Inhalt

Abbildungsverzeichnis

Listing Verzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADO.NET	Active X Data Objects
ASP.NET	Active Server Pages .NET
COM	Component Objekt Modell
CSS	Cascading Style Sheets
Exif	Exchangeable Image File Format
GPS	Global Positioning System
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
METRO	Kunstname
MVVM	Model View ViewModel
NFC	Near Field Communication
OLED	Organic Light Emitting Diode
PDF	Portable Document Format
QR-Code	Quick Response Code
RAD	Rapid Application Development
REST	Representational State Transfer
RPC	Remote Procedure Call
RTF	Richt Text Format
SDK	Software Development Kit
SMS	Short Message Service
SOAP	Simple Object Access Protocol
TIFF	Tagged Image File Format

UI	User Interface
UML	Unified Modeling Language
WAV	Kurzform WAVEFORM
WCF	Windows Communication Service
WNS	Windows Notification Services
XAML	Extensible Application Markup Language
XML	Extended Markup Language
XP	Extreme Programming

1 Aufgabenstellung

1.1 Ausgangssituation

Die ppedv AG[1] beschäftigt sich mit Wissensvermittlung im Microsoft-Technologie-Umfeld im deutschsprachigen Raum. So schult und coacht ppedv pro Jahr mehrere Tausend Teilnehmer. Besonders im Bereich Software-Entwicklung genießt, ppedv einen Vorreiterruf. Zu den besonderen Aufgaben im Rahmen der Wissensvermittlung gehört, sehr frühzeitig, neue Technologien analysiert zu haben und bewerten zu können. Im Rahmen der langjährigen Tätigkeit wird ppedv von Microsoft häufig in frühen Beta Phasen von neuen Produkten eingebunden, so auch bei Windows 8.

Der Autor hat rund 25 Jahre Erfahrung in der Software Branche und sich zuletzt vor allem mit Web-Technologien und Silverlight beschäftigt. Zu ASP und ASP.NET stammen drei Bücher[2] aus der Feder des Verfassers.

1.2 Motivation

Heutige Unternehmensanwendungen adressieren den klassischen Desktop PC und werden seit Windows 3.11 mehr oder weniger mit ähnlichen Konzepten, häufig mehrschichtig mit Präsentations-, Logik- und Datenschicht, entworfen und realisiert.[3]

Der Autor vertritt die Annahme, das sich in Zukunft die Mehrheit der Entwickler umstellen und nach neuen Vorgehensmodellen arbeiten muss. Microsoft als absoluter marktdominierender Faktor, bringt sein neues Betriebssystem Windows 8. Dies ist aktuell nur in einer Beta Version vorhanden und weder in Design noch API komplett. Es gibt kaum Musteranwendungen und mangelnde bis falsche Dokumentation. Gerade dies stellt für den Autor den besonderen Reiz dar, in einer frühen Phase zu untersuchen, wie sich diese Veränderungen in allen Software-Projektphasen auswirken können. Diese Arbeit soll die Belege dafür liefern.

Der Leser dieser Arbeit partizipiert an den Erkenntnissen und kann so letztendlich künftigen Projektaufwand reduzieren.

---

[1] Website www.ppedv.de

[2] Preishuber, Hannes (2001) ASP 3.0 / ASP+, Sybex
Preishuber, Hannes (2006) ASP.NET 2.0 Crashkurs, Beta 2 Edition, Microsoft Press
Preishuber, Hannes (2006) ASP.NET 2.0 Crashkurs, Microsoft Press

[3] Siehe auch ISO/IEC 42010

1.3 Vorgehensweise

Zunächst wird die aktuelle Situation im Marktsegment Mobile Endgeräte analysiert. Dabei wird der Fokus auf Gerätetypen und technische Fähigkeiten gelegt. Daraus resultieren typische und neuartige Anwendungsfälle.

Zudem wird ein Schwerpunkt auf die Benutzerschnittstellen gelegt, die aus Touch-Bedienung resultieren. Speziell erörtert wird die Frage wie im Entwurf darauf Rücksicht zu nehmen ist.

Am konkreten Anwendungsfall (Fax senden und empfangen) wird spezifiziert, wie diese technischen Rahmenbedingungen und die Anforderungen kombiniert werden können.

Daraus folgt, dass eine eingehende Untersuchung der Zielplattform Windows 8 nötig ist. Im Bottom-Up-Prinzip werden zunächst streng abgegrenzte Teilprobleme gelöst und damit die Machbarkeit evaluiert.

Anhand dieser Ergebnisse wird letztendlich ein Prototyp implementiert und die Erkenntnisse daraus am Ende zusammengefasst.

2 Einsatz von Smartphones und Probleme

Das Internet hat sich sowohl im privaten als auch geschäftlichen Bereich etabliert. Was mit der PC-Ära begann, verlagert sich zunehmend auf mobile Geräte. Der Formfaktor Smartphone wird immer wichtiger.

Ende 2011 gab es in Deutschland 21,3 Millionen Smartphones. Damit liegt Deutschland hinter seinen europäischen Nachbarn zurück. In Frankreich besitzen 40 Prozent der Einwohner ein Smartphone, in Italien sind es sogar 44 Prozent.[4]

Die unterschiedlichen Formfaktoren wie Tablet PC oder Smartphone, Betriebssysteme und Netzwerk-Infrastrukturen erfordern ein Umdenken in der Entwicklung der nötigen Software. Sowohl die Art der Anwendung als auch die Vertriebswege verändern sich. Ein Paradigmenwechsel für die gesamte damit beschäftigte Softwarebranche.

2.1 Geräte und Plattformen

Zunächst stellt sich die Frage nach der möglichen Plattform. Aus wirtschaftlichen Gründen ist eine Marktbetrachtung wesentliche Entscheidungsgrundlage für Investitionen in Software-Projekte. Da der Markt sehr volatil ist, sind die aktuellen Verkaufszahlen zwar interessant, aber nur bedingt hilfreich. Wir befinden uns in einer extremen Wachstumsphase, deren Ausgang offen ist.

Tabelle 1 Hersteller, Geräte und Betriebssysteme

Hersteller	Geräteklasse	Plattform
Apple	iPhone, IPad	iOS
Google, Samsung	Smartphone, Tablet	Android
Microsoft	Windows Phone	WP7.5, Windows 8
Microsoft	Windows Tablet	Windows 8
Research In Motion (RIM)	Blackberry	Blackberry OS
Nokia, Samsung	Smartphone	Symbion OS
Samsung	Smartphone	Bada

[4] http://www.comscore.com/Press_Events/Presentations_Whitepapers/2012/2012_Mobile_Future_in_Focus

So kann bestimmendes Merkmal die Art der Anwendung auf der zu adressierenden Plattform sein. Ein E-Book macht am meisten Sinn für den Amazon Kindle Reader, der auch Funktionen wie die G3 Connectivity eines Smartphones umfasst.

Um alle Plattformen unterstützen zu können, ist ein erheblicher Aufwand zu betreiben, der wirtschaftlich nur bei entsprechender Reichweite sinnvoll zu rechtfertigen ist. Die Telefonie und Chatsoftware Skype ist praktisch auf allen Geräten verfügbar, außer wenn dies wegen ihres Formfaktors dafür nicht geeignet ist. Die Hunderte Millionen von Benutzern rechtfertigen dies.

Die stärksten Zuwächse verzeichnet die Android-Plattform[5]. Das profitabelste Unternehmen ist Apple[6]. Die größte Basis hat Microsoft mit fast einer Milliarde installierten Windows PCs[7].

2.1.1 Tablet-Computer

Mit dem iPad hat Apple 2010 eine neue Geräteklasse zwischen Handy und Notebook definiert. Wobei andere Mitbewerber, wie Microsoft, schon jahrelang (seit wann?) Lösungen in diesem Bereich anbieten, allerdings bisher mit wenig Erfolg. Die Verbreitung des iPad hat aktuell hohe Zuwachsraten und verdrängt damit die gerade erst entdeckten Netbooks. Allerdings ist mit Windows 8 ein neues Betriebssystem am Start, das universell PC und Tablet adressiert. INTEL hat dazu passend eine neue Geräteklasse definiert, das Ultrabook[8]. Damit sollen sich die Vorteile von Notebook und Tablet vereinen. Schlankes Design, geringes Gewicht und 9 Stunden Akkulaufzeit könnten für den Konsumenten durchaus interessant sein. Darüber hinaus wird für Windows 8 eine neue Tablet-Geräteklasse erscheinen, die auf der ARM-Architektur basiert dessen Name Windows RT lautet[9].

Microsoft hat für den 26.Oktober 2012 eine eigene Hardware-Linie mit dem Namen Surface angekündigt, die ein Hybrid zwischen Tablet und Notebook sein soll. Als Hardware-Plattform werden sowohl x64 als auch ARM zum Einsatz kommen. Am selben Tag wird Windows 8 verfügbar sein.

---

[5] http://derstandard.at/1277339469037/Wettlauf-Android-zieht-dank-rasantem-Wachstum-am-iPhone-vorbei

[6] http://www.focus.de/finanzen/news/gewinn-um-93-prozent-gesteigert-apple-verkauft-mehr-als-35-millionen-iphones-und-viele-ipads_aid_742482.html

[7] Gartner http://www.gartner.com/it/page.jsp?id=1762614

[8] http://www.intel.com/content/www/us/en/sponsors-of-tomorrow/ultrabook.html

[9] http://windowsteamblog.com/windows/b/bloggingwindows/archive/2012/04/16/announcing-the-windows-8-editions.aspx

2.1.2 Gerätespezifikationen

Allen Geräten, egal ob Smartphone oder Tablet, ist gemein, dass sie auch mit Batterie betrieben werden. Der Benutzer möchte möglichst lange mit dem Gerät arbeiten können. Daraus folgt, dass mit geringerer Prozessorleistung gearbeitet werden muss. Die Wärmeabfuhr erfolgt in der Regel passiv und damit ohne Lüfter. Typischer Arbeitsspeicher reicht von 256 MB bis 1 GB. Da dieser ein wesentlicher Kostenfaktor ist, geht der Trend zu geringerem RAM. Anstatt von Festplatten werden interne SSD RAMs eingesetzt, die etwas günstiger und langsamer sind als DRAM. Aktuell sind z.B. 8 GB eine gängige Größe. Die Hersteller wollen durch Einbindung von Cloud-Diensten den Festplattenspeicher sozusagen erweitern. Daraus folgt die Notwendigkeit einer möglichst dauerhaften Netzwerkanbindung basierend auf Diensten wie G3/G4 oder WiFi.

Da ein Großteil der Smartphone Devices über die sogenannten Netzbetreiber (auch Carrier) vertrieben wird, unterliegen die Geräte teilweise technisch unnötigen Beschränkungen. Auch die Aktualisierung der Firmware erfolgt großteils über die Netzbetreiber. Funktionen wie Tethering[10] sind so für das idente Gerät und Plattform mal vorhanden und mal nicht.

2.2 Eingabe/Ausgabe

Eine Eigenschaft von Smartphones sind die erweiterten Möglichkeiten der Ein- und Ausgabe von Informationen. Waren seit Einführung der grafischen Benutzeroberflächen Tastatur und Maus die primären Eingabeschnittstellen, kommen nun Touch, Stift (Ink), Sprache und Kamera hinzu. Dazu stehen Technologien, wie Near Field Communication in den Startlöchern, mit denen ebenfalls Eingaben in Software durch den Benutzer initiiert werden können. Auch die Kamera kann als Eingabemedium genutzt werden, um z.B. per QR Codes auf eine Website zu navigieren. Sehr spezielle Anwendungsfälle nutzen die eingebaute Sensorik, wie GPS um Fotos mit Lokalisierungsinformation zu ergänzen. Mit dem Helligkeitssensor kann der Programmierer die Anwendung auf unterschiedliche Lichtverhältnisse anpassen. Sehr universell einsetzbar sind magnetische Sensoren oder Gyrometer mit dem z.B. das Drehen der Bildschirmansicht ausgelöst wird. Für den klassischen Windows-Entwickler ist es durchaus ungewohnt, für mindestens zwei verschiedene Bildschirmdimensionen entwickeln zu müssen. Im Gegensatz zum klassischen 4:3 Format, kann dies den Aufwand in der Entwurfsphase der Benutzerschnittstelle deutlich erhöhen.

2.2.1 Bildschirmmerkmale

Wesentliches Merkmal der neuen Gerätegenerationen ist der große, meist gerätefüllende Bildschirm. Die Vorderseite dient fast ausschließlich als Visualisierungsinstrument. Die physikalischen Schalter wandern meist auf die oft wenige Millimeter schmalen Seiten der Geräte.

---

[10] Tethering bezeichnet die Anbindung eines Computers an das Internet mittels Smartphone. Das Mobiltelefon übernimmt damit die Rolle eines Modems.

Die fehlenden Schalter werden virtuell (als Buttons) auf dem Bildschirm abgebildet. Da dem Benutzer dadurch die haptische Wahrnehmung[11] fehlt, muss das Programm nahezu verzögerungsfrei reagieren.

Die Bildschirme zeichnen sich durch hohe Auflösung und meistspiegelnde Displays aus. Videos wirken so brillanter. Das adressiert vor allem den Endkunden. Im Außeneinsatz erweisen sich spiegelnde Displays, speziell unter direktem Sonnenlicht als ungeeignet. Software-Entwickler können dies in der Planung berücksichtigen indem je nach Lichtverhältnissen die Anwendung alternative Darstellungen wählt. Größere Schrift, heller statt schwarzer Hintergrund und natürlich maximierte Hintergrundbeleuchtung können die Lesbarkeit erhöhen. Neuere Displays z.B. auf OLED[12]-Basis verzichten auf Hintergrundbeleuchtung und benötigen dann mehr Energie für helle Flächen.

Ein großes Problem stellen die verschiedenen Bildschirmauflösungen und Formate dar. Zwar kann durch den Einsatz von vektororientieren UI-Beschreibungen die Oberfläche skaliert werden, jedoch entstehen bei Pixelgrafiken oder geändertem Bildformat Design-Probleme. Apple löst dies recht einfach mit einem fixen Format und Größe. Software, die für das iPhone geschrieben wurde, kann so einfach auf dem iPad laufen, dessen Bildschirm genau doppelt so groß ist. Das Konzept hat aber auch seine Grenzen, wie das neue Retina[13]-Display beim iPad 3 zeigt, das sehr hochauflösend darstellt.

2.2.2 Touch-Bedienung

Wesentliches Merkmal der neuen Smartphone-Generation ist die Touch-Bedienung. Smartphones werden in der Regel mit dem Daumen bedient. Tablets werden dagegen häufig auch mit beiden Händen gleichzeitig bedient.

Ein wesentlicher Nachteil von touch-orientierten Systemen besteht darin, dass die Hand Teile der Oberfläche verdeckt. Typische Anwendungsfälle wie Context Menü funktionieren deshalb auch bei Touch-Oberflächen nur bedingt. Eine Lösung kann die Bedienung mit Stift sein. Das hat mehrere Vorteile. Man kann Skizzen oder Freitext erstellen. Darüber hinaus ist der Stift in jeder Situation, also auch mit Handschuhen, ein funktionierendes Bedienelement. Microsoft unterstützt in all seinen Windows Tablet-Editionen die Eingabe per Stift. Als Displays müssen dann sogenannte Dual Digitizer Displays verbaut werden, die sowohl Stift als auch Touch erkennen. Windows erkennt automatisch, wenn der Benutzer den Stift verwendet und schaltet daraufhin die Touch-Erkennung aus, um beispielsweise Probleme mit aufgelegtem Handballen zu vermeiden.

---

[11] Vgl. Dorau (2011 S. 214)

[12] organische Leuchtdiode - organic light emitting diode, OLED

[13] Retina-Display ist eine Handelsbezeichnung von Apple. Diese haben eine so hohe Pixeldichte haben, dass das menschliche Auge einzelne Pixel nicht erkennt.

Um Texteingaben zu unterstützen, bieten Smart Devices virtuelle On-Screen-Tastaturen. Für den Software-Entwickler stellen die gängigen Hersteller datenspezifische Keyboards bereit. So wird beim Eingeben einer E-Mail-Adresse das @ auf der Tastatur angezeigt. Ein Umstand, den man in der Entwicklung explizit berücksichtigen muss, um dem Benutzer das beste Nutzererlebnis zu ermöglichen.

2.3 Netzwerk und Roaming-Anforderungen

Smartphones werden im Hinblick auf kontinuierliche Konnektivität konzipiert. Eine Navigation per Google Maps ist nur mit Internetverbindung möglich. Technisch wäre es nicht unbedingt nötig, aber die Geschäftsmodelle der beteiligten Firmen bauen darauf auf.

Anders als bei PC Software sind die Qualität und Kosten der Verbindung starken Schwankungen unterworfen. Der Entwickler sollte berücksichtigen, ob er sich im WLAN oder G3/G4-Status befindet. Darüber hinaus ist es relevant, welche Verrechnungsmodelle die gewählte Netzwerkverbindung nach sich zieht. Der Worst Case ist ein Software Update im Roaming-Betrieb. Dies gilt es zu verhindern.

Die Integrität der Daten muss zu jeder Zeit gewährleistet sein, auch wenn die Verbindung plötzlich abbricht. Dies stellt für ein bewegtes Gerät in nahezu jeder Nutzungssituation ein Problem dar.

Eine Anwendung sollte auch nutzbar sein, wenn keine Netzwerkverbindung zur Verfügung steht. Bei bestehender Verbindung kann die Anwendung die fehlenden Daten holen oder Updates durchführen. Unterbrochene, speziell längere Aktivitäten, wie der Download eines Videos, sollten wieder aufgenommen werden können.

2.4 Eco System

Der Smartphone Markt wird geprägt von wirtschaftlichen Interessen. Apple hat bewiesen, dass man mit dem Follow-Up-Geschäft deutlich mehr Umsatz und vor allem Gewinn erzielen kann, als mit dem Verkauf der Geräte. Folglich versuchen die Mitbewerber nun dieses Segment zu erschließen. Das hohe Wachstum und dauerhafte Folgegeschäft scheinen jeden Aufwand dafür zu rechtfertigen. Neben den zahlungspflichtigen Cloud-Diensten, sind vor allem die Apps die Umsatzbringer. Bei Apple gehen 30% der Apps und Musikverkäufe direkt an Apple - bei sehr geringem Aufwand und 30 Milliarden Downloads[14]. Entsprechend bauen nun Anbieter die Store[15]-Idee nach.

Unter Umgehung klassischer Vertriebswege (Händler) und Technologien (DVD) lassen sich auf einem Windows Phone Gerät nur Anwendungen aus dem Store installieren. Bei Windows 8 Metro Style Apps hat Microsoft auf Proteste aus der Industrie reagiert und

[14] http://www.netzwelt.de/news/92608-app-store-apple-vermeldet-30-milliarden-downloads.html

[15] http://www.windowsphone.com/de-DE/marketplace

angekündigt auch einen direkten Rollout per Active Directory, genannt Side-Loading, anzubieten.

Die Regeln, um in einen Store aufgenommen zu werden, sind sehr strikt. Tricks mit undokumentierten Schnittstellen des Betriebssystems oder sexuellen Inhalten werden durch Ausschluss bestraft.

Ob dieses App-Konzept im Angesicht der beinahe diktatorischen Regeln, der mangelnden Flexibilität und des aufkommenden HTMLs dauerhaft so bestehen wird, mag bezweifelt werden.

2.5 Entwicklungswerkzeuge

Wesentliches Entscheidungskriterium war bisher die Produktivität einer der teuersten Ressourcen, dem Entwickler. Allgemein gilt, systemnahe Programmiersprachen bedeuten erhöhten Aufwand. Microsoft bietet für Windows 8 drei sehr unterschiedliche Zweige, um letztendlich dasselbe Ergebnis zu erzielen. Das sind jeweils die Paarungen XAML und .NET, HTML 5[16] und JavaScript, XAML und C++.

Microsoft möchte mit dem HTML 5-Zweig neue Entwicklerschichten ansprechen. Für diese Arbeit hat sich der Autor für XAML und VB.NET entschieden, da dort der beste Kenntnisstand vorhanden ist. Das Grundthema zeigt ohnehin enorm viel Komplexität, so dass eine neue Programmiersprache das Scheitern des eigentlichen Software-Projektes wahrscheinlicher macht.

2.6 Probleme

Um eine fundierte Plattformentscheidung zu treffen, sind die Entwicklungszyklen aktuell viel zu kurz. Die Geräte werden dank fest eingebauter Akkus quasi auf einen zweijährigen Nutzungszeitraum hin entwickelt. Daraus folgt ein höheres Risiko, dass die getroffene Entscheidungen in naher Zukunft falsch erscheint.

