Elementbaum von unten nach oben durchsucht, bis schlussendlich noch die anwendungsweiten Ressourcen durchsucht werden.

In den bisherigen Beispielen erfolgte der Verweise auf eine Ressource immer mittels {*StaticResource*}. Erwartungsgemäß existiert auch eine {*DynamicResource*}. Der Unterschied besteht im Ladeverhalten:

Statische Ressourcen werden nur ein mal beim Initialisieren des Elements geladen. Sollte sich also das worauf sich die Ressource bezieht verändern, würde das

die Bindung nicht widerspiegeln. Ein dynamischer Ressourcenverweis hingegen funktioniert genau wie eine Datenbindung.

## 4.5 Besonderheiten auf dem Handy

Die Entwicklung auf Mobilgeräten unterliegt zum Teil gänzlich anderen Anforderungen, als die Entwicklung für den Desktopbereich. Diese Unterschiede sind größtenteils nicht WP7 spezifisch, müssen hier aber dennoch erwähnt werden, da WP7 einige individuelle Lösungsansätze bietet.

### 4.5.1 Performance

Während auf "normalen" PCs sowohl Rechenleistung als auch Arbeitsspeicher im Überfluss vorhanden ist, müssen sich Entwickler im mobilen Bereich über diese Aspekte wieder Gedanken machen. Nur dann kann eine ausreichende Performance auf dem Gerät zu erzielt werden.

Dieses Ziel lässt sich auf zwei verschiedene Arten erreichen: Entweder schränkt man sich soweit ein, dass die Performance des Gerätes ausreicht oder man beeinflusst die wahrgenommene Performance. Letzteres beschreibt den Effekt, dass einem Anwender eine App die ihn warten lässt, bis alle Daten geladen sind sehr viel langsamer vorkommt, als eine App die ihm wären des Ladevorgangs zumindest schon einzelne Daten anzeigt. Und das obwohl diese meistens insgesamt sogar länger braucht als erstere. Um die wahrgenommene Performance zu verbessern gibt es abhängig von der Situation verschiedene Möglichkeiten:

Wenn eine lange Liste an Daten über eine langsame Handy Verbindung übertragen werden soll bietet es sich an, die Daten in kleinen Blöcken zu laden, und erst dann den nächsten Block zu laden, wenn der Benutzer an das Ende der List gescrollt hat.

Sobald man die lange List im Speicher hat steht man allerdings vor dem Problem, dass die Scroll-Performance stark leidet - besonders bei komplexen Daten Templates. SILVERLIGHT hat für dieses Problem bereits eine eingebaute Lösung - "UI Virtualisierung". Dabei werden die Daten-Container, die aus dem sichtbaren Bereich heraus-gescrollt werden wiederverwendet für die nachfolgenden Daten. Somit wird aufwendiges Zuweisen von neuem Speicher vermieden, und es müssen keine Container für alle Daten-Elemente erzeugt werden.

Falls diese Technik nicht ausreicht um ein flüssiges Scrollen zu ermöglichen, weil die Daten Templates zu komplex sind, gibt es ein auch dafür eine Lösung.

In einem erweiterten ListBox Steuerelement[8] ist es möglich ein alternatives, vereinfachtes Data Template anzugeben, das verwendet wird, während der Benutzer scrollt. So kann z.B. während des Scrollens nur der Name eines Films angezeigt werden und sobald die Liste angehalten wird, auch ein Bild und weitere Informationen.

### 4.5.2 TOMBSTONING

Eine Konsequenz der geringen Ressourcen ist das so genannte TOMBSTONING. WP7 unterstützt kein Multitasking[9], wie man es auf dem Desktop gewöhnt ist. Stattdessen wird ein Programm sobald es inaktiv wird aus dem RAM entfernt. Bevor dies geschieht wird der App zuvor die Möglichkeit gegeben ihren Zustand zu sichern.

Dies geschieht an zwei Stellen:

Zum einen kann jede Page lokale Daten halten. Da Pages bei jedem Aufruf neu erstellt werden, müssen diese Daten, für eine eventuell später ausgeführte "zurück" Navigation, bei jedem Verlassen gespeichert werden. Dies geschieht durch setzen eines STATE Dictionaries, auf das der Navigationsservice Zugriff hat, welcher die Daten dann serialisiert.

Zum Anderen kann die APP Klasse globale Daten halten. Diese können entweder auch mit einer Dictionary basierten Methode gespeichert werden, oder manuell auf dem normalen Dateisystem.