Nicht umsonst hat sich der Begriff App etabliert. Die Anwendungen umfassen in der Regel wenige hunderte Codezeilen und entstehen in wenigen Tagen bis Wochen. Der Kaufpreis liegt selten über zehn Euro. Daraus folgt, dass Software für Smartphones gänzlich anders entsteht, als man dies bei typischen Geschäftsanwendungen kennt. Es gibt keine umfangreichen Planungen, wie im Wasserfallmodell definiert. Die Entwickler haben meist eine andere Historie, Erfahrungswerte und grundsätzliche Ideale.

Der erreichbare Technologievorsprung und die Investitionssicherheit von Unternehmen gegenüber Mitbewerbern, wird immer geringer.

[16] Standard der W3 Org http://www.w3.org/

Gerade im Android-Umfeld zeigen sich erhebliche Probleme in der Fragmentierung des Gerätemarktes, die mehrere Varianten der Software für die gleiche Plattform nötig machen.

3 Analyse technischer Rahmenbedingungen

Für diese Diplomarbeit wird als Plattform das kommende Betriebssystem Windows 8 aus dem Hause Microsoft eingesetzt. Zum Zeitpunkt der Erstellung dieser Arbeit liegt Windows 8 in einer Beta-Version, der Windows 8 Release Preview vor. Als Entwicklungsumgebung wird Visual Studio 2012, ebenfalls in der Beta, eingesetzt.

Zum Zeitpunkt des Drucks gibt es keine gedruckte Literatur und nur mangelhafte Dokumentation[17]. Darüber hinaus werden sich Details in Funktion und API bis zum Erscheinungszeitpunkt noch verändern können.

Der Erfahrungsschatz mit Anwendungen für touch-optimiertes System ist ganz generell noch gering. Es fehlen die üblichen „best practices“[18], auf die man im Entwurfs- und Entwicklungsprozess zurückgreifen könnte.

3.1 Hardware-Spezifikation

Zum jetzigen Zeitpunkt ist es praktisch unmöglich eine Entscheidung für eine gewisse Geräteklasse zu treffen. Bildschirmgrößen, -auflösungen und -formate sind in nahezu jeder Variante vorhanden. Für das Projekt wurde der Samsung Slate PC Serie 7 in einer speziellen Developer-Variante gewählt. Dieser ist ein Prototyp, der das komplette Spektrum an möglichen Funktionen in seiner Hardware implementiert hat. Es handelt sich dabei um einen Tablet Computer, den man per Docking Station und Bluetooth-Tastatur in einen Notebook-ähnlichen Zustand versetzen kann.

Abbildung 1 Samsung Slate

Mit der Hardware Ausstattung von 4 GB RAM in Kombination mit einem Intel Core I5 Pro-

[17] http://msdn.microsoft.com/en-us/library/windows/apps/

[18] „Der Begriff best practice, auch Erfolgsmethode genannt, stammt aus der angloamerikanischen Betriebswirtschaftslehre und bezeichnet bewährte, optimale bzw. vorbildliche Methoden, Praktiken oder Vorgehensweisen im Unternehmen“ [Wikipedia]

zessor und einer SSD ist damit auch Software-Entwicklung möglich. Darunter leidet die Akkulaufzeit, die drei Stunden nicht übersteigt.

Wesentlich für die Entwicklung ist der Bildschirm mit einer Auflösung von 1366x768 Pixel was genau der minimalen Auflösung von Windows 8 entspricht um die volle Funktion der Metro Shell nutzen zu können.

3.2 User Interface

Die falltypische Verwendung des Gerätes bestimmt das Design der Benutzerschnittstelle. Videos betrachtet man auf dem Sofa und hält damit das Gerät mit beiden Händen, weil es mit einer Hand zu schwer wird.

Software entwickelt man unter Nutzung einer externen Tastatur in der Docking Station. Für das Schreiben einer E-Mail hält man das Gerät mit der linken Hand am Rand und tippt mit rechts. Betrachtungen des Bedienerlebnisses[19] sind für den Entwickler Teil des Paradigmenwechsels.

Wenn man von einer beidhändigen Bedienung ausgeht, müssen Aktionsschaltfläche, welche die klassischen Menüs in Sachen Funktion ersetzen, im Aktionsradius der beiden Daumen liegen. Trotzdem muss die Bedienung auch mit Maus und Tastatur funktionieren, was z.B. bei iPad-Anwendungen nicht der Fall ist.

3.3 Sensorik

Das Samsung Slate Device hat umfangreiche Sensoren für geografische und physikalische Position, Bewegung oder Lichtverhältnisse. Darüber hinaus kann man auch Mikrophone oder Kamera als Sensor betrachten und Einsetzen. Der Samsung Slate hat je eine Kamera mit unterschiedlicher Auflösung auf der Vorder- und Rückseite verbaut.

3.4 Service-Schicht

Basis sind die Dienste des Visendo Fax Servers[20]. Dies ist ein Produkt aus dem Hause ppedv. Es gibt Lizenzen von wenigen Benutzern bis zu Tausenden gleichzeitig. Der Größte Kunde hat circa 5000 Arbeitsplätze und 32 ISDN Kanäle im Hintergrund. Entwickelt wurde der Fax Server mit C++. Ein direkter Zugriff ist aktuell mit einer Socket API implementiert. Darüber hinaus existiert ein WCF Service Interface.

[19] Vgl Dorau (2011, S 20)

[20] http://www.visendo.com/de/default.aspx

4 Anforderungen aus Nutzersicht an Fax Client

In dieser Untersuchung soll eine Fax Client Software mit üblichen Funktionen, wie Senden, Empfangen und Sichten, nach völlig neuen Gesichtspunkten erstellt werden. Dabei wird das Visendo Team zu Rate gezogen.

Bisher existiert ein Windows Druckertreiber der per Socket Connection mit einem Dienst kommuniziert. Die eigentliche Logik und Faxkommunikation geschieht in der Service-Schicht. Dort wird auch die Steuerung von Hardware wie ISDN oder Fax Modem realisiert. Dies soll sich auch nicht ändern. Eingehende Faxe werden per E-Mail zugestellt.

Eine Portierung der Windows-Anwendung auf ein touch-orientiertes System wird kein optimales Ergebnis für den Benutzer erzielen. Es soll erforscht werden, wie eine typische Geschäftsanwendung auf Tablet Devices aussehen und funktionieren könnte. Wesentlicher Bestandteil einer derartigen Anwendung ist der Verzicht. Die Reduktion der Benutzerschnittstelle auf das aktuell unbedingt Nötige.

Obwohl diese Arbeit die Entwicklung für das mobile Endgerät vorsieht, muss auch die Service-Schicht in die Betrachtung mit einbezogen werden. So gibt es heute üblicherweise eine strenge Verbindung von Server und Client. Beleuchtet wird die Möglichkeit, diese Bindung aufzubrechen und durch eingebettete Metadaten das Faxobjekt nur für sich universell nutzbar zu machen. Aus Benutzersicht ist ein derartiges Objekt ganz simpel als Datei nutzbar, die ähnlich wie ein Stück Papier, kopiert oder verschoben werden kann.

4.1 Konfiguration

Die Einstellungen, wie Server-Namen, Benutzerdaten, Faxnummer und ähnliches sollen mit minimalem Aufwand erfolgen. Am einfachsten ist es, wenn der Benutzer nur seine E-Mail-Adresse eingeben muss und sich daraus die weiteren Einstellungen automatisch ergeben.

4.2 Fax senden

Das Versenden eines Faxes soll mit minimalen Klicks erfolgen. Faxe können auch aus mehreren Seiten bestehen. Fraglich bleibt, ob ein Mixversand aus PDF- und Word-Dokumenten in einem Vorgang unterstützt werden soll. Letztendlich wird der Aufwand beim Erstellen des Prototyps dies zeigen.

Als Quelle kommt die eigene Webcam in Betracht. Der Beleg liegt am Tisch, der Benutzer fotografiert ihn.

Außerdem können existierende Dokumente, die beispielsweise per E-Mail empfangen worden sind, versendet werden. Mit passender Grafiksoftware von Drittanbietern, kann auch das Dokument per Stift oder Finger unterschrieben werden, bevor es gefaxt wird.

Wunschformate sind sicherlich PDF und Word-Dokumente. Darüber hinaus JPG, PNG, TIFF. Die weiteren Forschungsergebnisse werden zeigen, welche Formate umgesetzt werden können.

Es soll vermieden werden, eine zusätzliche Empfängerverwaltung einzubauen. In der Regel haben mobile Geräte Adressbücher, die genutzt werden sollen. Gewünschte Funktionen: Suche, Editieren, Neu.

Geprüft werden soll auch die Möglichkeit aus der Adressverwaltung die passenden Faxe zu sichten.

4.3 Fax empfangen

Ein übliches Faxgerät steht in einem Büro, selten direkt an einem Arbeitsplatz. Ab und zu schaut jemand, ob ein neues Fax angekommen ist. Meist wird der Empfang auch mit akustischen Signalen wie einem Beep unterlegt. Die Erwartungshaltung des Benutzers für eine Fax Client Software entspringt diesem Nutzungsmuster.

Auf einem mobilen Gerät ist es üblich, wie bei SMS oder Anruf, über dem Programmsymbol die Anzahl der neuen Eingänge als Ziffer anzuzeigen. Wenn das Gerät im Blickfeld ist, kann auch eine Pop-Up-Dialogbox gewünscht sein oder ein Tonsignal. Dies muss wählbar sein.

Die empfangenen Faxe sollen als Übersichtsliste dargestellt werden. Funktionen: Suchen, Sortieren, Löschen und Anzeigen.

4.4 Funktionsreduktion

Denkbar ist, dass Faxdokumente am Server auf Textinhalte per Texterkennung gescannt werden, so dass auch eine indizierte Suche möglich ist. Auch denkbar und technisch möglich ist eine automatische Übersetzung der Texte. Dies wird für die aktuelle Implementierung explizit ausgeschlossen.

5 Anwendungsfall Fax Client

Im folgenden Abschnitt wird der Fax Client beschrieben. Die verschiedenen Anwendungsfälle werden mit Hilfe von UML-Diagrammen[21] dokumentiert.

Es erweist sich als einigermaßen schwierig, die Anwendungsfälle mit UML zu beschreiben[22] und dabei auf Windows 8 Rücksicht zu nehmen. Es gibt einige Windows-spezifische Eigenschaften, die im Use Case abgebildet werden sollten. Die Service-Kommunikation geschieht Asynchron, was sich mit UML so nicht abbilden lässt. In speziellen Fällen kann man sogar Windows 8 als Akteur betrachten, wenn zum Beispielen anderes Programm und damit ein anderes System wie ein Benutzer agiert.

Im W3C wird mit dem Entwurf des MBUI[23] Standard diese Problematik adressiert.

[21] UML - Unified Modeling Language,ist eine graphische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von Software und Systemen. UML ist ein ISO Standard.

[22] Vgl. Grässle, Baumann (2000 S. 66

[23] MBUI Model Based UI http://www.w3.org/TR/task-models/

5.1 Use Case per UML beschreiben

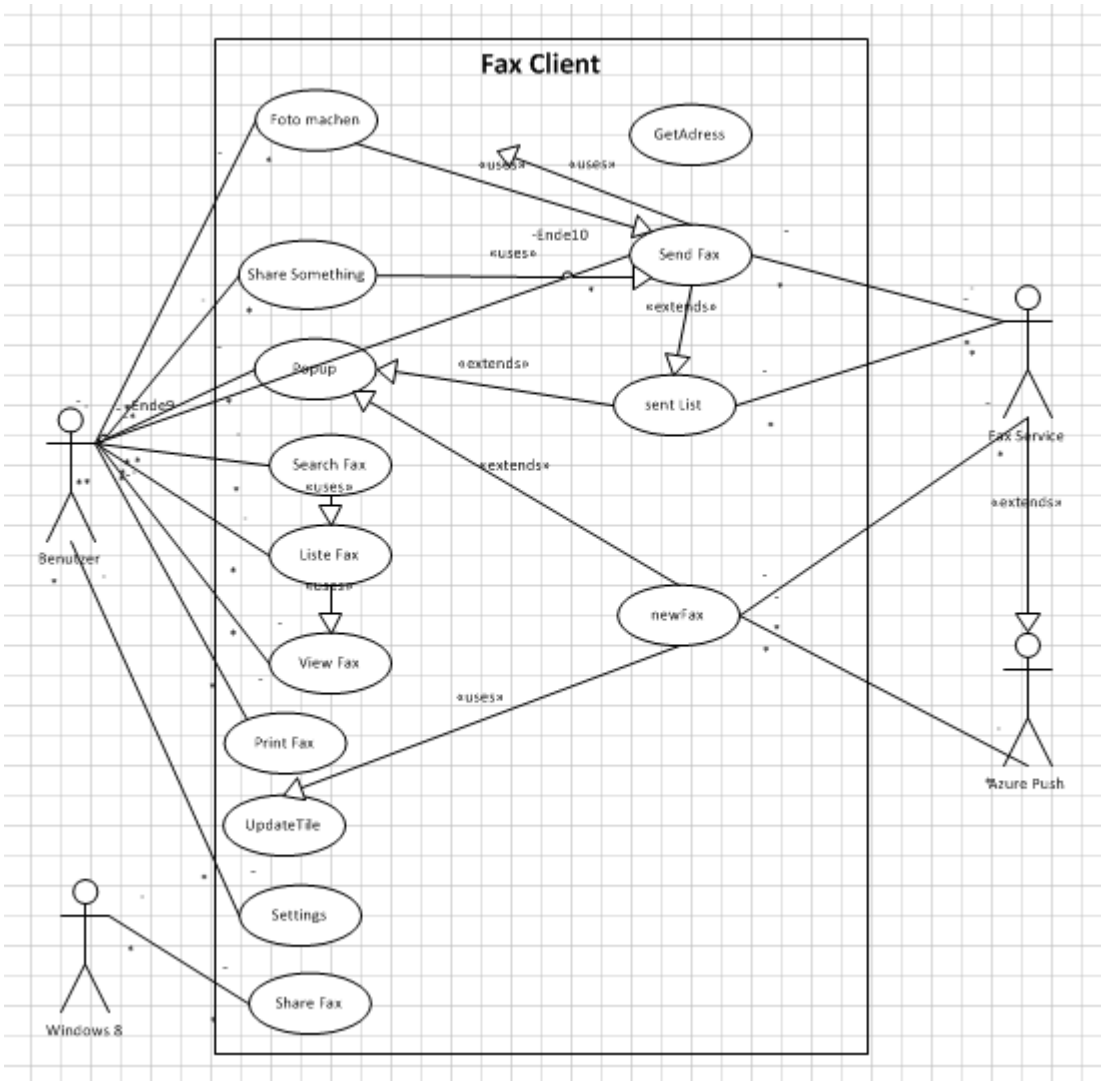


Abbildung 2 UML Use Case Diagramm

5.2 Benutzerdefinition

Der Benutzer muss viele Aktivitäten ausführen können ohne dabei eine gewohnte Menüstruktur verwenden zu müssen. Im Kern empfängt und versendet er Faxe. Das Besondere ist aber die Interaktion mit bestehenden Anwendungen über Standard-schnittstellen, so dass wesentliche Funktionen ausgelagert werden. So wird eben explizit keine eigene Adressdatenbank gepflegt.

5.3 Windows 8 als Nutzer

Im Diagramm wird Windows 8 als eigener Benutzer angeführt. Dies dient dazu, um Funktionen, die eigentlich typischerweise vom Anwender, aber in diesem Fall autark vom Betriebssystem gestartet werden, zu definieren. Als Beispiel kann man eine Task nehmen, die alle 15 Minuten einen Service aufruft. Bei Windows 8 ist es möglich das andere Metro-Programme den Fax Client als Datenquelle nutzen. Dies ist

konzeptionell recht neu, weil dadurch auch der Start des Programms initiiert werden kann.

5.4 Service-Schicht

Der vorhandene Fax Server exponiert einen WCF Service [24]. Dieser verwendet zwar SOAP[25] als Protokollstandard, dieser ist aber aus verschiedenen Gründen im mobilen Umfeld wenig geeignet.

Moderne Anwendungen verwenden REST[26] Services mit JSON[27] serialisierten Daten.

Die Gründe liegen im kompakten Datenformat, das aus den verschiedensten Programmiersprachen einfach und mit geringem Overhead an Speicher und Prozessorbedarf de- und serialisiert werden kann. Die Verwendung von REST über HTTP garantiert eine einfache Kommunikation, die auch in den Firewalls keine expliziten Regeln benötigt.

Als Programmier-Framework für die Service-Schicht kommt ASP.NET Web API zum Einsatz. Die Kodierung kann mit .NET erfolgen. Das Hosting kann, muss aber nicht, im Web Server IIS stattfinden. Der Vorteil besteht darin, dass es keinen Technologiebruch gibt und so die Gesamtanwendung einfach zu entwickeln und zu pflegen ist. Trotzdem steht die REST Service-Schicht jeder Art von möglichen zukünftigen Clients offen. Ein durchaus wahrscheinlicher Anwendungsfall ist ein HTML5 Web Interface mit JavaScript.

Ein zusätzlicher Nutzen der REST Query-Syntax in Zusammenarbeit mit der ASP.NET Web API ist, dass auch partielle Abfragen abgesetzt werden können, die weniger Datenverkehr verursachen. In der Regel ist ein WCF Service Call in der Form *HoleKunden* angelegt. Mit REST kann dies über Querystring beliebig dynamisch erweitert werden, wie *HoleKunden, die aus Burghausen sind*. Ein WCF Call erwartet einen spezifischen Service Contract[28], in der Praxis die Funktionsparameter, die nicht ohne weiteres erweiterbar sind. REST ist da wesentlich flexibler.

---

[24] WCF Windows Communication Foundation ist eine serviceorientierte Kommunikationsplattform für verteilte Anwendungen in Microsoft Windows auf der .NET-Plattform.

[25] SOAP Simple Object Access Protocol ist ein auf XML basiertes Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können.

[26] REST Representational State Transfer bezeichnet eine Query Syntax für Webanwendungen. Es gibt keine explizite Norm, lehnt sich aber an den HTTP Kommandos GET, POST, PUT, DELETE an. Die URI wird als eindeutige Repräsentanz einer Datenmenge verwendet.

[27] JSON JavaScript Object Notation, ist ein kompaktes Datenformat in einfach lesbarer Textform zum Zweck des Datenaustauschs zwischen Anwendungen. Jedes gültige JSON-Dokument soll ein gültiges JavaScript sein und per eval() interpretiert werden können. JSON ist nicht standardisiert.

[28] Contracts sind eine sprachenübergreifende Vereinbarung zum Austausch von Objekten.

Eine weitere Möglichkeit von Windows 8 ist, per Push den Empfang eines Faxes dem Benutzer zu signalisieren. Ob dies genutzt wird, wird im späteren Machbarkeitsteil evaluiert.

6 Windows 8 Plattform-Betrachtung

Wenn man Software-Spezifikationen schreibt, kann man erheblichen Entwicklungsaufwand sparen, indem man sich an den technischen Eigenschaften der Zielplattform orientiert. In gängigen Entwurfsmodellen, wie UML[29], wird bevorzugt die konkrete technische Implementierung außen vor gelassen. Kunden wünschen häufig exakte Nachbildungen Ihrer Vorstellungen ohne Kenntnis der daraus resultierenden Kosten. Um den Implementierungsaufwand des Fax Clients möglichst gering zu halten und gleichzeitig maximale Funktionalität unter Nutzung aller Möglichkeiten von Windows 8 zu gewährleisten, erfolgt im nächsten Schritt eine Untersuchung und Beurteilung. Ziel ist es, diese Erkenntnisse im weiteren Entwurf zu verwenden. Da es weder Literatur noch eine im Ansatz komplette Dokumentation gibt, muss dies hier in Kurzform erfolgen. Es kann also nicht auf Quellen verwiesen werden.

Vorausgesetzt werden Kenntnisse in Web Service-Architekturen, XAML und .NET.

Die folgende Analyse der Software-Entwicklung für Windows 8 bezieht sich zum einem auf die bisher formulierten Anforderungen und die spezifischen Eigenheiten von Windows 8. Es ist kein Handbuch und keine Dokumentation. Soweit möglich wird immer der der Einstiegspunkt angeführt. Ansonsten beschränken sich die Ausführungen auf Architektur und Konzeption.

6.1 Merkmale von Windows 8

Auf den ersten Blick ist Windows 8 ein Windows 7 mit einem neuen Startmenü, den Kacheln, auch Tiles genannt. Dann fällt noch auf, dass es eine neue Kategorie von Anwendungen gibt, die Metro Style Apps. Diese sind rahmenlos und im Vergleich zu einer Windows Desktop-Anwendung erheblich in Funktion und Aussehen beschränkt. Faktisch sind dies zwei Shells. Auf einem ARM Gerät (siehe 2.1.1) wird es nur die Metro Shell geben und somit keine Möglichkeit mehr, eine typische Windows-Anwendung zu nutzen. Wenn man aber in die Details blickt, steckt wesentlich mehr im neuen Windows. Es ist ein Bruch mit der Vergangenheit, optimiert auf neue Gerätetypen, wie Smartphone und Tablet PC. Folgende wesentliche Merkmale unterscheiden die neuen Windows-Version von den Vorgängern.

- Gemeinsame Runtime WinRT
- Stromsparendes Design – Anwendungen können in einen Suspended-Modus versetzt werden
- touch-optimierte Oberfläche

[29] Siehe auch 5.1

- Metro Designsprache
- Immer Fullscreen[30]
- Maximal zwei Vordergrundanwendungen
- Background Task-Prinzip kann auch nicht laufende Anwendungen bedienen
- meist asynchrone Ausführung
- Weitreichende Sensor-Unterstützung
- Gemeinsame Funktionen per Contract – z.B. Suche über alle Anwendungen
- Funktionsintegration ins Betriebssystem, wie aktive Kacheln
- Fallbedingte UI-Dialoge, wie gedrehter Bildschirm

Persönlich schätzt der Autor den Unterschied zwischen Windows 7 und Windows 8 in weiten Teilen ähnlich groß ein, wie der Umstieg von DOS auf Windows. Es wird eine völlig neue Generation von Anwendungen in Sachen Funktion und Oberfläche entstehen. Beispielhaft sei hier die Bedienung des Fernsehers über das Smart Device[31] genannt.

Abbildung 3 Windows 8 Startbildschirm

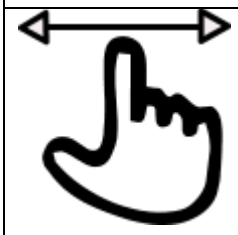
6.2 Bedienkonzept

Auch wenn Touch das zentrale Bedienkonzept von Windows 8 ist, so lässt sich doch jede Anwendung mindestens genauso gut mit der Maus und Tastatur bedienen. Der Entwickler muss dies auch kaum berücksichtigen, da die vorhandenen Steuerelemente diese automatisch handhaben. Überraschenderweise reduziert sich die Anzahl der Gesten auf einige wenige. Im Folgenden werden Gesten und deren Anwendung beschrieben.

[30] Wenn zwei Anwendungen gleichzeigt angezeigt werden sind dies Ansichten Snapped und Filled

[31] http://samsungremotewp7.codeplex.com/

Tabelle 2 Touch Gesten Überblick

	Tap: Führt Aktion aus, ähnlich dem Mausklick, also zum Start einer Anwendung oder Auswahl eines Listenelements. Es existiert auch eine Variante Double Tap.
	Tap Hold: Üblicherweise werden so Aktivitäten mit der Zwischenablage ausgelöst.
	Zoom: Damit vergrößert der Benutzer beispielsweise ein Bild. Wird auch genutzt mit dem Semantic Zoom Control, um gruppierte Sichten einer Datenmenge darzustellen. Das Startmenü nutzt dies, um eine Navigation über die Anfangsbuchstaben zu ermöglichen.
	Turn: Drehen eines Objektes.
	Slide horizontal: Damit kann der sichtbare Fensterausschnitt verändert werden, wie beim Startmenü. Slide über den linken Bildschirmrand wechselt die sichtbare Metro App. Slide auf der rechten Seite über den Bildschirmrand von außen zeigt das Charm Bar Menü.
	Slide vertikal: Eine Anwendung wird geschlossen, indem man über den oberen Bildschirmrand diese ganz nach unten zieht. Wenn man nur wenige Zentimeter zieht, werden die App Bar Menu Controls eingeblendet. In Grid-Darstellung wird der Slide auf dem Objekt nach unten genutzt, um das Objekt auszuwählen, wie zum Beispiel im Startmenü.
	Typischerweise werden die Slide-Gesten mit dem Daumen ausgeführt. Mögliche Aktionen App Bar anzeigen, App Bar Aktion ausführen, Charm Bar einblenden.

Windows 8 bietet für die Gestenerkennung eigene Events an. Der Entwickler kann auch eigene komplexe Gesture Events erstellen, wie ein vertikaler Slide mit zwei Fingern.

Die Events zu den Gesten sind Tapped, DoubleTapped, RightTapped und Holding.

Häufiger wird man mit den Pointer Events arbeiten, die universell für Maus, Stift und Touch ausgelöst werden. Aus .NET bekannte Events, wie MouseleftButtonDown existieren in WinRT nicht mehr.

Für Aktionen, die mehrere Finger benötigen, wird das Event Manipulation verwendet.

6.3 Metro Designsprache

Mit Windows Phone 7 hat Microsoft eine neue Designsprache[32] eingeführt, angelehnt an den Bauhaus-Architekturstil. Dieser propagiert den Verzicht als zentrales Designelement. Beispielsweise sind Serifen in Schrifttypen überflüssig. Die entsprechenden Guidelines sind sehr umfangreich. Zusammengefasst ergeben sich folgende Schwerpunkte.

- Verschlankung der Oberflächen
- Navigation
- User-zentrierter Entwurf
- Typografie

Microsoft bezeichnet aktuell Anwendungen, die für WinRT geschrieben werden als Metro Style App. Aufgrund von Konflikten um die Marke Metro, wird sich dieser Name ändern.

6.3.1 Single Window-Anwendungen

Windows Metro Style Apps sind Chromeless [33]. Das bedeutet, sie haben keinen Rahmen oder Close Icon und füllen den Bildschirm aus. Darüber hinaus gibt es auch keine Mehrfachfenster, wie MDI[34] oder Pop-Up-Dialoge. Anwendungen im Hintergrund werden automatisch pausiert. Als weitere Konsequenz gibt es auch keine Steuerelemente, die in solchen Oberflächen münden würden, wie beispielsweise ein Treeview.

Ganz generell wird empfohlen, der Anwendung „Raum“ zu geben. Ränder einzuhalten, keine Trennlinien oder Gruppierungsboxen verwenden, Leerflächen lassen.

6.3.2 Kacheln im Startmenü

Zuerst sichtbar ist die Kachel (Tile), über die die Anwendung gestartet wird. Der wesentliche Unterschied ist um klassischen Menü ist, dass diese als Minianwendungen ein Eigenleben entwickeln. Einfaches Beispiel ist, die Anzahl der Anrufe in Abwesenheit im Pro-

---

[32] http://download.microsoft.com/download/7/7/3/77371BBD-6613-4C1A-ACBF-08365C09D5FA/UI%20Design%20and%20Interaction%20Guide%20for%20Windows%20Phone%207%20v2.0.pdf

[33] Mit Chrome bezeichnet man alles was außen um das Dokument platziert ist, wie Scrollbar, Statusleiste, Menüleiste. Kiosk Mode Anwendungen sind auch Chromeless.

[34] MDI Multiple Document Interface ist Form der grafischen Benutzeroberfläche für Programme. Es können in einem Programmfenster gleichzeitig mehrere Dokumente geöffnet werden. Typischer Einsatzzweck sind Texteditoren.

gramm-Icon anzuzeigen. Mögliche Darstellungselemente sind Text und Bilder, die durch verschiedenste Mechanismen auf die Tile gelangen können, auch wenn die Anwendung nicht läuft. Dies wird als Live Tile bezeichnet.

Abbildung 4 Live Tile-Beispiele

Typischerweise sind Kacheln quadratisch oder doppelt breit, wie in der Abbildung. Der Benutzer kann dies jederzeit ändern, um sein Start Menu anzupassen.

Die Tiles können auch sozusagen doppelten Content beinhalten, der dann abwechselnd und animiert angezeigt wird. Maximal wird zwischen fünf verschiedenen Ansichten animiert gewechselt.

Zusätzlich können Tiles auch verwendet werden, um ähnlich wie in Websites bestimmte Stellen der Anwendung zu bookmarken und dann diese Stelle ins Menü zu pinnen. Dies wird als Secondary Tile bezeichnet. Anwendungsfall ist ein Programm zur Verwaltung von Aktien, in dem der Aktienkurs eines börsennotierten Unternehmens so auf den Startbildschirm geheftet werden kann. Auch dies sind Möglichkeiten, die man bisher so in der Software-Entwicklung nicht kennt.

Damit verbunden ist auch eine eigene Statusverwaltung, um die Anwendung in einen funktionsfähigen Zustand zu versetzen, ohne den üblichen Programm-Flow durchlaufen zu haben.

6.3.3 Navigation in einer App

Bisher arbeiten typische Computerprogramme mit einer Darstellung der Daten oder des Formulars am Bildschirm. Wenn der Screen nicht reicht, werden Bereiche auf und zuklappbar gehalten (Treeview, Expander, Tab). Für Funktionswechsel werden meist hierarchische Menüs verwendet. Das alles ist für Touch überhaupt nicht geeignet, da Finger breiter als ein Mauszeiger sind.

Es gibt zwei Grundkonzepte in Metro Design. Zum einen die Hierarchische Lösung, bei der sich unterhalb die nächste Stufe befindet. Für die nächste Hierarchiestufe wird in der Regel ein Element oder Datensatz ausgewählt und dann in die neue Ansicht gewechselt. Empfohlen wird die Anwendung mit dem Visual Studio Navigation App Template anzulegen, da dort automatisch mit Frame gearbeitet wird und so eine Vor-. und Zurück-Navigation möglich ist.

Die zweite Möglichkeit ist die Flache Navigation, die nur mit Wischgesten gesteuert wird. Sollte der Platz nicht reichen, wie bei einem mehrseitigen PDF, kann auch nach unten gescrollt werden.

Abbildung 5 Navigationskonzept Windows Phone 7

6.3.4 Oberflächen aufgabenspezifisch gestalten

Den Benutzer und seine aktuellen Bedürfnisse in das Zentrum zu stellen, sollte eigentlich selbstverständlich sein. Ist es aber heute nicht. Selbst in modernsten Anwendungen wird der Benutzer von einer Fülle von Funktionen, Bildern und Text überflutet, die funktionsgetrieben vom Entwickler im Dialog gruppiert angeordnet werden.

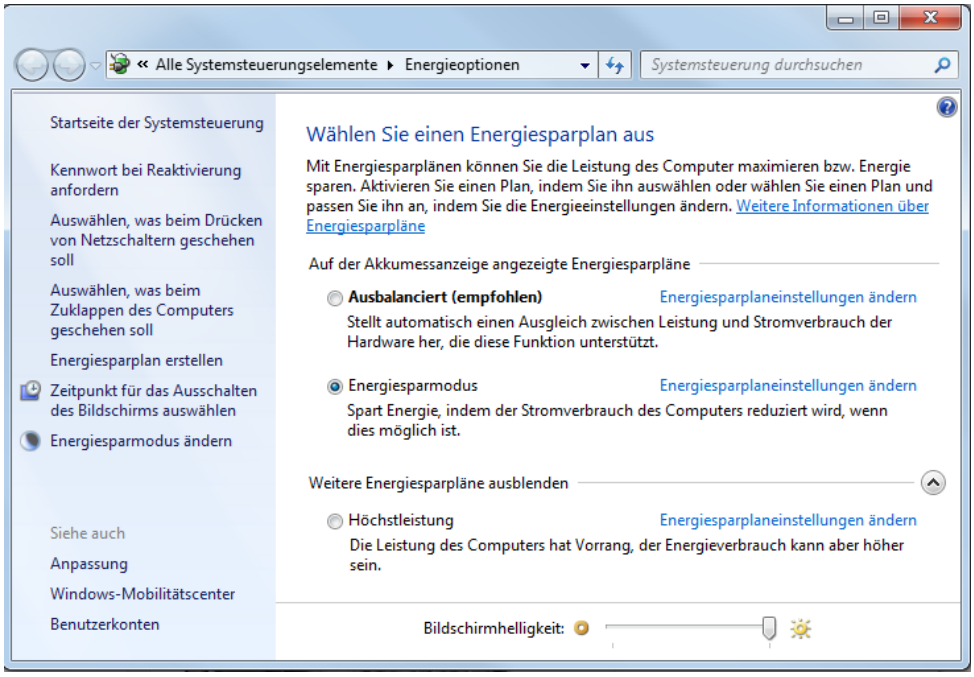


Abbildung 6 Windows 7 Systemdialog

Microsoft fordert von seinen Metro Designern, dem Benutzer Platz zu lassen. Das bedeutet auf Steuerelemente und Informationen zu verzichten, die hier und jetzt nicht benötigt werden.

Der Vergleich der Dialoge, hier für die WLAN-Verbindung zeigt das Konzept sehr gut.

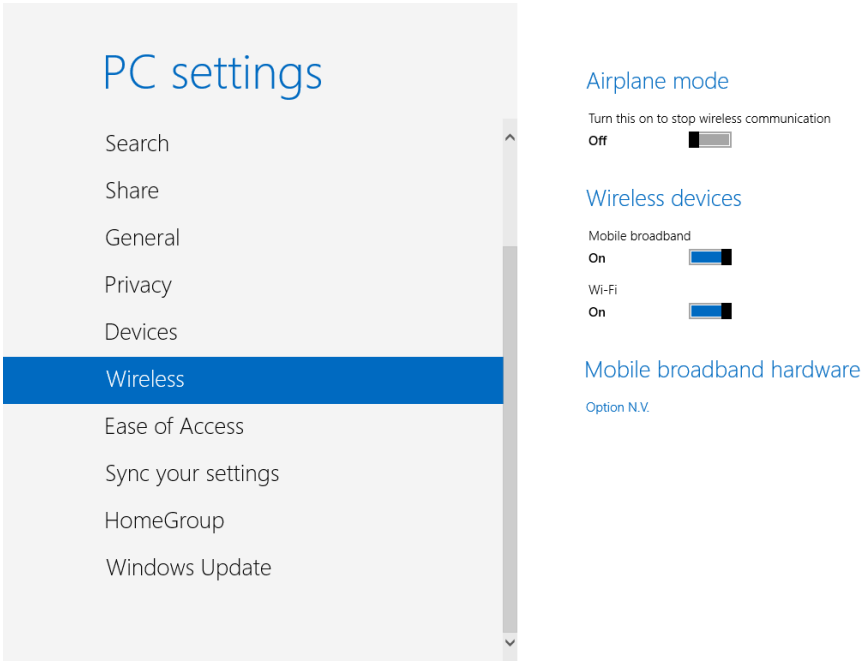


Abbildung 7 Windows 8 Metro Style Settings

Für Studienzwecke wurden bestehende Apps auf der iPad-Plattform untersucht. Auch dort wird sehr ähnlich vorgegangen. Benutzer scheinen es zu akzeptieren, weniger Möglichkeiten zu haben, auch wenn ein Windows Desktop-Anwender diese eigentlich gewöhnt ist.

Der Spagat, den der Entwickler hier machen muss, ist erheblich. Wird eine benötigte Funktion entfernt, kann es sein, dass die Benutzer die Anwendung ablehnen.

6.3.5 Symbolik und Typografie

In den letzten Jahren hat sich Design immer mehr dahin entwickelt, möglichst ausgefeilte Symbole in 3D-Optik zu entwerfen. Piktogramme helfen dem Menschen, sich schneller zu orientieren. Einige wenige Farben stehen für bestimmte Situationen, wie Rot für Warnung.

Abbildung 8 Windows Vista Icon

Faktisch sind Symbole aber nur sehr begrenzt und teilweise auch abhängig vom Kulturkreis nutzbar. Lokalisierung von Symbolen ist nach den Recherchen des Autors nach in der Software-Entwicklung unbekannt.

Microsoft hat mit dem portablen Media Player Zune seit 2006 einen typografischen Ansatz, der sich seit Windows Phone 7 mit der Metro Designsprache fortsetzt.

Abbildung 9 Symbole und Texte als Orientierungshilfe

Wenn man seine eigene Lebensrealität reflektiert, kommt man schnell zum Schluss, dass Symbolik oft an seine Grenzen stößt. Eine einfache Beschriftung kann aufschlussreicher sein. Hinzu kommt, dass mit Schriften viel leichter Kontrastwechsel realisiert werden können, wie sie in Hell/Dunkel-Situationen vorkommen. Nichtsdestotrotz ist eine rudimentäre Symbolik nützlich und entspricht eher menschlichen Lernmustern.[35] Bekannte Icons sind beispielsweise der Zurück-Pfeil oder - im aktuellen Kontext etwas antiquiert - die Diskette zum Speichern. Diese Icons werden in Metro so schlicht und einfarbig gehalten, dass ein Kontrastwechsel möglich ist.

6.3.6 Windows 8 Schriften

Die geläufigen Icons liegen als Windows-Schriftart vor. Der Name ist Windows Segoe UI Symbol und die Schrift ist auf jedem Windows 8 System vorhanden.

Abbildung 10 Segoe UI Symbol

Generell verlangt Microsoft[36] den sparsamen Einsatz von Schriften. Nur eine Variation in der Größe ist empfohlen. Vorherrschender Schrifttyp ist Segoe UI für Überschriften und Beschriftung eines Buttons. In Calibri soll Text eingesetzt werden, den der Benutzer liest und schreibt, wie in einem Chat. Für die bessere Lesbarkeit von längeren Textblöcken kommt Cambria zum Einsatz.

---

[35] Piktografische Darstellung Lidwell, Holden, Buttler (2003 S. 110)

[36] http://msdn.microsoft.com/en-us/library/windows/apps/Hh700394.aspx

Selbst die Größe wird vorgegeben. Standardwert soll 11pt sein. Dazu 9pt, 20pt und 42pt. Explizit abgeraten wird von kursiven Texten.

Eine Überschrift mit 42pt sieht folgendermaßen aus:

Für das generelle Seitenlayout gelten Regeln, die jeder Mediendesigner kennt. Microsoft bietet dazu spezielle Entwurfsmuster an, die nach Anwendungsfällen[37] optimierte User Interfaces erläutern.

6.4 Merkmale von WinRT

Neben den nativ entwickelten Programmen ist die Nutzung einer Runtime, die meist verbreitet Software-Architektur[38]. Windows 8 bietet auf dem Betriebssystem-Kernel eine Runtime, die WinRT genannt wird. Diese Runtime wird es für Intel 32/64 Bit und ARM-Prozessoren geben. Die Runtime bietet eine Reihe von Funktionen per API an, wie Datei lesen und Datei schreiben. Wer Komponenten für die WinRT schreiben möchte, kann dies per C++ tun. Das Komponentenmodell ist an COM[39] angelehnt.

Windows 8 ist zwar nach wie vor ein Multitasking System, hält aber weite Teile für den Anwender versteckt. In der Regel wird dieser nur eine oder maximal zwei Anwendung zeitgleich nutzen.

Auf der Runtime kann, in mindestens drei verschiedenen Frameworks programmiert werden.[40] Für diese Arbeit wird.NET herangezogen. Dabei handelt es sich um echtes .NET mit all seinen Eigenschaften, wie automatische Garbage Collection, Intermediate Language und den Sprachen VB.NET und C#. Microsoft spricht von einem limitierten .NET-Profil, da der Funktionsumfang erheblich reduziert ist. Es ist auch nicht möglich [41] .NET Assemblies außerhalb des Profils zu referenzieren. Die Beschreibung derselben erfolgt in einem neuen Format, WinMD (Windows Metadata).

Für den Entwickler ist es oft nicht ganz leicht zu erkennen, ob er gerade einen WinRT API Call durchführt oder eine .NET-Funktion aufruft. Wichtigster Hinweis sind die Namensräume, die mit System für .NET und Windows für WinRT beginnen. Dies ist von erhebli-

---

[37] http://msdn.microsoft.com/library/windows/apps/hh465424

[38] Gemeinsame Laufzeitumgebung. Vgl. Java Runtime

[39] COM Component Objekt-Modell

[40] Bereits ausgeführt in 1.5 HTML5/JavaScript, XAML/.NET, XAML/C++

[41] Es ist davon auszugehen, dass dies möglich ist, aber den Richtlinien nicht entspricht.

cher Bedeutung, da die Datentypen zwischen WinRT und .NET unterschiedlich sind und vom System per Projektion automatisch gemappt werden müssen.[42] In der folgenden Tabelle werden einige davon genannt und man sieht, dass die WinRT-Typen mit I beginnen.