Wenn eine Anwendung oder eine Seite keinen inhärenten Zustand hat, ist die Verwendung dieser Methoden aber nicht verpflichtend. Sie werden dann einfach neu erstellt, wie beim ersten Aufruf.

### 4.5.3 UI Design

Ein weiterer Aspekt, auf den hier aber nicht näher eingegangen werden soll, ist die Bauform. Durch sie bedingt steht sehr viel weniger Bildschirmfläche zur Verfügung, was u.U. ein Komplett unterschiedliches Design nötig machen kann. Außerdem sind in diesem Bereich Eingabemethoden üblich, die auf dem Desktop oft gar nicht zur Verfügung stehen. (Z.B. Multi-Touch-Screens, Beschleunigungs- und Lagesensoren)

[8] Das benötigte Steuerelement ist nicht Teil der Standardbibliothek und kann hier gefunden werden: http://goo.gl/14H2U

[9] Das OS selbst unterstützt Multitasking und ein zukünftiges Update soll es auch Apps zugänglich machen.

## 4.6 Entwicklungsumgebung

Microsoft stellt sowohl Visual Studio 2010 als auch Expression Blend in einer speziellen Version für WP7 kostenlos zur Verfügung. Visual Studio richtet sich dabei an die Entwickler und Blend an die Designer. Aber prinzipiell könnte man beide Tools für die gesamte Entwicklung verwenden.

Über beide Tools könnte man ganze Bücher schreiben, was aber ganz klar den Rahmen dieser Arbeit sprengen würde. Allerdings gibt es einen Punkt den man als Entwickler für WP7 beachten muss:

Der Emulator der im WP7 SDK mitgeliefert wird, verhält sich nicht 100%ig wie ein echtes Gerät. Da der Emulator-Code auf x86 portiert wurde ist die Performance in machen Teilen besser, und in anderen Teilen schlechter. Manches Verhalten lässt sich nur auf einem von beiden reproduzieren und H.264 Videos können nur auf echten Geräten wiedergegeben werden.

Grundsätzlich ist es immer das beste auf dem Gerät zu testen. Nach Möglichkeit auf mehr als nur einem.

# 5 Fazit - Erfolgschancen von WP7

Zum heutigen Tag hat der WP7 Marketplace über 17.000 Apps und mehr als 36.000 Personen haben sich für das Entwicklerprogramm registriert.[6] Seid einem Monat rollt Microsoft das erste große Update für WP7 aus. Dieses enthält Kopieren&Einfügen als auch viele Performance- und Stabilitäts-Verbesserungen. Außerdem wurde das "Mango" Update vorgestellt, dass im Herbst für alle WP7 Geräte kommen soll und endgültig mit Android und iOS in Sachen Funktionsumfang gleichziehen soll.

Ein weiterer Faktor der die Zukunft von WP7 mit Sicherheit entscheidend beeinflussen wird, ist die Allianz mit Nokia, die diesen Monat endgültig beschlossen wurde.

Auch wenn Microsoft bis heute keine offiziellen Zahlen vorgelegt hat, wie viele *Geräte* tatsächlich verkauft wurden, sind diese Fakten ein starker Beleg dafür, dass WP7 nicht so bald wieder vom Markt verschwinden wird.

Die größte Schwäche von WP7 ist im Moment noch, dass im Gegensatz zu Windows Mobile 6.5 (heute "Windows Phone Classic") es noch nicht für Geschäftskunden tauglich ist. Da der Business-Sektor traditionell Microsofts stärkster Geschäftsbereich ist, muss dieses Manko so schnell wie möglich beseitigt werden. Dann steht dem Erfolg von *Windows Phone 7* nichts mehr im Weg.

# Literatur


[1] F. Rapp, “Entwicklung der Open-Street-Applikation für Windows Phone 7,” *Universität Ulm*, 2010.

[2] W. Rolke, Abrufdatum: 02.05.2011. [Online]. Available: http://www.wolfgang-rolke.de/wince/platform.htm

[3] S. Guthrie, “Silverlight,” Abrufdatum: 02.05.2011. [Online]. Available: http://weblogs.asp.net/scottgu/archive/2007/05/07/silverlight.aspx

[4] M. de Icaza, “Moonlight 4 preview 1 is out,” Abrufdatum: 02.05.2011. [Online]. Available: http://tirania.org/blog/archive/2011/Feb-16.html

[5] D. Beck, “WPF & Silverlight,” *Universität Stuttgart*, 2008.

[6] W. P. Applist, “Wp7 app statistics.” [Online]. Available: http://www.windowsphoneapplist.com/stats/