Tabelle 3 Datentypen Auszug

Windows Runtime	**.NET Framework**
IIterable<T>	IEnumerable<T>
IIterator<T>	IEnumerator<T>
IVector<T>	IList<T>
IVectorView<T>	IReadOnlyList<T>
IMap<K, V>	IDictionary<TKey, TValue>
IMapView<K, V>	IReadOnlyDictionary<TKey, TValue>
IBindableIterable	IEnumerable
IBindableVector	IList
IObserableVector	ObservableCollection

Metro Apps laufen in einer Sandbox und sind in weiten Bereichen limitiert. So können nur die Verzeichnisse in Bibliotheken (Dokumente, Bilder, Videos, Musik) per Code gelesen und geschrieben werden. Wenn die Aktion Benutzer initiiert ist, üblicherweise per Filepicker-Steuerelement, können auch andere Verzeichnisse oder Laufwerke angesprochen werden.

6.5 .NET auf WinRT

Gerade bei Touch-Systemen muss das UI umgehend und bemerkbar auf Benutzeraktionen reagieren. Da typische Systemrückmeldungen, wie das Klicken des Buttons, fehlen, ist die Toleranzzeit extrem niedrig. Entsprechend muss jede Aktion, die länger als 50ms dauert, asynchron ausgeführt werden, um ein Blockieren des User Interfaces zu verhindern.

Asynchrone oder parallele Programmausführung gibt es in .NET natürlich schon länger. Dies erfordert aber Tasks oder Callback Event Handler. Mit der Kombination ASYNC und

[42] Vermutet wird, dass in der Planungsphase von WinRT .NET keine Betrachtung gefunden hat. Ursprünglich was der Weg über HMTL anvisiert worden. Die Reaktionen aus Community und Industrie hatten erst im Juni 2011 einen Schwenk zu XAML veranlasst.

AWAIT in .NET 4.5 kann man so programmieren, wie man dies beispielsweise von Winforms kennt.

```
Private Async Function Button_Click_1(sender As Object, e As
RoutedEventArgs) As Task
  Dim http = New HttpClient
  Dim resp = Await Http.GetStringAsync(
New Uri("http://www.heise.de"))
  text1.Text =
Windows.Data.Html.HtmlUtilities.ConvertToText(resp)
End Function
```

Listing 1 ASYNC und AWAIT-Beispiel

6.6 Applikationslebenszyklus

Energiesparende Hardware ist eines der zentralen Entwicklungsziele von Windows 8. Dazu zählen vor allem ARM-basierte Systeme. Die klassische Betriebssystemarchitektur von Windows nimmt aber kaum Rücksicht auf den Energiebedarf. Unzählige Dienste laufen kontinuierlich. Die Multiprocess- und Multithreaded-Unterstützung erzeugt eine konstante Last am Prozessor, auch wenn das Programm aus Nutzersicht nicht benötigt wird.

Windows 8 nimmt Metro Style Software aus dem Scheduler des Betriebssystems und legt diese nach üblicherweise fünf Sekunden in einen Suspended Modus[43]. Nach einer weiteren, nicht spezifizierten, Wartezeit gibt das Betriebssystem den Großteil des belegten Arbeitsspeichers wieder frei. Dennoch ist das Metro-Programm nach Aktivierung durch den Benutzer (Wipe von links) nahezu ohne Verzögerung wieder im Vordergrund und im vorherigen Zustand nutzbar.

Windows 8 kann außerdem völlig eigenständig Anwendungen, die sich im Suspended Modus befinden, komplett terminieren. Der Software-Entwickler muss diesem Umstand Rechnung tragen. Laut Dokumentation geschieht dies aus einer Reihe von Gründen, wie Speichermangel. Faktisch ist dem Autor bisher dieses Verhalten nicht begegnet.

Um diesen Lebenszyklus per Logik verwalten zu können, hat die zentrale Code-Datei (App.xaml.vb oder app.xaml.cs) entsprechende Events.

- Supsending
- Resuming
- Activated

Zusätzlich kann man per Code feststellen (*ApplicationExecutionState.Terminated*), ob die Anwendung erzwungen entladen wurde.

[43] http://msdn.microsoft.com/en-us/library/windows/apps/hh464925.aspx

Aus Entwicklersicht ist es deshalb wichtig, kontinuierlich relevante Statusinformationen fest zu speichern, so dass nie Daten verloren gehen.

6.7 Anwendungsdaten sichern

Alle Einstellungen der Anwendung müssen gespeichert werden. In modernen Anwendungen muss dies mehr oder weniger automatisch geschehen, jedenfalls so weit als möglich unter Verzicht eines Speichern-Buttons. Eine weitere Forderung von Metro ist, auch immer eine Rückgängig-Option anzubieten. Daher steht dies scheinbar im Widerspruch zum konstanten Speichern aller Einstellungen. Es handelt sich hier um typische Settings oder Session-Daten und nicht um echte Daten, die typischerweise als Datensätze in Datenbanken verwaltet werden.

Windows 8 bietet drei Speicherorte und zwei grundsätzliche Methoden an.

- Lokal lesen/speichern
- Temporär lesen/speichern
- Roaming lesen/speichern

Die erste Methode verwendet ein Key Value Dictionary. So kann ein Basis-Datentyp per Windows.Storage.ApplicationDataContainer-Klasse und entsprechenden Schlüssel abgelegt werden. Für größere Mengen können diese in Container (CreateContainer) gruppiert werden.

Die zweite Methode wird primär für komplexe Datentypen verwendet. Dabei wird ein beliebiges Objekt , zum Beispiel nach JSON, serialisiert und per Dateizugriff mit ApplicationData.Current.LocalFolder auf der Festplatte gespeichert.

Etwas ungewohnt ist die Möglichkeit, Settings in der Cloud zu speichern. Damit kann eine Anwendung nahtlos auf zwei oder mehreren Geräten genutzt werden. Für den Programmierer ist das ganz einfach. Es wird nur der Typ Windows.Storage.ApplicationData.Current.RoamingSettings dem DataContainer zugewiesen. Die Settings werden sofort und lokal gespeichert und erst nach wenigen Sekunden von Windows automatisch mit der Cloud synchronisiert. Um eine laufende Metro-Anwendung bei Änderung aktuell zu halten, gibt es das Event DataChanged, das ausgelöst wird und im Code behandelt werden muss.

6.8 XAML-Grundlagen

Die Extended Markup Language ist eine auf XML basierende Deklarationssprache für User Interfaces, die mit WPF[44] und Windows Vista eingeführt wurde. Ziel ist es, skalierbare Oberflächen für verschiedenste Bildschirmformate und Auflösungen zu gestalten. Dies

[44] WPF Windows Presentation Foundation ist ein Bestandteil des .NET Frameworks seit Version 3.0.

geschieht mit einem vektororientierten Ansatz. Windows 8 verwendet XAML sowohl für den .NET als auch C++-Entwicklungszweig.

XAML wird in Silverlight und WPF verwendet. Moderne Alternative ist die Kombination HTML5/CSS3, die allerdings in Funktion bei weitem nicht an XAML heranreicht. XAML wird erst zur Laufzeit kompiliert, HTML interpretiert.

Technisch noch erwähnenswert ist, dass durch direkte Verwendung von vorhandener Hardwarebeschleunigung XAML auch im 3D-Bereich performant zu nutzen ist.

Eine XAML UI besteht aus einer Page oder Usercontrol in der Container-Steuerelemente (Canvas, Grid, Stackpanel) eingebettet werden. Je nach Art des Container werden dort die eigentlichen Steuerelemente, wie Button oder TextBox, platziert.

```
<Grid>
  <Button Margin="26,22,347,245" Content="Hallo Welt" />
</Grid>
```

Listing 2 XAML-Beispiel

Mit WinRT kommt eine Reihe neuer ungewohnter Steuerelemente hinzu, die den Touch-Anforderungen geschuldet sind. Ein Beispiel ist der ToggleSwitch, der die Funktion eines (Licht)Schalters nachbildet und optisch größer ausfällt. Dies ermöglicht den Benutzern den Zustand auch zu erkennen, wenn sein Finger sich darauf befindet.

6.8.1 Content Templates

Eine der besonderen Stärken von XAML besteht darin, dass Controls keine Black Box darstellen. In den meisten Fällen kann man das Verhalten, die Darstellung und auch die Logik soweit beeinflussen. Damit entfällt in vielen Fällen die Notwendigkeit eigene Custom Controls zu entwickeln. Das wichtigste Konzept ist das Templating. Ein Control kann in seine verschiedensten Template-Bestandteile zerlegt werden, so dass aus einer Zeile XAML für den Button dann durchaus hundert Zeilen XAML Code werden können.

Dieser wird üblicherweise zwecks Wiederverwendung in Resources, auch per eigene Datei, ausgelagert. In diesem Beispiel, aber direkt innerhalb des Buttons. Man sieht, dass ein Button unter anderem auch aus einem Rechteck besteht.

```
<Button Margin="26,22,347,245" Content="Hallo Welt" Style="{DynamicResource
ButtonStyle1}" >
  <Button.Resources>
     <Style x:Key="ButtonFocusVisual">
       <Setter Property="Control.Template">
         <Setter.Value>
       <ControlTemplate>
         <Rectangle Margin="2" SnapsToDevicePixels="true"
Stroke="{DynamicResource {x:Static SystemColors.ControlTextBrushKey}}"
StrokeThickness="1" StrokeDashArray="1 2"/>
       </ControlTemplate>
```

Listing 3 Button Control Template internas

In der Praxis kann so aus einer Checkbox in der Darstellung eine Alufelge werden, die aber für den Entwickler nach wie vor ein Checked Event besitzt.[45]

6.8.2 Steuerelement animieren

Moderne Oberflächen geben dem Benutzer ein Gefühl, das der Lebensrealität entsprechen soll. Bewegungen unterliegen physikalischen Regeln und schwingen manchmal (bouncing). Diese Effekte lassen sich per XAML deklarativ einbauen. In der Praxis geschieht dies meist durch einen Designer, der wiederum eigene Werkzeuge, wie Microsoft Expression Blend verwendet. Um den Entwickler zu entlasten, haben die Steuerelemente bereits Animationen eingebaut. Wenn man diese verändern möchte, kann man dies über das Templating tun. Die Verwaltung der Animationen und Zustände erfolgt dann über den VisualState Manager.

In dieser Arbeit wird auf Einsatz dieser Möglichkeiten aus Platzgründen weitestgehend verzichtet.

6.9 Daten an Steuerelemente binden

Kernfunktion jeder Geschäftsanwendung ist der Datenfluss von Datenbank, Logikschicht und User Interface. Der grundlegende Ansatz der Datenbindung in Windows 8 ist die Verbindung jeweils einer Eigenschaft von zwei Objekten. Oft sind dies ein Steuerelement und dessen Textinhalt mit einem Datenobjekt bzw. dessen Property.

Datenbankzugriffstechnologien, wie das aus .NET bekannte ADO.NET fehlen in Windows 8 Metro-Anwendungen vollständig[46]. Durch das asynchrone Verhalten ist auch ein klassischer Workflow „Daten holen“ – „Controls zuweisen“ – „User editiert“ meist nicht praktikabel. Man bekommt Laufzeitprobleme. Das User Interface samt User kann schon aktiv werden, bevor die Daten endgültig eingetroffen sind.

6.9.1 MVVM Model und View

Nicht nur in Windows 8 kommt seit einiger Zeit das MVVM[47]-Entwurfsmuster zum Einsatz. Allerdings braucht man in vielen Fällen deutlich mehr Code im Vergleich zum klassischen Ansatz um die Trennung von UI und Logik zu bewerkstelligen. Das Entwurfsmuster gibt aber keine konkrete Implementierung vor, so dass sich in der Praxis zahlreiche Frameworks finden, die versuchen den entstandenen Mehraufwand zu kompensieren.

Der Autor sieht den Einsatz von MVVM in zwei Stufen. Dem reinen Datenmodell und den Kommandos. Letzteres, auch Command-Bindung genannt, wird in diesem Projekt vermut-

[45] Der Autor hat für Mazda den Car Configurator auf Basis von Silverlight entworfen. Dort wurde u.a. vom Templating einer Checkbox für die Alufelgenauswahl Gebrauch gemacht.

[46] Als Open Source Datenbank bietet SQLLite eine Metro-Variante http://www.sqlite.org/download.html

[47] MVVM Model-View-ViewModel Design Pattern

lich kaum verwendet, da hier der Aufwand meist nur getrieben wird, um automatische Unit Tests zu ermöglichen.

Das Viewmodel als Datencontainer benötigt auf Eigenschaftsebene das INotifypropertychanged Interface,beziehungsweise dessen Implementierung. Damit ist sichergestellt, dass gebundene Steuerelemente über Änderungen benachrichtigt werden und letztendlich die geänderten Daten auch anzeigen.

Im Gegensatz zu direkter Datenbindung in klassischer Programmierung hat der Entwickler/Designer zur Entwurfszeit die Daten nicht vorliegen. Man benötigt also Demodaten. Dies kann im ViewModel recht einfach realisiert werden, so dass im XAML UI Designer zur Entwurfszeit sichtbar ist, wie die Anwendung letztendlich auch aussieht. Dies ist gerade bei Listen sehr hilfreich.

6.9.2 Daten binden per Deklaration

Es gibt zwei entscheidende Attribute von XAML Controls um Datenbindung einzuleiten: Datacontext für die Zuweisung von ViewModels und Itemssource für generische Listen, die typischerweise auf dem Typ ObservableCollection basieren. Dabei ist es egal, ob die Zuweisung im Programmcode oder in der XAML-Deklaration geschieht. Wenn man Entwurfszeitdaten braucht, wird man in der Regel per Deklaration im Bereich Resources eines XAML-Elements wie folgt arbeiten.

```
<local:demoViewModel x:Key="MyVM" />
```

Die Namensraumdefinition „local“ erledigt Visual Studio 2012 im Page Header automatisch. Dann wird das derart instanziierte Objekt per Binding Expression zugewiesen.

```
<Grid DataContext="{Binding Source={StaticResource myVM}}">
```

Das Grid steht meist hierarchisch im XAML-Dokument an oberste Element. Bindings gelten durch die komplette Verschachtelung bis ins letzte Kind-Element. Aufgehoben wird das Binding durch ein neues Bindung oder durch die Zuweisung x:Null.

Jetzt erst beginnt das Binden an die einzelnen Attribute. Diese müssen zwingend vom Typ DependencyProperty sein. Folgend wird also mit demoViewModel.Eigenschaft der Textbox.Text gefüllt. Durch weitere Attribute wie Mode=TwoWay wird festgelegt, dass auch die Änderungen wieder zurück ans Objekt gegeben werden.

```
<TextBox x:Name="TextBox1"  Text="{Binding Eigenschaft, Mode=TwoWay}" />
```

6.9.3 Datentyp-Converter für Bindung

Typischerweise liest man in XAML-Code Dinge wie BackColor=“red“. Das dürfte eigentlich nicht funktionieren, da BackColor vom Typ Color ist, genauer gesagt sogar ein Brush um etwa Farbverläufe definieren zu können. Dabei ist aber „red“ eindeutig ein String. Eingebaute implizite Typ Converter erledigen die Konvertierung automatisch. Wer mehr

braucht, kann sich Converter mittels Interface iValueConverter und seinen beiden Methoden Convert und ConvertBack selber erstellen. Die Instanziierung erfolgt wie vorig mit MVVM per XAML local: Deklaration. Im Binding kommt dann ein zusätzliches Attribut Converter zum Einsatz.

```
Text="{Binding Path=StartDate, Converter={StaticResource dateConverter}}"
```

6.9.4 Benutzer benachrichtigen mit Notification

Letzter zwingender Bestandteil einer Bindung in beide Richtungen ist die Fähigkeit des Objektes, dem Steuerelement mitzuteilen, dass sich seine Inhalte ändern sollen. Das ist dem asynchronen Ausgangszenario einfach geschuldet. Dies geschieht über ein gemeinsames Interface INotifyPropertyChanged. Das Viewmodel muss dieses implementieren. Jede Eigenschaft, die sich ändert, muss das zugehörige Event PropertyChanged mit dem Parameternamen als Parameter auslösen. Eine ObservableCollection hat dies bereits implementiert.

Dieses Vorgehen repräsentiert einen kürzest möglichen Einstiegspunkt für die Dokumentation und den Namensraum System.ComponentModel[48].

6.10 Snapped, Flipped-Darstellung

Eine weitere Besonderheit von Apps für Windows 8 ist, dass sie auf verschiedene Bildschirmszenarien hin entworfen werden und müssen. Tablet-Geräte können systembedingt in einem Quer- (Landscape) und Hochkant-Modus (Portrait) betrieben werden. Hinzu kommt in Windows 8, das man zwei Anwendungen gleichzeitig auf dem Bildschirm unterbringen kann, sofern es die Auflösung zulässt. Davon befindet sich eine App im sogenannten Snapped Modus mit einer Breite von circa 320 Pixel und die zweite im Filled Modus mit 1024 Pixel. Nimmt eine Anwendung den gesamten Schirm ein, dann spricht man von Fullscreen und das sollte im Portrait-Modus mindestens 1366x768 Pixel oder ein skalierter Faktor davon sein.

Das Design einer Anwendung für diese unterschiedlichen Darstellungsarten erledigt man heute am einfachsten mit Hilfe des Werkzeuges Expression Blend. Dieses kann die sogenannten View States verwalten. Mit View States kann man Status-Informationen der UI deklarativ verwalten und deren Übergänge animieren. Das erklärt sich am Beispiel einer Ampel recht einfach. Es gibt drei Statuszustände (rot, gelb, grün). Der Übergang von Grün auf Gelb bzw. Rot kündigt sich, zumindest in Österreich, durch dreimaliges Blinken an. Das sind die Zustände und die Animationen dazu. Wie lange es rot ist, wird nach wie vor durch Programmlogik gesteuert. Im Falle von WinRT funktioniert dies mit dem Systemereignis ViewStateChanged.

[48] http://msdn.microsoft.com/de-de/library/z82ykwhb

6.11 Windows 8-Menüführung

Ein touch-orientiertes System erlaubt keine klassischen hierarchischen Menüs. Auch das Kontextmenü bereitet Schwierigkeiten, da es unterhalb des eigenen Fingers erscheint und so vom Benutzer nicht gelesen werden kann.

Das Kernkonzept beruht auf den sogenannten App Bars. Diese werden oben und unten am Bildschirmrand eingeblendet. Wie bereits im Punkt 5.2 beschrieben, erscheinen diese per Wipe-Geste oder auf Rechtsklick mit der Maus.

Der Designer muss dieses Steuerelement in einer Page (und nur dort) definieren und kann anschließend übliche Controls, wie Buttons, darin platzieren.

Allerdings gelten dabei einige Gestaltungsregeln:

- Sichtbar sind nur Buttons, die im aktuellen Kontext Sinn machen und nutzbar sind.
- Die Buttons werden rechts und links außen angeordnet, um mit den Daumen gut erreichbar zu sein.
- In der oberen App Bar werden Navigation oder Übersichten dargestellt.
- Die Buttons sind rund und greifen auf Standard-Designs zurück, die in Stylevorlagen definiert sind.

Abbildung 11 Internet Explorer 10 im Metro Style Top& Bottom Appbar

Zusätzliche Optionen auf einem Aktionselement, wie die Frage „Wirklich löschen?“ werden als Flyout realisiert, in XAML mit dem Pop-Up Control und dem Attribut islightDismissenabled=true.

Die App Bars können statisch angezeigt werden oder auf Benutzeranforderung und wieder automatisch verschwinden.

6.12 Netzwerk Verbindungen

Die Welt der mobilen Geräte ist ohne Cloud nicht denkbar. Die Daten liegen in der Cloud und werden von Services zur Verfügung gestellt. Soweit die Theorie. In der Praxis liegen diese noch sehr oft in klassischen RDBMS[49], wie dem Microsoft SQL Server. In .NET existiert mit ADO.NET eine Zugriffsschicht. WinRT kennt dies nicht. Alle Datenbankzugriffe, egal ob Cloud oder lokal, laufen über eine Service-Schicht. Unterstützt wird WCF in leicht eingeschränkter Form, basierend auf dem etwas schwergewichtigen SOAP.

Üblicherweise werden in Visual Studio-Projekten per Add Service Reference, Proxy Objekte generiert, die zwar leicht zu verwenden, aber wenig flexibel sind.

Mit dem ebenfalls unterstützen REST-Ansatz können am Service jederzeit Attribute hinzugefügt werden und der Client wird vermutlich weiter funktionieren. Die fehlende strenge Typisierung hat auch Nachteile.

Im jedem Fall wird heute in Windows 8 die Klasse HTTPClient verwendet um GET, POST, PUT, DELETE Requests an einen REST Service abzusetzen. Die Rückgabe erfolgt in einem serialisierten Format wie JSON.

Folgendes Beispiel liest einfache RSS Informationen, deserialisiert diese und weist sie als Liste einem Steuerelement zu.

```
Dim uri As Uri = New Uri("http://www.heise-
marktplatz.de/system/RSS/category/7")
Dim http As New HttpClient()
http.MaxResponseContentBufferSize = 20000
Dim ixml = XElement.Parse(Await http.GetStringAsync(uri))
Dim q = From x In ixml.Descendants("item")
Select New With {.Title = x.Element("title").Value,
.Link = x.Elements("link").FirstOrDefault.Value}
Gridview1.ItemsSource = q
```

Listing 4 HTTP Download eines RSS Feeds

Leider gibt es in der aktuellen Version kein Caching, das ein wiederholtes Laden der identen Daten verhindern würde. Dies muss manuell über die HTTP Header und lokales Speichern im Dateisystem abgebildet werden. Die Regeln dazu ergeben sich aus dem HTTP 1.1 Standard.

Viele Rest Services unterstützen eine umfangreiche Abfragesyntax. Mit einem Funktionsaufruf wie HoleKunden kann so gleichzeitig die Rückgabemenge mit Parametern eingeschränkt werden. Dies ist zwar nicht Windows 8-spezifisch, aber ein Konzept mit dem der Datentransfer erheblich optimiert werden kann. Paging sei als Beispiel genannt.

[49] RDBMS Relational Database Management Systeme

Eine besondere Herausforderung ergibt sich für den Entwickler auch in Sachen Network Requests. Nicht nur, dass eine Verbindung bestehen kann oder nicht (GetNetwork-ConnectivityLevel), die Verbindung kann auch Kosten verursachen. Gerade wenn man sich im Ausland im Roaming-Betrieb per UMTS befindet, wäre ein großer Download sehr teuer.

In den Netzwerk-Profilen (GetConnectionProfile) können alle Details (NetworkCostType) ausgelesen werden - bis zum Restkontingent (DataLimit) einer sogenannten Flat Rate. Ob Daten dann aus lokalen Zwischenspeichern oder in reduzierter Form übertragen werden, obliegt der Entscheidung des Entwicklers.

6.13 Hintergrundprozesse

Beinahe alles, was Software-Entwickler gewohnt sind, ist unter Windows 8 irgendwie anders. Ein Datentransfer, der einmal gestartet wird, kann sehr schnell unterbrochen werden, wenn die Anwendung suspended wird. Soweit die Tests zeigen, gibt es neben den fünf Sekunden fürs Suspenden nochmals den gleichen Zeitraum, also insgesamt zehn Sekunden, wenn ein Transfer läuft. Dann ist der Prozess definitiv beendet und die Anwendung ist vorerst ohne Prozessorzeit.

6.13.1 Background Download

Um trotzdem ,zum Beispiel ein laufendes Video, kontinuierlich zu transferieren, gibt es die Möglichkeit einen Background Transfer zu starten. Mit der Klasse BackGroundDownloader wird der Download als Datei behandelt und lokal gespeichert, auch wenn die Anwendung längst geschlossen ist. Wenn die Anwendung wieder aktiv wird, kann auf den fertigen oder noch laufenden Download wieder zugegriffen werden.

6.13.2 Background Tasks ausführen

Zeitgesteuerte Betriebssystemdienste oder die Ausführung von Hintergrundprogrammen sind in Windows 8 nicht möglich. Im Vordergrund steht Energieeffizienz, so dass es nicht möglich ist, dauerhaft und zeitgesteuert eine Funktion laufen zu lassen.

Es gibt natürlich Anwendungsfälle, die einen Prozess im Hintergrund benötigen. Ein Beispiel ist der Abruf eines Fax Services, ob neue Faxe angekommen sind. Gerade dafür wurden die Windows Push Services konzipiert, die später noch erläutert werden.

Mehrfach erwähnt wurde das Metro-Anwendungen sich vielerlei von klassischen Desktop-Anwendungen unterscheiden. Für Hintergrundprozesse gilt das noch vielmehr, gerade da eine Anwendung auch suspended oder inaktiv sein kann.

Das Windows 8-Konzept basiert auf Triggern, die ein Event auslösen können, aber nicht müssen. Wenn das Device auf Batterie läuft oder keine Netzwerkverbindung hat, reduziert sich die Sinnhaftigkeit, bestimmte Funktionen durchzuführen.

Trigger können sein: SMSReceived, PushNotificationTrigger oder TimeTrigger. Insgesamt gehören dazu rund zwanzig Trigger und sechs Bedingungen.

Man würde meinen, dass es ganz einfach ist, alle fünf Minuten einen Service zu befragen, ob neue Faxe da sind. Ist es aber nicht. Die minimale Zeitspanne für den TimeTrigger ist 15 Minuten und der Trigger funktioniert nur für LockScreen Apps [50]. Dafür funktioniert dies auch, wenn das Device ausgeschalten ist. Dies nennt Microsoft Connected Standby. In diesem Modus wacht das Gerät alle 15 Minuten auf, prüft ob eine Internet Connection vorhanden ist und teilt jeder Lockscreen-Anwendung zwei Sekunden Zeit zu, um Code auszuführen.

Wenn das Device an einer Stromversorgung hängt, gelten etwas freizügigere Regeln bezüglich BackGround Tasks. Die Metro App registriert mit der BackgroundTaskBuilder-Klasse seine Task. Der Code für den Task wird als Klasse in einem eigenen Projekt mit dem Interface IBackgroundTask implementiert. Im Manifest der Metro-Anwendung muss die Deklaration BackGround Task dem Entry Point mit dem Klassennamen zugewiesen werden.

Mit StetTrigger kann im Code der MaintenanceTrigger aktiviert werden, der ähnlich dem TimeTrigger funktioniert. Allerdings läuft der Code dann nicht in der Metro App, sondern im Adressraum des BackgroundTaskHost. Außerdem ist Schluss, sobald das Device auf Akkubetrieb wechselt.

6.14 Serialisierung von Daten

Nicht unbedingt neu ist die Anforderung, Objekte in andere Zustände überzuführen. Um ein Objekt Kunde von einem Ende der Leitung ans das andere zu übertragen, muss dieses Objekt serialisiert und deserialisiert werden. Ein weiterer Anwendungszweck ist das dauerhafte oder auch temporäre Speichern auf einem Medium. Für die Basistypen steht immer eine Methode ToString zur Verfügung. Ein Datum in der Form „13/12/13“ vermisst aber notwendige Metainformationen, in dem Fall die Culture; um es in jedem Fall korrekt wiederherstellen zu können.

Da der Vorgang rechen- und speicherintensiv ist, empfiehlt es sich, diesen in einem separaten Thread auszuführen, mit Async und Await ist dieser Vorgang recht simpel.

WinRT und .NET haben unterschiedliche Datentypen. Viele Standardtypen werden per Projektion automatisch abgebildet - andere, hier speziell Streams, jedoch nicht. Um die-

[50] Windows 8 Anwendungen die im LockScreen angezeigt werden.

ses Problem zu lösen, bietet .NET spezielle Extension-Methoden, [51] wie AsStream oder AsStreamforWrite.

```
Dim http As New HttpClient()
Dim resp As Stream = Await http.GetStreamAsync(url)
Dim ras = New InMemoryRandomAccessStream()
Await resp.CopyToAsync(ras.AsStreamForWrite())
' System.IO.WindowsRuntimeStreamExtensions
Dim bi As BitmapImage = New BitmapImage()
bi.SetSource(ras)
image1.Source = bi
```

Listing 5 Streams konvertieren

Extension-Methoden werden nicht von Intellisense vorgeschlagen, solange der passende Namensraum nicht per Imports (VB) oder Using (C#) Statement deklariert ist.

Aktuell sind zwei Formate verbreitet. JSON und XML. Dafür bietet .NET auch die passenden erialisierer-Klassen an. Für ersteres DataContractJsonSerializer und für zweiteres den DataContractSerializer aus dem Namensraum System.Runtime.Serialization. Die beiden Methoden ReadObject und WriteObject führen dann die eigentliche Serialisierung durch. Obwohl das durchaus dauern könnte, sind die Klassen nicht asynchron ausgelegt. Gegebenenfalls müsste man den Code in eine eigene Task auslagern.

Darüber hinaus kann man Serialisierer auch selbst entwickeln und es gibt im Web Source Code und Bibliotheken[52].

Selbst Microsoft bietet eine alternative Methode an, JSON zu lesen und zu schreiben, und zwar im Namensraum Windows.Data.Json. Ohne es gemessen zu haben, kann man davon ausgehen, dass es wesentlich schneller und damit effizienter ist, eine WinRT-Klasse zu verwenden. Es wird allerdings auch mehr Programmcode benötigt.

6.15 Einstellungen speichern

Für den Anwendungslebenszyklus ist es auf den ersten Blick nicht nötig, Daten zwischenzu-speichern. Wenn eine Anwendung von Suspended zu Aktiv wechselt, sind alle Einstellungen, Variablen und Benutzereingaben vorhanden. Allerdings möchte man dauerhaft Einstellungen speichern können, um so auch bei einem Terminate der App durch das Betriebssystem nach Neustart wieder die wichtigsten Einstellungen vorzufinden.

Die Microsoft-Beispiele beinhalten allesamt eine Hilfsklasse mit dem Namen SuspensionManager. Diese verwendet ein XML-Format und legt im Local Anwendungsverzeichnis

[51] Extension Methoden sind eine Spracherweiterung seit .NET 3.5, um bestehende Klassen mit zusätzlicher Funktionalität erweitern zu können, ohne in den Code der Klasse eingreifen zu müssen.

[52] http://json.codeplex.com/

eine Datei mit dem Namen _sessionsstate.xml ab. Grundsätzlich ist dagegen wenig zu sagen (Überlegungen zur Performance ausgenommen), so dass diese Klasse in das eigene Projekt kopiert wird.

In der Applikation-Klasse app.xaml.vb wird dann im Suspending und Resuming Event dieses Objekt genutzt.

```
Protected Async Sub OnSuspending(ByVal sender As Object,
ByVal args As SuspendingEventArgs) Handles Me.Suspending
        Dim deferral As SuspendingDeferral =
args.SuspendingOperation.GetDeferral
Await SuspensionManager.SaveAsync
        deferral.Complete()
End Sub
```

Listing 6 SuspensionManager speichert Session Daten

Hier sieht man, dass man mit dem Kommando Deferral das Suspending der Anwendung verhindern kann. Dies geht bis maximal zehn Sekunden, dann schließt Windows auch diese App.

Der SuspensionManager wird auch verwendet, um per Dictionary Key-Value Werte zu lesen und zu schreiben. Der Key ist immer String und der Value ein beliebiges Object.

```
SuspensionManager.SessionState("Wert")
```

6.16 Sensorik und Kamera

Auf den ersten Blick ist es sehr ungewöhnlich von der gewohnten Desktop-Anwendung zur per Sensorik gesteuerten App zu wechseln. Entsprechend sind die Klassen und auch Steuerelemente alle komplett neu. Hinzu kommt, dass manche Sensoren aggregiert werden, um eine höhere Genauigkeit zu erreichen, wie Bewegungssensoren. Andere Sensoren, wie Position, werden bei Abwesenheit einfach simuliert. Diese kann per GPS auf den Meter genau, per WLAN Triangulation auf 100 m genau, oder anhand der IP Adresse im Kilometerbereich bestimmt werden. Alles, ohne dass der Programmierer dies aufgreifen muss.

6.16.1 Umgebungshelligkeit

Der Lichtsensor liefert über die statische Klasse LightSensor die AnsiLumen. Änderungen an der Helligkeit können auch per Event ReadingChanged ausgelesen werden. Dabei ist einzig zu beachten, dass das Event im BackGround Thread ausgelöst wird, wie für WinRT typisch. So ist kein Zugriff auf das User Interface möglich. Ein Dispatcher hilft in dem Falle aber um aus dem Lightsensor Event das UI zu aktualisieren.

```
Private Sub eswerdelicht(sender As LightSensor, args As LightSen-
sorReadingChangedEventArgs)
   Dispatcher.Invoke(Windows.UI.Core.CoreDispatcherPriority.Normal, Sub(s, a)
```

```
                                                    textbloxk1.Text =
args.Reading.IlluminanceInLux
                                                                    End Sub, Me,
Nothing)
    End Sub
```

Listing 7 Thread Dispatching

In der Praxis wird man anhand von Schwellenwerten das User Interface zur Laufzeit verändern, um so zum Beispiel bei hellem Sonnenlicht die Lesbarkeit zu gewährleisten. Im dunklen würden helle Flächen dagegen blenden.

6.16.2 Lage und Position

Die Position oder Ausrichtung im dreidimensionalen Raum festzustellen, kann für Anwendungen von bestimmten Typen essentiell sein. Darüber hinaus kann man aus der Bewegung eine spezifische Geste erkennen und das Verhalten der Anwendung anpassen. Viele Telefone, die auf dem Display liegen, deaktivieren dieses und sparen so Energie und unterdrücken zusätzlich den Klingelton.

In der Programmierung ist die Vorgehensweise ganz ähnlich, wie mit dem Lichtsensor. Die statische Klasse OrientationSensor bietet ein Event Reading Changed. Die OrientationSensorReadingChanchedEventargs erlauben dann den Zugriff auf das „reading“ des Sensors. Wahlweise als RotationMatrix oder mit nur vier Parametern als Quaternation[53]. Per Parameter X, Y, Z und W sind so Beschreibungen von Bewegung im dreidimensionalen Raum möglich.

6.16.3 Kamera

Eine Kamera ist eigentlich kein Sensor, kann aber durchaus so verwendet werden. Hell/Dunkel-Erkennung wäre möglich, alternativ auch Bewegung oder natürlich einfach nur ein Foto machen. Auf dem Smartphone-Markt gibt es eine Anwendung, die anhand des Helligkeitsunterschieds des aufgelegten Fingers den Puls erkennt.

Direkter Datenaustausch zwischen Metro Apps funktioniert nur per Code Contracts. Technisch wird dies über ein gemeinsames Interface definiert. Es gibt eine Reihe von vordefinierten Contracts, wie Search[54] und Share. Es ist mit WinRT und den Contracts möglich, per Sharing Contract auf die Fotofunktionalität einer anderen Anwendung zuzugreifen. Primär soll aber das Bild eines Dokumentes mit Hilfe der Kameravorschau gemacht werden.

Moderne Tablets haben mehrere Kameras, sogar ein Scanner ist manchmal per Treiber als Kamera ansprechbar. Entsprechend braucht man die Liste der kameratauglichen Treiber in der DeviceInformationCollection.

---

[53] Beides mathematische Methoden, um Objekte im Raum zu beschreiben

[54] Vgl Kapitel 6.18

Weiter wird per MediaCaptureInitializationSettings ein Kameraobjekt initialisiert und die aktive mit VideoDeviceID ausgewählt. Es kann sinnvoll sein, die Settings wie Auflösung oder Helligkeit anzupassen (VideoDeviceController.Brightness).

Um ein Dokument komplett ablichten zu können, benötigt der Benutzer ein Vorschau Fenster. Für diesen Zweck gibt es ein eigenes Steuerelement mit dem Namen CaptureElement dessen Attribut Source die Media Quelle zugewiesen wird.

```
Dim mcs = New MediaCaptureInitializationSettings()
mcs.VideoDeviceId = cams(0).Id
mc = New MediaCapture()
Await mc.InitializeAsync(mcs)
Capture1.Source = mc
```

```
Await mc.StartPreviewAsync()
```

Listing 8 per Mediacapture Bild aufnehmen

6.17 Lokale Dateien

Ob Download oder soeben aufgenommenes Bild, es besteht die Notwendigkeit, diese Daten dauerhaft zu speichern. Windows 8 beschränkt die Dateizugriffe auf die Ordner der Bibliotheken (Dokumente, Bilder, Videos, Musik).

Ausnahme davon ist der Fall, dass der Benutzer den Dialog per Windows.Storage.Pickers.FileOpenPicker startet. Dieser kann per Contract arbeiten, damit sogar aus den Daten einer anderen Anwendungen gewählt werden, sofern diese als Source definiert ist. Anwendungsbeispiele sind Bilder. Viele Anwendungen, wie beispielsweise Facebook pflegen eigene Bilddaten aus denen der Benutzer ebenfalls auswählen kann.

```
Dim fo As FileOpenPicker = New FileOpenPicker()
fo.ViewMode = PickerViewMode.Thumbnail
fo.FileTypeFilter.Add(".png")
Dim fileList As IReadOnlyList(Of StorageFile) = Await fo.PickMultipleFilesAsync()
```

Listing 9 FileopenPicker angewendet

In vielen Anwendungsfällen wird der Benutzer allerdings erwarten, dass sich das soeben geschossene Foto ganz automatisch an der gewohnten Stelle im Dateisystem befindet. Dazu wird die Methode CapturePhotoToStorageFileAsync verwendet. Im folgenden Beispiel wird in einem Image Control das erstellte Bild noch angezeigt.

```
Private Async Function Button_Click_3(sender As Object, e As RoutedEventArgs) As
Task
  Dim sf As StorageFile = Await KnownFold -
ers.PicturesLibrary.CreateFileAsync("nurtest.jpg", CreationCollision-
Option.ReplaceExisting)
  Dim img As ImageEncodingProperties = New ImageEncodingProperties()
  img.Subtype = "JPEG"
  img.Width = 2048
  img.Height = 1536
  Await mc.CapturePhotoToStorageFileAsync(img, sf)
  img2.Source = New BitmapImage(New Uri(sf.Path))
```

```
End Function
```

Listing 10 Bilddatei erzeugen

Der Verzeichnisname „Bilder“, oder bei Englischen Betriebssystem „Images“, für den Speicherort, ist immer der gleiche, dank der KnownFolders-Klasse.

Die dritte Möglichkeit, Bilder auszutauschen und zu speichern ist die Zwischenablage. Damit kann der Benutzer ein Bild auch in eine klassische Desktop-Anwendung übernehmen. Verwaltet werden die unterschiedlichen Typen an Daten in der Zwischenablage über DataPackageView Element, die man in die statische Clipboard-Klasse per GetContent oder SetContent liest und schreibt.

6.18 Daten visualisieren

Nachdem bereits in das Konzept der Datenbindung und das dafür nötige Interface eingeführt wurde, wird nun die konkrete Umsetzung beschrieben. Die größten Unterschiede bemerkt man in den Steuerelementen und deren Windows 8-spezifische Eigenschaften. Gewohnte Controls wie DataGrid, DataPager oder Treeview existieren nicht.

6.18.1 ListView

Der Einsatzzweck des Listview-Steuerelements ist die Darstellung von Daten in einer Liste. Es ersetzt damit Listbox und Combobox. Beide sind aber noch vorhanden.

Gescrollt wird von oben nach unten. Selbstverständlich ist das Control touch-optimiert und unterstützt entsprechende Gesten, wie Eintrag selektieren.

Spezielle Events und Funktionen für Neuer Datensatz, Daten ändern oder Daten löschen fehlen und müssen manuell etwa per App Bar Menü implementiert werden.

Über das Attribut ItemSource sind Listen zu binden. Das kann ein String Array sein, weit häufiger aber eine generische Liste. Sobald die Liste sich ändert, bietet sich der Typ ObservableCollection [55] an. Diese implementiert das Notification Interface und kann über LINQ[56] auch sortiert oder gefiltert werden.

Um aus einer generischen Liste die passende Eigenschaft zur Anzeige auszuwählen, exisitert das Attribut DisplayMemberPath, das üblicherweise im XAML-Code deklariert wird. Für den dahinter liegenden Wert passt SelectedValuePath.

[55] Stellt eine dynamische Datenauflistung dar, die Benachrichtigungen bereitstellt, wenn Elemente hinzugefügt oder entfernt werden, oder wenn die gesamte Liste aktualisiert wird.

[56] LINQ - Language Integrated Query ist eine Erweiterung des Microsofts .NET-Framework zur Abfrage von beliebigen Datenquellen. Statt SQL-Kommandos als Text zu schreiben, werden diese unterstützt durch intellisense und Compiler direkt codiert.

Wenn die Liste mehr als ein Feld anzeigen soll, etwa ein Bild mit Text, wird das Itememplate angepasst.

Das folgende konzeptionelle Beispiel bindet das Listview an eine Eigenschaft des ViewModels-ListenItems. Das Viewmodel muss wiederum per Resource Declaration instanziert werden. Innerhalb der ListItems-Auflistung gibt es eine Eigenschaft FeldName, die an die Texteigenschaft eines Controls gebunden wird.

```
<ListView ItemsSource="{Binding ListenItems}"
  <ListView.ItemTemplate>
    <DataTemplate>
        <TextBox Text="{Binding FeldName}"  />
     </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

Listing 11 Datatemplate exemplarisch mit Listview

Häufig sieht man auch, dass die Template-Definitionen nicht wie hier innerhalb des Controls vorgenommen werden, sondern in der Page oder gar im APPX Resource Dictionary.

```
<ListBox   ItemTemplate="{StaticResource tmplate}" >
```

Das Ziel ist, den XAML-Code in der einzelnen Seite überschaubar zu halten. Es gibt noch eine Reihe weiterer Templates.

Obwohl die Darstellung vertikal ist, lässt sich mit dem ItemsPanel auch das verwendete Stackpanel austauschen, so dass eine horizontale Ausrichtung erfolgt. Ein sinniger Anwendungsfall ist der Einsatz des CarouselPanel Controls, das eine rotierende Darstellung erlaubt. Das heißt, wenn man am Ende der Liste angekommen ist, geht's vorne wieder los.

Das Listview-Steuerelement bietet sich auch als alternative Sicht auf die Daten im Snapped Modus an.

6.18.2 Gridview

Das GridView ist im Grunde genommen nahezu ident mit dem Listview, spezialisiert für horizontale Darstellung. Der Benutzer scrollt von links nach rechts und umgekehrt. Das kann per Touch, Scrollrad oder auch Mauszeiger geschehen.

Das Windows 8 Startmenü dürfte intern ein GridView sein. Dabei fällt auf, dass es dort Gruppen gibt, die als Überschrift in den Bereichen auftauchen. Dies wird später beim Semantic Zoom-Steuerelement noch weitere Bedeutung erlangen.

Um Daten in ihrer Darstellung zu variieren, wird in WPF oder Silverlight eine Weile die Klasse CollectionViewSource verwendet. Diese existiert auch in WinRT. In der Regel wird die Instanz von CollectionViewSource in der XAML-Datei deklariert und diesem per Source-Attribut eine Liste zugewiesen. Bereits die Liste muss die Gruppierungsinformati-

onen beinhalten. Verschiedene Möglichkeiten, um gruppierte Daten zu erstellen, finden sich in den SDK-Beispielen[57] von Microsoft.

6.18.3 SemanticZoom

Um in größeren Datenmengen per Touch navigieren zu können, bietet sich das Semantic Zoom-Steuerelement an. Dieses beinhaltet zwei Templates, den ZoomOutView und den ZoomInView.

Der ZoomOutView stellt in der Regel die Übersicht dar. Mit Benutzer-Click auf einen Eintrag wechselt die Darstellung animiert in den ZoomInView, in dem die kompletten Daten sichtbar werden. Häufig wird für die Übersicht eine Listview oder StackPanel verwendet. Das Binding des darstellenden Steuerelements erfolgt über den Groupname aus der Datensicht.

Die beiden unterschiedlichen Sichten auf die Daten können eine gruppierte Darstellung sein, müssen es aber nicht.

Zwingend benötigt wird ein CollectionViewSource-Objekt, dessen Attribut IsSourceGrouped gesetzt wird. Die Daten müssen beide Views und die Verbindung derer enthalten. Die im ZoomInView und ZoomOutView beinhalteten Listensteuerelemente bekommen dann über die Eigenschaft ItemSource den jeweiligen View zugewiesen.

6.19 Search

Die Style Guides von Windows 8 erfordern, dass die Suche über den rechten Bereich, Charm Bar genannt, erfolgen soll. Das heißt, der Benutzer streicht von rechts über den Bildschirmrand und bekommt eine universelle Suchbox. Mit dieser sucht man nicht nur im aktiven Programm, sondern in allen Programmen, sofern sie den Search Contract implementiert haben.

Um einen Search Contract zu implementieren, muss im Manifest der Metro App die Deklaration für Search hinzugefügt werden. Wird eine Suche in der Anwendung vom Benutzer durchgeführt, wird das Event OnSearchActivated ausgelöst. Dies kann global per app.xaml.vb für die ganze Anwendung gelten oder für einzelne Views. Wenn die Suche aus der Sicht angestoßen wird, macht es Sinn in dieser zu suchen.

Wenn der Benutzer global nach einem Begriff sucht, bieten die meisten Metro Apps eine Standardliste an, die für die spezifischen Daten dieser Anwendung gelten. Für diesen Zweck bietet Visual Studio 2012 ein Template an, dass über Add Item als Search Contract enthalten ist.

[57] SDK Software Development Kit-Beispiele sind das hauptsächlich genutzte Mittel, um diese Arbeit zu schreiben. Die Dokumentation der API ist speziell im .NET-Bereich mangelhaft und unvollständig.

Die beschränkte Suche in der Sicht einer Anwendung wird über das Event QuerySubmitted realisiert. Mit dem Parameter QueryText kann die Ergebnisliste eingeschränkt werden. Üblicherweise nimmt man dazu ein LINQ Statement und baut in die Where-Bedingung den QueryText ein. Die Query wird dann wieder der Itemsource eines Listensteuerelements zugewiesen.

6.20 Share Contract

Der Datenaustausch zwischen Metro Apps auf der gleichen Maschine erfolgt mittels Contracts. Es gibt immer eine Source und ein oder mehrere Target-Anwendungen. So kann ein Programm für Bildverwaltung das Bild per Mail versenden. Die Mail-Anwendung kann Bilder aus unterschiedlichsten Anwendungen mit gleichem Contract Interface in eine Mail einbetten.

6.20.1 Share Source

Jede Anwendung kann Daten zur Verfügung stellen und so anderen Anwendungen erlauben, diese zu verwenden. Unterstützte Standardformate sind unter anderem, Text, RTF oder Bitmap. Darüber hinaus werden auch eigene Formate unterstützt, für die natürlich auch ein passender Empfänger verfügbar sein muss.

Das Teilen ist ein vom Benutzer eingeleiteter Vorgang. Der Benutzer streicht vom rechten Bildschirmrands und aktiviert damit die Charm Bar und wählt das Symbol Share. Zu diesem Zeitpunkt wird in der aktiven Anwendung das Event DataRequested des DataTransferManagers ausgelöst. Der Entwickler muss in diesem Event die Parameter dem Argument DataRequestedEventArgs übergeben. Das sind z.B. Request.Data.Properties.Title oder Description. Die eigentliche Information wird mit einer Set Methode typspezifisch zugewiesen. Für Text, Request.Data.SetText, bei ganzen Dateien SetStorageItems.

In der Charm Bar erscheinen passende Anwendungsziele. Das ist nach aktuellem Stand für Text nur die Mail App. Wenn mehrere Anwendungen diesen Datentyp gebrauchen können, erscheinen diese auch alle untereinander aufgereiht rechts im Charm-Bereich. Der Benutzer muss dann eine wählen oder abbrechen, indem er woanders hinklickt.

Im Code der Source App wird das Event TargetApplicationChosen gefeuert. So kann man dem Benutzer noch zusätzliche Optionen anbieten, die nur im Context der bestimmten Zielanwendung Sinn machen.

6.20.2 Share Target

Um Windows bekannt zu geben, dass die eigene Anwendung ein mögliches Ziel ist, muss im Manifest der App der Share Contract hinzugefügt werden. In der Regel wird man dort auch Einschränkungen auf „Bitmap“, „Text“ oder „HTML“ vornehmen (es existieren noch weitere Typen).

Im Event SearchActivated der Anwendungsklasse app.xaml.vb wird dann eine Instanz der Share View Page erstellt. Es wird nicht die Anwendung mit seiner Standardstartseite gestartet, sondern eine spezielle Page. Diese beinhaltet Share-typische Elemente, wie eine Image Thumbnail-Vorschau und belegt nur einen Bruchteil der Seite.

Der Benutzer wählt also in einer App (zum Beispiel. ImageViewer) das Icon Share im Charm. Dann erscheint anhand identer Typen eine Liste der Anwendungen, die als Target dienen können, aus der der Benutzer die gewünschte auswählt (z.B. Mail). Von dieser Anwendung wird die ShareTargetPage angezeigt.

Dann kann die übermittelte Information in der Zielanwendung verwendet und der Vorgang abgeschlossen werden.

Visual Studio 2012 bietet einen Wizard „Share Target Contract“ an, der die Einstellungen im Manifest vornimmt, die Anwendungsklasse ergänzt und die ShareTargetPage anlegt.

6.21 Protocol Activation

Es gibt in Windows 8 zwei Möglichkeiten, um eine Anwendung von einer weiteren Anwendung aus zu starten.

Der eigenen Anwendung kann per Manifest die Dateierweiterung einer oder mehrerer Dateien zugewiesen werden. Konkret ist dies eine „File Type Association“, wie .jpg oder .txt. Wenn nun so eine Datei vom Benutzer im Explorer per Doppelklick geöffnet werden möchte, wird unter anderem die eigene App dafür angeboten.

Es kann deklarativ im XAML mit dem vordefinierten Protokoll ms-appx auf eine Bilddatei verwiesen werden, die sich innerhalb des eigenen Anwendungspakets befindet.

```
<Image Width="347" Source="ms-appx:///Images/ppedv.jpg"/>
```

Einen ganz ähnlichen Ansatz bietet die Aktivierung einer Anwendung per Protokoll. Mit dem aus HTML bekannten mailto:hannesp@ppedv.de?subject=Betreff wird die Mail App gestartet. Ein weiteres bekanntes Protokoll ist http://, das genauso wie https:// in Windows 8 unterstützt wird, um Ressourcen einzubinden oder eine Anwendung zu aktivieren.

Um nun in einem Text, der beispielsweise in einer E-Mail enthalten ist, eine Faxanwendung zu starten, könnte man ein Protokoll definieren wie fax://. Um eine Metro App mit dem Protokoll zu versehen, muss man dies in der Protocol Declaration des App-Manifest eintragen. Anschließend wird in der app.xaml.vb das spezifische onLaunched Event überschrieben und dort der Code eingetragen, der ausgeführt werden soll, wenn die App per Protocol Activation gestartet wird. Üblicherweise ist das die Instanzierung einer XAML-Seite. Die Start-Parameter werden wie bei der MailTo Variante in der Kombination Host, Pfad und Querystring übergeben. Die URI-Klasse hilft bei der Zerlegung in die einzelnen Bestandteile.

Zum Aktivieren des Protokolls und damit zum Start der Anwendung kann die URL in den Adressbereich des Browsers oder des Datei-Explorers eingetragen werden. Um aus einer Metro-Anwendung eine zweite per Protocol zu starten, nimmt man die Launcher-Klasse. Der Methode LaunchUriAsync übergibt man dann die URI als Zeichenkette in der Form wie das eben mit der Mail gezeigt wurde.

6.22 Drucken

Das Drucken von mehrseitigen Dokumenten ist generell eine anspruchsvolle Aufgabe. Gerne wird daher auf Report-Generatoren und deren API ausgewichen. So etwas steht direkt in Windows 8 nicht zur Verfügung. Nach wie vor offen ist dagegen der Weg, das Reporting mit Standard Tools auf der Service-Schicht auf einem Server erledigen zu lassen und dann nur mehr den Report als PDF herunterzuladen.

Will man direkt auf dem Gerät drucken, muss das Zusammenstellen des Dokuments mit der Klasse PrintDocument sozusagen per Hand erfolgen. Es ist nötig, das Event Paginate für Seitenwechsel, das Event AddPages und das Event GetPreviewPage für eine Vorschau zu codieren.

Die Anzeige des Druckerdialogs erfolgt mit Printmanager.ShowPrintUiAsnyc im rechten Bildschirmbereich. Wenn der Druckerdialog per Charm ausgelöst werden soll, findet der Benutzer dies unter den Icon Devices, wenn ein PrintTaskRequested ausgeführt wird.

6.23 Contact und Contact Picker

Über die Contracts ist es einfach, Adressinformation aus anderen Anwendungen zu nutzen. Es gibt eine Windows 8 Metro Style App, die Kontakte verwaltet. Die Mail-Anwendung nutzt diese.

Wenn die eigene Anwendung Anbieter von Adressinformation sein soll, muss im Manifest der Contact Picker ausgewählt werden.

Um das Rad nicht zwei Mal zu erfinden, wird man häufig die bestehenden Contact-Quellen nutzen. Dazu wird eine Instanz der ContactPicker-Klasse erstellt und wahlweise mit der Methode PickSingleContactAsync oder PickMultipleContactAsync aufgerufen.

Der Benutzer muss dann die Metro App auswählen, die die Adressdaten liefern soll und erhält dann ein oder eine Liste von ContactInformation-Objekten. Dieses beinhaltet übliche Felder, wie Name. Andere sind als Collection ausgeführt, wie PhoneNumbers. Für alle weiteren Felder gibt es noch die CustomFields.

6.24 Push Service-Verfahren

Um eine Anwendung mit einem Service, wie z.B. Aktienkursen, verbunden zu halten und ohne wesentliche Verzögerung aktuell zu halten, gibt es zwei Szenarien: Man kann den

Service in Intervallen aufrufen oder der Service meldet sich bei der Anwendung. Push bietet zweiteres und damit auch die wesentlich ressourcenschonendere Alternative. Hinzu kommt, dass Push auch im Connected Standby-Modus, also wenn das Gerät keine Anzeige oder sichtbare Aktivitäten hat, unterstützt.

Da wir hier in einem sehr volatilen Bereich agieren, sei hier nur kurz das Konzept erklärt. Es gibt eine Website zur Verwaltung von WNS[58]. Auf dieser wird eine Anwendung vom Entwickler registriert. Er erhält dann für die Authentifizierung der App einen Token, dort als Client Secret bezeichnet. Dazu einen eindeutigen Package Name, der im Visual Studio-Projekt im gleichnamigen Feld des Manifests eingetragen wird.

Wenn die Anwendung das erste Mal vom Entwickler gestartet wird, muss über die Klasse PushNotificationChannelManager der Kanalname herausgefunden werden. Dieser bleibt über die Lebensdauer der Anwendung gleich. Allerdings gibt es keine Aussagen, was die Zukunft dazu bringt, folglich sollte man die URL nicht im Programm kodieren.

Der eigentliche Code, der beim Push ausgeführt werden soll, wird in das Event PushNotificationReceived programmiert. Über die Methoden Argumente des Events erhält man den Typ der Notification. Unterstützt werden heute Toast, Tile und Badge. Dazu im nächsten Abschnitt gleich mehr.

Zusätzlich wird eine Anwendung oder ein Service benötigt, der den Push Service aktiv benachrichtigt, wann er Pushen auslösen soll. Dieser benötigt die Service URL und den Paketnamen. Als Parameter wird der Push Typ übergeben und eine spezielle XML Syntax mit der Information für Tile, Toast oder Badge. Diese Anwendung kann ohne weiteres mit klassischem .NET als Winforms-Anwendung geschrieben werden, benötig aber etwas Code, da die Authentifizierung am Service mit dem OAuth-Protokoll durchgeführt werden muss.

Die Datenmenge im Push soll extrem klein gehalten werden und nur die eigentliche Nachricht umfassen, dass neue Informationen vorliegen. Falls der Benutzer diese in der Gänze benötigt, muss er den Request an einen üblichen WCF oder REST Service initiieren.

Dennoch gibt es eine RAW Notification mit der es möglich sein soll, auch eigene Datenformate zu übermitteln.

6.25 Tile und Toast-Benachrichtigungen

Was zuerst auffällt, sind die rechteckigen Menüpunkte im Startscreen.. Hinter diesen Kacheln steckt aber wesentlich mehr als nur eine Schaltfläche, um ein Programm zu starten. Tiles sind aktiv. Das heißt, sie können Ihr Aussehen, ihre Beschriftung und Ihre Bilder ändern. Laut Design Guidelines sollen sie das sogar. Der Inhalt der Kacheln wird mit einer

[58] https://manage.dev.live.com/AddApplication.aspx?tou=1. Es handelt sich dabei um eine Live ID oder Windows ID, die nötig zur Anmeldung beim Windows Notification Service WNS ist.

eigenen XML-Sprache deklarativ beschrieben. Dieses Format wird auch von Push genutzt. Wenn die Kachel nur einen einfachen Text ändert, nennt sich das Badge. Üblicher Anwendungsfall ist das Anzeigen der Anzahl der neuen E-Mails. Konkret können Zahlen von 1-99 oder eines von zehn Symbolen angezeigt werden. So ein Standardsymbol ist beispielsweise ein grauer Kreis im rechten unteren Eck der Kachel für unavailabale.

Das initiale Layout der Kacheln wird im Anwendungsmanifest definiert. Es beinhaltet zwei Images im Format 150x150 Pixel für die Kachel und 30x30 für Vorschau. Achtung: Kacheln können vom Benutzer auch auf doppelte Weite vergrößert werden!

Wesentlicher Unterschied zu bisherigen Windows-Anwendungen ist, dass die Tiles auch Funktionalitäten haben, wenn die Anwendung gar nicht gestartet ist. Ein Push erzeugt in der Tile neuen Inhalt. Es können aber auch andere Hintergrund-Events sein, die Tiles updaten.

Um eine Kachel zu füllen wählt man per Notifications.BadgeUpdateManager.getTemplateContent das passende XML Template und ersetzt die Elementwerte, im Falle des Badge „Value“ mit der gewünschten Info. Mit diesem XML wird dann letztlich die Kachel aktualisiert.

```
Dim badge = new BadgeNotification(badgeXml)
BadgeUpdateManager.CreateBadgeUpdaterForApplication().Update(badge)
```

Listing 12 Badge aktualisieren

Das Update der Kachel funktioniert analog nur mit einem anderem XML Template und der Methode TileUpdateManager.CreateTileUpdaterForApplication.

Von Vorteil ist hier, dass aktuelle Kacheln dauerhaft sichtbar sind, wenn der Benutzer auch aktiv diese betrachtet. Eine weitere Möglichkeit ist die laufende Anwendung mit einem Pop-Up, in Windows 8 Toast genannt, zu überlagern und so die Aufmerksamkeit auf die Neuigkeiten einer anderen Anwendung zu lenken. Chat-Programme tun dies in der Regel. Die Vorgehensweise mittels XML Template ist nahezu ident. Die Methode ToastNotificationManager.CreateToastNotifier() zeigt die Nachricht dann an.

Es können durchaus mehrere Toasts erscheinen, diese verschwinden auch wieder automatisch. Toasts sind anklickbar und lösen so einen Start der App aus. Darüber hinaus können Toasts auch WAV Files abspielen.

```
Dim audio = toastXml.CreateElement("audio")
audio.SetAttribute("src", "ms-winsoundevent:Notification.IM")
toastTextElements[0].AppendChild(audio)
```

Listing 13 Sound abspielen bei Notification

6.26 Security für Autorisierung

Sicherheit setzt sich aus Authentication und Authorization zusammen. Die Szenarien für Benutzeranmeldung werden vielfältiger. Windows 8 unterstützt eine Reihe von gängigen

Verfahren bis hin zum neuen OAuth[59], das unter Zuhilfenahme von Providern wie Facebook arbeitet. Die Web Authentication WinRT API beinhaltet die entsprechenden Klassen und Methoden, wie WebAuthenticationBroker.AuthenticateAsync.

Selbst die Anmeldung am Windows-System kann mit einer Web ID vorgenommen werden. Microsoft versucht die Anmeldung per Live ID (neu Windows ID) zu promoten. Windows 8 integriert sich so in Microsoft-Dienste, wie Skydrive, ohne dass der Benutzer jeweils eine neue Anmeldung vornehmen muss. Das Konzept wird SSO Single Sign on genannt.

Wenn man Active Directory-Authentifizierung möchte, muss im Manifest die Capability Enterprise Authentication aktivieren. Das wäre nötig, wenn sich ein Service innerhalb des Firmennetzwerk befindet. Einer HTTPClient Instanz wird dann ein HTTPHandler mit dem Attribut UseDefaultCredentials übergeben.

Auch Verschlüsselung in beiden Formen, symmetrisch und asymetrisch, werden unterstützt, um Daten lokal oder bei der Übertragung zu sichern. Der Namensraum dafür Windows.Security.Cryptography.

Zusätzlich ist auch ein Teil des .NET Security Systems vorhanden, im Namensraum System.Net etwa die Methode NetworkCredential. Für den Autor ist dies die erste Wahl um einen Webservice mit abweichenden Credentials aufzurufen.

[59] OAuth ist ein Protokoll, zur Autorisierung für Applikationen. Als Erfinder gelten Blaine Cook und Chris Messina. Aktuell liegt der Standard 1.0 vor. Die Spezifikationen sind offen und abrufbar unter http://oauth.net

7 Grobeentwurf eines Prototyps

Im nächsten Schritt wird eruiert wie Prototypen entworfen werden können. Mithilfe des Prototyps kann zukünftige Nutzung geprobt und optimiert werden. Üblicherweise sollte die Regel gelten, dass der Entwurf nicht für die Implementierung der Lösung verwendet wird. Die Praxis läuft meist anders und mündet mit RAD[60] in einem eigenen Paradigma. In jedem Fall muss der Entwurf visuell erfolgen, unter Berücksichtigung der speziellen Windows 8 Eigenschaften. Ein von einem Designer mit Photoshop gezeichnetes UI wird dem nicht gerecht.

7.1 Prototyping mit Expression Blend

Mit der Expression Blend Produktfamilie bietet Microsoft seit einigen Jahren design-orientierte Werkzeuge um HTML, Silverlight und WPF-Anwendungen zu entwerfen. Ein spezieller Projekttyp, der sich Sketchflow[61] nennt, zeichnet die Benutzerschnittstellen abstrakt. Im Feedback-Prozess mit dem User wird so die Diskussion um design-orientierte Nebensächlichkeiten vermieden.

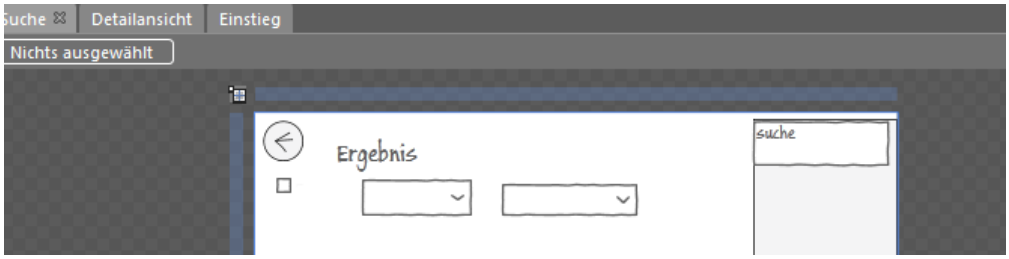


Abbildung 12 Expression Blend Sketchflow

Der Fokus liegt auf den wesentlichen Anwendungseigenschaften aus Sicht der User. Darüber hinaus kann mit Blend und Sketchflow auch der Programmfluss nachgebildet werden.

Alle Aktionselemente, wie Buttons, können Aktivitäten auslösen, wie die Navigation auf eine andere Seite. Selbst das Anzeigen von Dummy-Daten ist so möglich.

[60] Rapid Application Development für schnelle Anwendungsentwicklung. Ein von Barry Boehm entwickeltes Konzept zur Software-Entwicklung mit einem prototypischen Vorgehensmodell.

[61] Sketching ist im Industriedesign ein geläufiger Begriff, der die schnell erstelle Freihandskizze umschreibt.

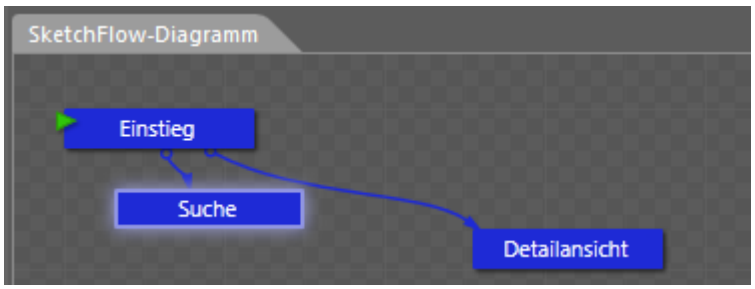


Abbildung 13 Expression Blend Programmfluss als Sketchflow

Letztendlich kann ein Sketchflow-Projekt sogar gepackt und an den Auftraggeber versendet werden, der wiederum die Anwendung probiert und mit Kommentaren versehen kann. Die fertige Sketchflow-Anwendung basiert auf XAML, so dass das User Interface ohne Aufwand übernommen werden kann.

In den ursprünglichen Planungen war vorgesehen, die Anwendung per Expression Blend und Sketchflow als Prototyp zu entwerfen. Leider fehlt diese Funktion bis heute für Windows 8 und es gibt keine Aussicht, dass dieser Zustand sich in den nächsten Monaten ändern wird. Ein Sketching auf eine alternative Zielplattform wie WPF erscheint wenig sinnvoll, da die Unterschiede zu groß sind.

7.2 Entwurf per Stift

Letztendlich erfolgt der Anwendungsentwurf rudimentär per Bleistift und Papier. Gezeichnet werden die Dialoge und die Abfolge im Sinne eines Storyboards. Während der Implementierungsphase erfolgt die Feinanpassung. Die Screenshots dazu finden sich im nachfolgenden Abschnitt 8, Implementierung.

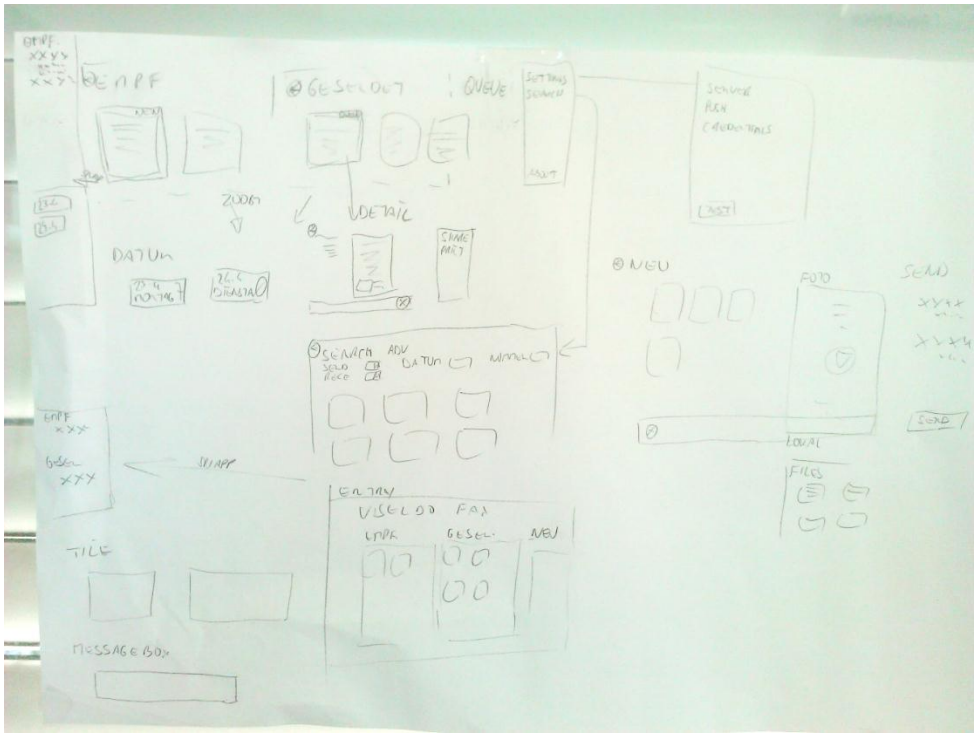


Abbildung 14 Entwurf mit Bleistift und Papier

8 Implementierung

Während der Planung wird versucht, anhand der funktionellen und nicht funktionellen Anforderungen einen Entwurf zu skizzieren, der Spezifika von Windows 8 berücksichtigt. Ziel dieser Vorgehensweise ist es, den Aufwand im Software-Projekte zu minimieren. Während der eigentlichen Realisierungsphase kommen laufend neue Erkenntnisse hinzu, die sich in zahllosen Änderungen niederschlagen. Eine Verwaltung derselben mit einem Verfahren für Change Request Management erzeugt überbordenden Aufwand.

Faktisch kristallisiert sich heraus, dass nur wesentliche Fortschritte erzielt werden können, wenn man viele und kleine Iterationen durchläuft. Erkenntnisse, die aus der agilen Entwicklung und deren Methoden, wie SCRUM[62] bestätigt werden. In der Praxis erscheint bei hochinnovativen Projekten mit wenigen Erfahrungswerten ein sehr kurzer Zyklus von wenigen Tagen oder gar Stunden vorteilhaft.

Im Nachgang betrachtet, bietet sich in dieser Ausgangslage auch der Einsatz von Pair Programmierung an, einer Methodik aus XP[63].

Die Beschreibung der Lösung erfolgt in zwei Teilen, der Benutzerschnittstelle und der Codelogik. Der Codierungsteil beschränkt sich wegen des Umfangs auf die wichtigsten Elemente. Für die Details wird das Studium des beiliegenden Visual Studio-Projektes[64] empfohlen.

Wesentliche Erkenntnisse, unabhängig von der Plattform und Technologie, ergeben sich aus dem kompletten Verzicht auf eine Datenbank. Da Windows 8 WinRT aktuell keine RDBMS API beinhaltet (vgl. 6.9), war es von Nöten, das Thema grundlegend zu hinterfragen. Die daraus resultierende und eigentlich natürlich erscheinende Betrachtung der Aufgabe Fax als Dokument bringt eine Reihe von Vorteilen.

Die Entscheidung, ein Fax als ein Dokument zu behandeln und auf getrennte Metadaten oder Datenbanken zu verzichten, wirkt sich deutlich in der Lösung aus. So sind die Dokumente unabhängig von der Anwendung nutzbar und transportabel. Selbst der Datei-Explorer zeigt die Dokumenteigenschaften immer komplett in der Bildansicht an. Sehr wichtig, aber leider auch mit Mehraufwand verbunden, ist die Entscheidung, den kompletten Faxauftrag in eine Datei zu packen. Das Grafikformat TIFF unterstützt Multipage. Je-

[62] Scrum Ken Schwaber u.a. Agiles Manifest

[63] XP Extreme Programming

[64] Download unter www.ppedv.de/diplomprojekt.zip und auf der beiliegenden CD ROM

des moderne Grafikprogramm kann so als alternativer Reader verwendet werden. Die Bindung an eine Plattform entfällt damit.

Wenn man die Lösung in die Cloud bringen möchte, lässt sich der Service leichter implementieren, da Scale Out-Szenarien mit relationalen Datenbanken schlecht harmonieren. Microsoft geht diesen Weg mit Azure Table Storage[65].

Ein Gutteil des Projektaufwands fliest in Aufgaben, die mit der ursprünglichen Zielsetzung wenig zu tun haben. So erweisen sich Grafikformate, wie auch das eben ausgeführte Multipage, als durchaus komplex. Zudem hat sich herausgestellt, dass im Bereich Dateizugriff ein Betriebssystem viele Funktionen bietet, die sich unter anderem auf Geschwindigkeit und Energieverbrauch auswirken. Passende und aktuelle Literatur dazu war nicht zu finden.

8.1 Benutzerschnittstelle

Beim Entwurf des User Interface wurde Wert darauf gelegt, bisherige Denkmuster fallen zu lassen. Ein erfahrender Benutzer bezieht seine Muster aber genau aus seiner geübten Praxis. Folglich erscheint es auch nicht zielführend, im Dialog mit dem Benutzer ein revolutionäres neues UI zu entwerfen.

Im Fokus steht immer die Überlegung nach dem Bedarf und den Erwartungen des Nutzers an das Programm zu einem bestimmten Zeitpunkt. Dies führt zum Teil zu interessanten, überraschenden Ansätzen. In anderen Darstellungen ist der banalste Lösungsweg letztendlich der bessere.

8.1.1 Startbildschirm

Neben dem Splash Screen sieht der Benutzer zuerst den eigentlichen Startbildschirm. Auf diesem befinden sich die drei wesentlichen Funktionen.

- Empfangene Faxe
- Gesendete Faxe
- Neues Fax

Windows Metro Apps erzeugen durch ein Hintergrundbild Emotionen. Passend zur Faxaufgabe wurde der Eingangsbereich der ppedv AG Berlin gewählt.

Für die Darstellung der Faxe wurde ein Gridview-Element mit acht Elementen, den Bildern der neuesten Faxe und einem Element für Alle Faxe definiert.

[65] http://www.windowsazure.com/en-us/home/features/data-management/

Abbildung 15 Startbildschirm Visendo Fax

Im Grid „gesendet“ wird durch ein Icon symbolisiert, dass das Fax noch in Bearbeitung ist. Das heißt, der Fax Service hat das Dokument bekommen, aber noch keine Sendebestätigung zurück übermittelt.

Letztendlich die häufigste Funktion eines Faxgerätes ist das Senden des Faxes. Die Aktionsschaltfläche liegt im Bereich des rechten Daumens. Durch die für den Anwender quasi fehlende Multitask-Fähigkeit (vgl. 6.3.1) von Windows 8 Apps muss der Entwickler davon ausgehen, dass das Erstellen eines Faxauftrages an jeder Stelle unterbrochen werden kann. Häufig wird ein ungeduldiger Anwender eine andere Anwendung nutzen und die aktuelle wird dadurch in den Suspend Modus versetzt oder gar vom Betriebssystem terminiert.

Nach Aktivierung der Metro App zeigt die Anwendung deswegen die Anzahl der Dokumente an, die für das Versenden vorgesehen sind.

Um den Darstellungsmodus Filled[66] zu untersützen, wurde der Aktionsraum mit den gesendeten und empfangenen Faxen samt des Buttons in ein Scrollview-Element gelegt. Dadurch kann der Benutzer vergrößern und verkleineren und horizontal scrollen.

[66] Vgl 6.10

Abbildung 16 Filled-Darstellung

Aus Benutzersicht wird eine Kommunikationsanwendung durchaus im Snapped-Modus betrieben. Interesse besteht dann eher an neuen Informationen. Entsprechend wurden für die Darstellung nur die empfangenen Faxe in verkürzter textueller Darstellung gewählt.

Abbildung 17 Snapped-Darstellung der empfangenen Faxe

Für die technische Umsetzung wird statt vom Page-Objekt von einer Klasse LayoutAwarePage geerbt. Diese ist im Projekt-Template enthalten und enthält die Logik für das Wechseln der ViewStates, also zwischen Filled, Snapped oder Full. In der XAML Page wird unter Zuhilfenahme des VisualStateManagers jeweils die Designsicht erstellt. Die Werkzeuge Visual Studio und Expression Blend enthalten visuelle Unterstützung dafür. In diesem konkreten Fall werden die Daten im Snapped Mode mit einem Listview-Element und im Full und Filled mit dem Gridview dargestellt. Darüber hinaus wird über Animationen, zum Beispiel die Größe und Position der Überschrift „Visendo Fax“ entsprechend des Views geändert.

8.1.2 Einstellungen Dialog

Die Einstellungen des Fax Clients werden vom Benutzer über die Windows 8 Charm Bar aktiviert. Der Charm selber ist nicht Bestandteil der App. Deswegen wird per Settings-

Command ein neuer Menüpunkt eingefügt. Da dies für gesamte Anwendung gelten soll, geschieht dies im Event OnSettingsRequested in der Anwendungsapplikationsklasse app.

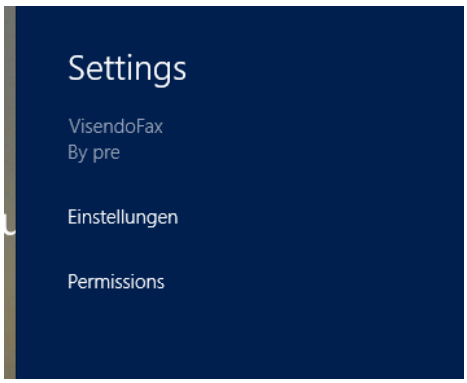


Abbildung 18 Settings Charm mit Visendo Fax Command

Um den Benutzer die Eingaben zu erleichtern, wurde eine automatische Konfiguration implementiert. Im Kern reicht es, die Mail-Adresse einzugeben. Im Hintergrund wird dann entsprechend der Domain ein Zugriff auf die Datei VisendoDiscover.xml auf dem zugehörigen Webserver durchgeführt.

Die Settings-Daten werden dann dauerhaft und lokal gespeichert.

Noch nicht implementiert wurde die Authentifizierung beim Service. Um den grundsätzlichen Konzepten und Ideen von OAuth zu folgen, wird diese später nicht mit Benutzername und Passwort durchgeführt. Der Vorteil ist, dass man einzelnen Endgeräten selektiv die Erlaubnis wieder entziehen kann.

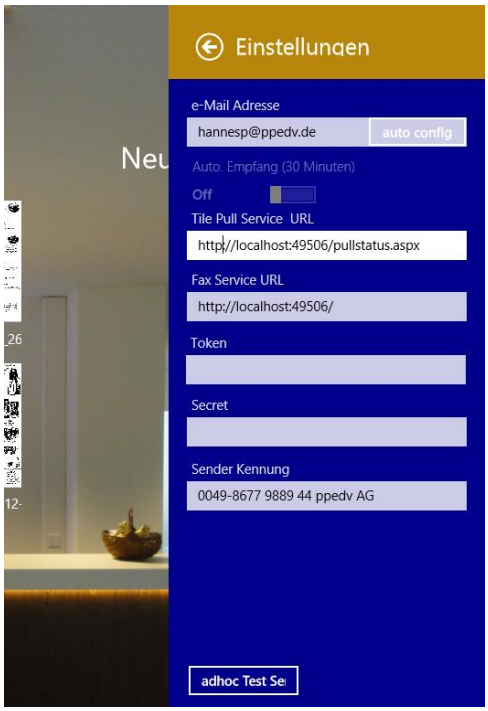


Abbildung 19 Settings-Dialog mit automatischer Konfiguration

8.1.3 Alle Faxe

Viele der Metro-Beispielanwendungen arbeiten nach dem Schema, Bilder in Grid darzustellen und bei Click auf Detailansicht zu wechseln. Mit der Navigation wird wieder zurück gewechselt.

Diese Darstellung ist grundsätzlich auch für Faxe sinnvoll. Hauptsächlicher Unterschied ist, dass Faxe als Dokumente in der Porträtansicht dargestellt werden müssen und damit nur zwei Reihen möglich sind.

Durch Wischgesten kann beliebig horizontal gescrollt werden. Es stellt sich die Detailfrage, wie die chronologische Darstellung erfolgen soll. Mögliche Muster können folgendermaßen aussehen:

Tabelle 4 Grid Darstellungsreihenfolge

1 2 3 4 5 6 7 8 9 10	1 3 5 7 9 2 4 6 8 10

Die Leserichtung in unserem Kulturkreis entspricht Variante eins. Aus technischen Gründen muss allerdings die Variante zwei gewählt werden um eine sinnvolle horizontal scrollbare Darstellung zu erreichen.

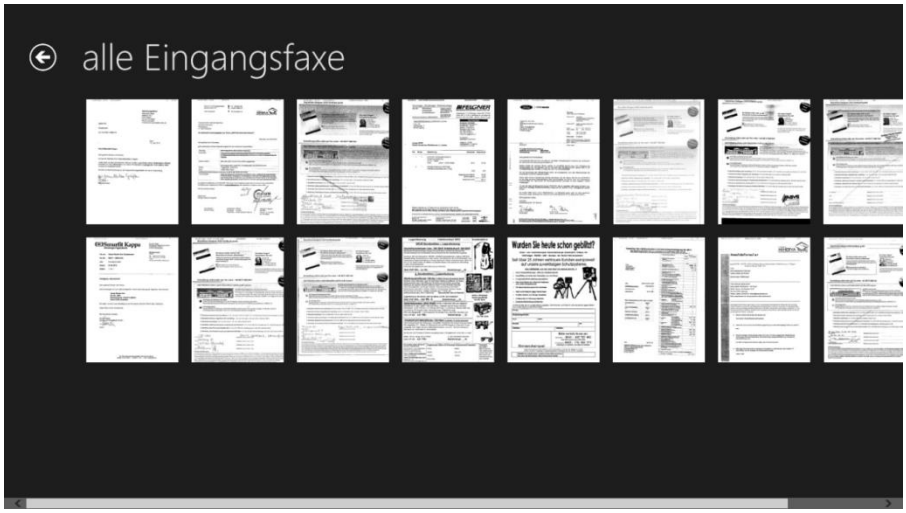


Abbildung 20 Alle Eingangsfaxe

Eine Frage zur Usability bleibt unbeantwortet. Wie verhält es sich, wenn große Mengen an Faxdokumenten vorhanden sind? Mögliche Szenarien sind Gruppierungen oder Sortierungen. Die Antwort darauf wird sich erst mit intensiver Nutzung der Anwendung finden lassen. Bis dorthin bleibt die Möglichkeit der Suche in den Dokumenten.

8.1.4 Faxsuche per Search Charm

Die Suche im Windows 8 Charm erlaubt Benutzern eine anwendungsunabhängige Suche. Der idente Ausdruck lässt sich in Anwendungen suchen, die den entsprechenden Search Contract implementiert haben, auch wenn diese nicht läuft.

Für die Darstellung wurde ein ähnlicher Ansatz gewählt wie für alle Faxe. Dazu gekommen ist eine Gruppierung für eingehende und ausgehende Faxe. Da die Dateinamen der Faxe die Faxnummer oder Senderkennung enthält, lässt sich so in diesen suchen. Das Konzept ist aber auch auf die Datei Attribute erweiterbar.

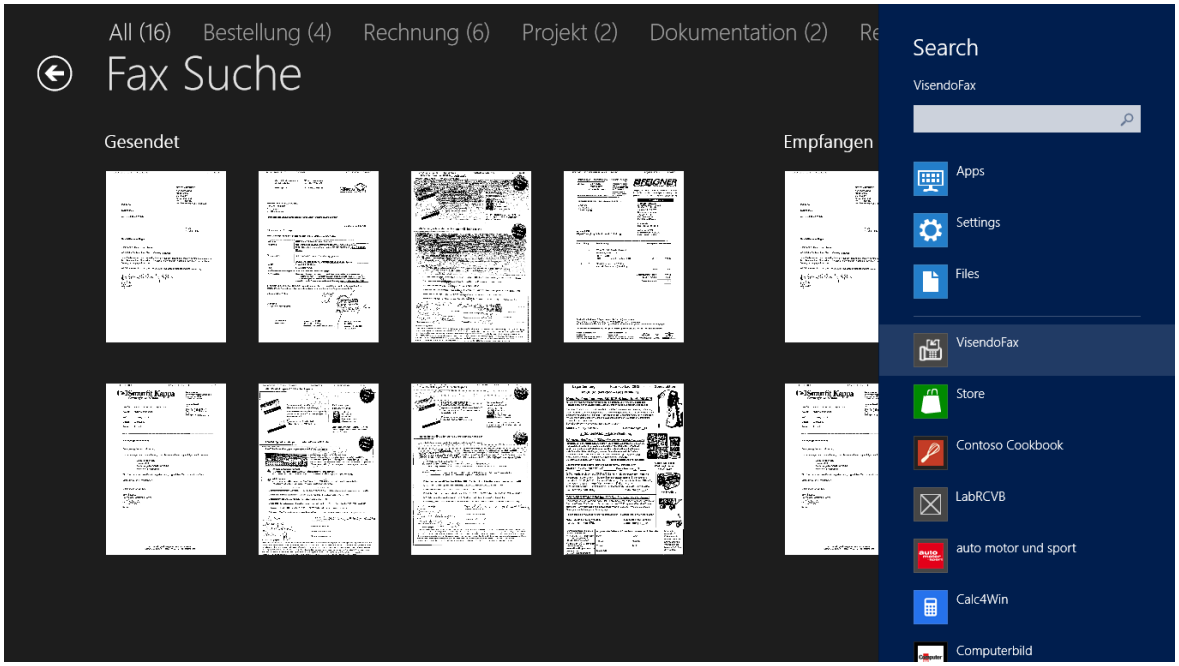


Abbildung 21 Suche aus dem Charm-Dialog

Zusätzlich wird am oberen Rand eine Leiste eingefügt, die dem Benutzer das Filtern der Ausdrücke erlaubt. Quelle ist das Tags-Attribut der Datei, im deutschsprachigen Windows Explorer als Markierungen bezeichnet. Eine Datei kann ein oder mehrere Tags haben.

Um den Benutzer schon sein voraussichtliches Ergebnis zu visualisieren, werden die Anzahl der Dokumente in Klammern angezeigt.

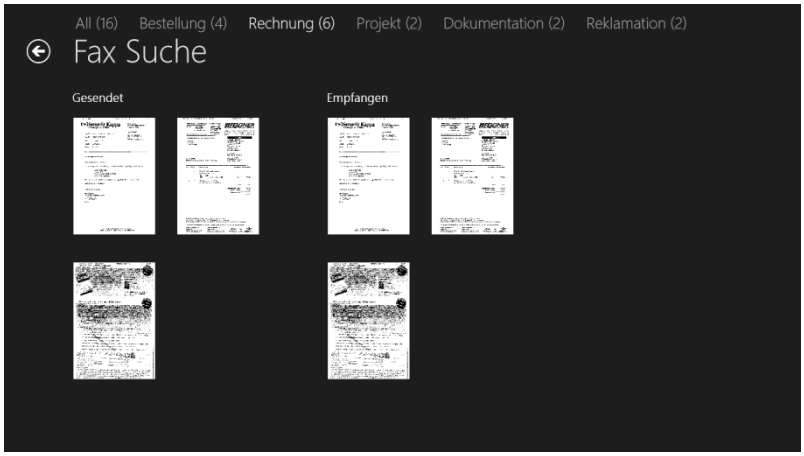


Abbildung 22 Filtern der Ergebnismenge

Auf eine Datenvisualisierung im Snapped View wurde verzichtet, da man davon ausgehen kann, dass dies bei einer Suche des Nutzers in möglichst großer Darstellung passiert.

8.1.5 Fax Detailansicht

In der Detaildarstellung eines Faxes gibt es zwei Anforderungen. Zum einem will der Benutzer wissen, welches Dokument ihm vorliegt und zum anderen Details zum eigentlichen Inhalt.

Nachdem der Benutzer ein Faxdokument auswählt, werden ihm das Fax und die Metadaten angezeigt. Wie sich in der Abbildung erkennen lässt, werden manche Metadaten nicht korrekt angezeigt. Der Grund liegt darin, dass WinRT und .NET keine identen Datentypen besitzen. Ein Mechanismus, der Projektion genannt wird, übersetzt diese. Offensichtlich ist dieser aber nicht komplett implementiert. Um dies zu demonstrieren, wurde der Code so belassen. Letztendlich könnte in der finalen Version dies behoben sein. Als Workaround bietet sich eine Converter-Klasse an, mit der das Darstellungsproblem auch jetzt gelöst werden kann.

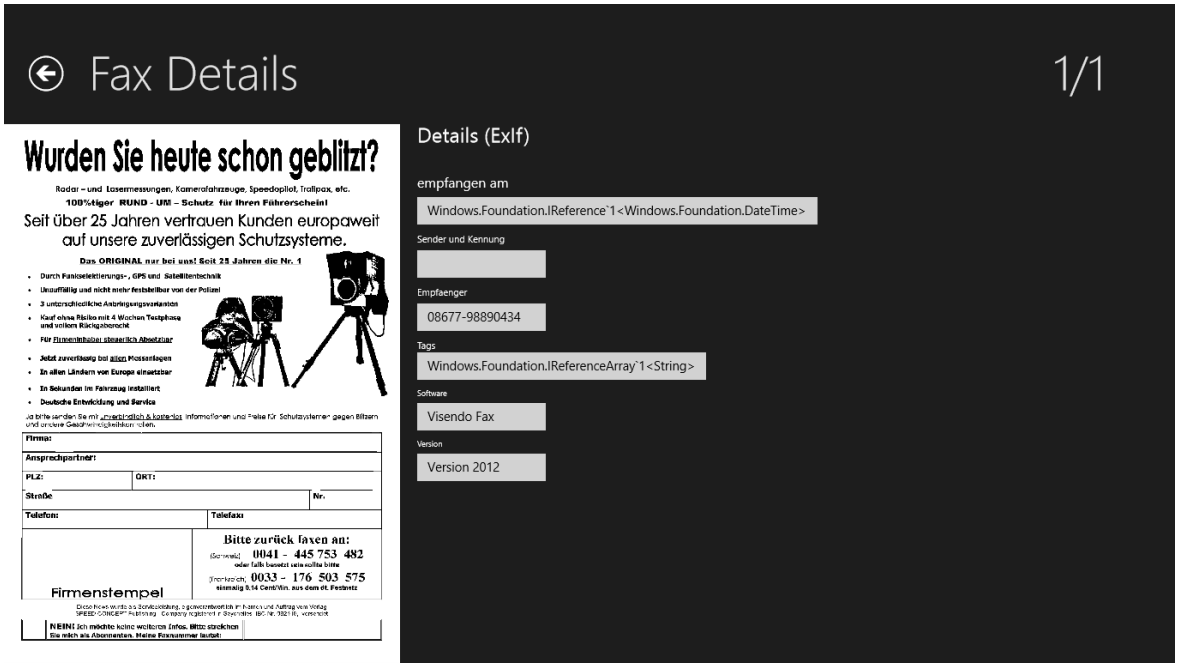


Abbildung 23 Fax Detailansicht mit Metadaten

Es wurde hier ein Grid Layout gewählt um das Dokument in allen möglichen Sichten passend zu präsentieren.

Wenn der Benutzer den Inhalt des Dokumentes lesen möchte, kann er das Tablet drehen in den Porträt-Modus und das Dokument nimmt den vollen Raum ein. Da die Formate von A4 und Windows 8 nicht übereinstimmen, bedarf es eines Kompromisses bezüglich Höhe oder Breite.

Die Anzeige der Attribute wurde mit Textboxen, statt Textblöcken umgesetzt, um dem Benutzer ein Copy Paste per Zwischenablage zu erlauben.

Abbildung 24 Fax Detail in Porträt-Darstellung

Ein interessantes Lösungsdetail findet sich darin, dass die Anwendung das Dokument schließt und in die vorige Übersichtsdarstellung (z.B.. Alle Faxe) zurückwechselt, wenn der Benutzer das Tablet in die Horizontale zurückdreht. Damit wird also die „zurück“-Funktion durch den Bewegungssensor ausgelöst.

Da Faxe mehrere Seiten umfassen können, kann der Benutzer mit einer Wischgeste durch diese Blättern. Durch die Anzeige der Blattnummer rechts oben ist die Position zu erkennen. Da die Navigation innerhalb des Dokumentes, sozusagen in die Tiefe, erfolgt, wurde als Gestenrichtung Oben/Unten gewählt.

Das dafür verwendete Steuerelement Scrollviewer, kann mittels Snappoints die Navigation in einer animierten Darstellung bis zum nächsten Bild vollenden. Selbst das Zoomen mit zwei Fingern ist möglich.

Andere Varianten, wie manuelles Erkennen des Events oder entsprechende Buttons wurden nach reiflichen Überlegungen verworfen.

8.1.6 Dokument an App weitergeben

Mit dem Ansatz, Fax als Dokument im Dateisystem zu speichern und zu verwalten, ergeben sich auch neue Perspektiven. So lässt sich über den DataTransferManager das Fax Dokument an Metro Apps, die den Share Target Contract mit dem passenden Type implementiert haben, weitergeben. In der folgenden Abbildung wählte der Benutzer die Mail App aus.

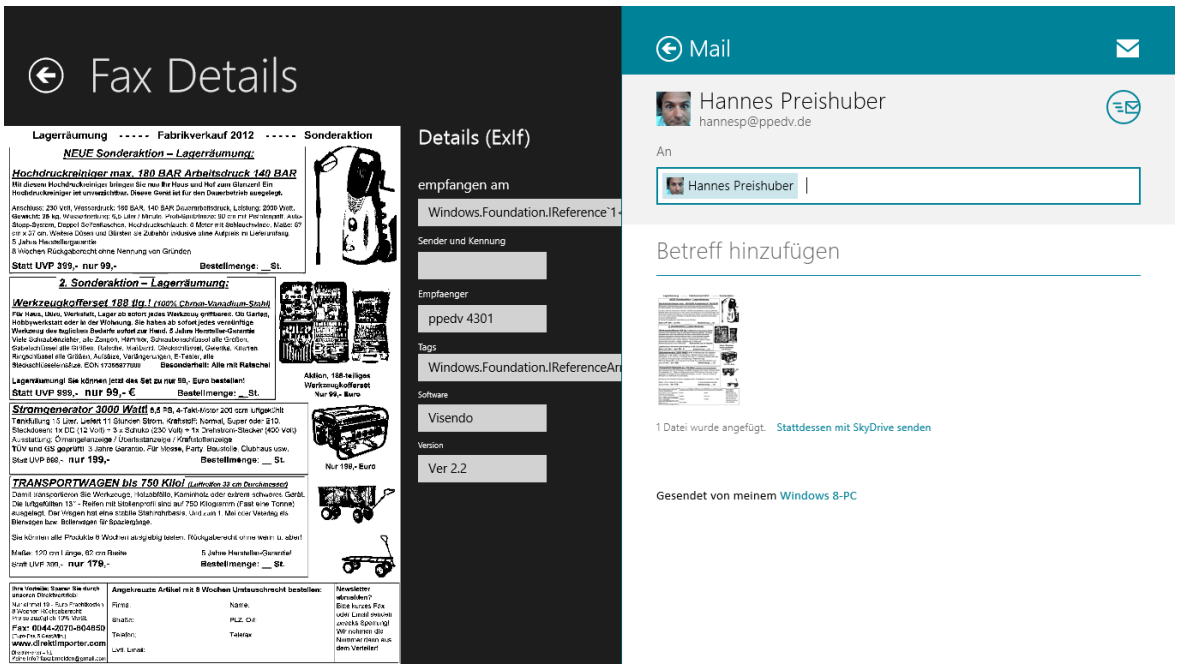


Abbildung 25 Fax als Mail versenden

8.1.7 Neuen Fax Job erstellen

Zentrale Funktion eines Fax Clients ist das Senden. Bisher arbeitet derNutzer des Visendo Fax Server meist mit einem Druckertreiber um aus jeder Anwendung senden zu können.

Mit Windows 8 ergeben sich auch hier neue Möglichkeiten. So kann der Benutzer aus einer Anwendung für Bilderverwaltung per Share dieses Bild an die Faxanwendung und deren Sende Job-Queue übermitteln.

Die Faxanwendung muss als Capability den Share Contract implementieren. Dazu braucht diese eine spezielle XAML-Seite, die für den Share-Dialog zur Verfügung gestellt wird. Das Besondere ist, dass der Benutzer danach wieder in seine Ausgangsanwendung zurückkommt und nicht dauerhaft in der Zielanwendung arbeitet. Entsprechend ist der Workflow zu gestalten.

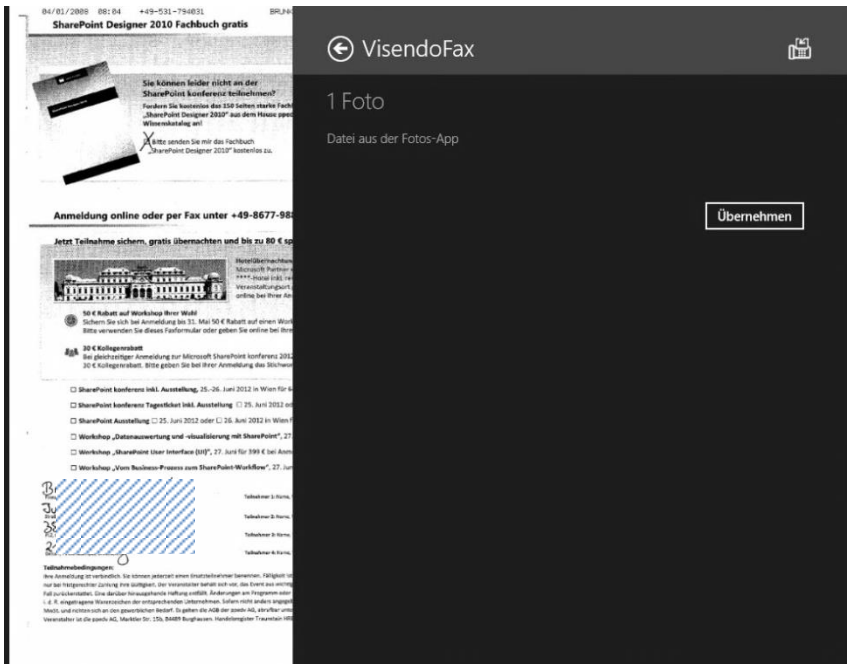


Abbildung 26 Bild Share to Fax

Wenn der Benutzer nun später die Faxanwendung öffnet, ist der Fax Sende-Counter auf der Startseite auf 1 beziehungsweise auf dem Wert, der in der Warteschlange liegt.

Der eigentliche Teil in der Anwendung für das Senden umfasst drei Funktionen.

- Bild per Webcam erstellen
- Bild aus Datei hinzufügen
- Emfpänger auswählen und senden

Diese drei Punkte finden sich in der unteren Appbar. Dabei ist die Aktion Senden links separat platziert um mögliche Fehlbedienungen zu reduzieren. Diese Kommandos sind dauerhaft sichtbar und damit ist die Appbar mit dem Attribut isSticky fixiert.

Die obere Appbar enthält eine Liste der aktuellen Dokumente für diesen Fax-Job - mit 50% Transparenz, damit diese den Inhalt des Foto Preview nicht verdecken. Darüber hinaus verschwindet die obere Appbar, wenn der Benutzer auf den Bildschirm tippt. Mit der Wischgeste vom oberen Rand nach unten erscheint sie wieder.

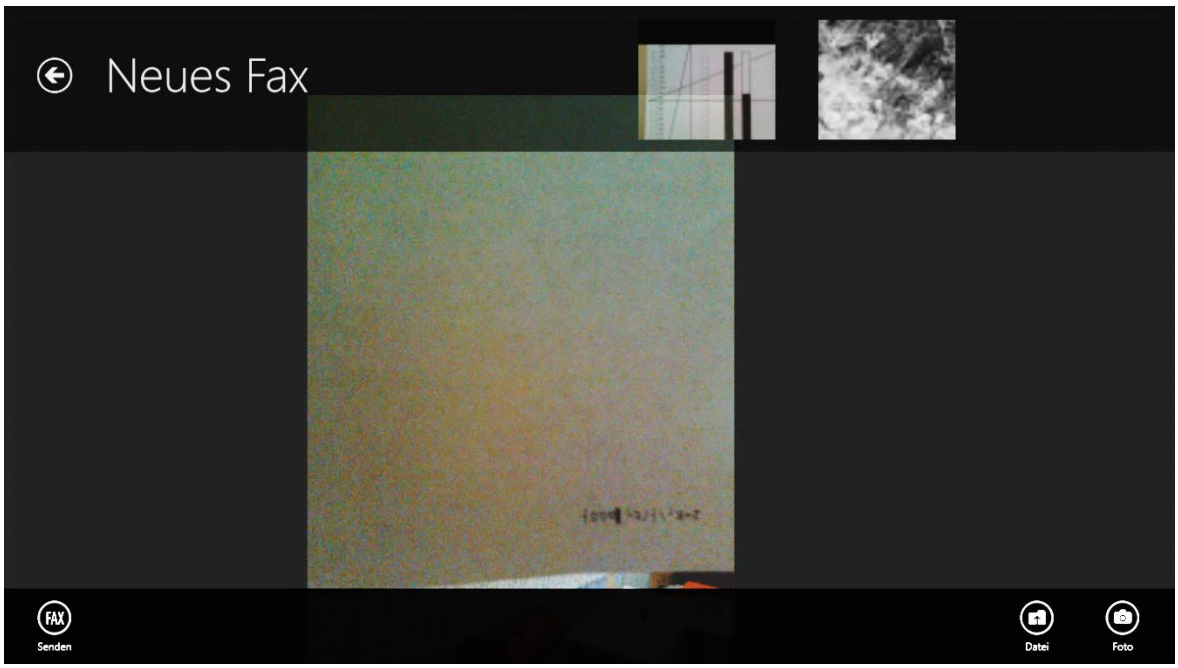


Abbildung 27 Fax Sende-Hauptdialog

Es hat sich in dem Test als problematisch herausgestellt, mit den Grafikeigenschaften der Kamera zu arbeiten. Ursprünglich waren im Dialog noch Schieberegler für Helligkeit und Kontrast vorhanden. Auch die Auflösung und Ausrichtung des Bildes waren einstellbar. Hier möchte der Autor abwarten, wie in der finalen Windows 8-Version der Kamerastandardsettings-Dialog ausgelegt ist.

Das fertige Bild des Dokuments wird in einem weiteren Dialogfenster angezeigt, vom Benutzer durch Drücken des grünen Bereiches bestätigt und in die Queue übernommen. Anschließend wird zum vorigen Dialog zurücknavigiert.

Abbildung 28 Bestätigen des Faxdokumentes

Obwohl die Funktion ähnlich einem OK-Cancel-Dialog sehr einfach ist, wurde sehr viel Zeit in diesem Dialog und Workflow investiert. Offene Fragen bleiben zur Benutzung der Kontrollansicht im Porträt-Modus und einer optionalen Symbolik für OK und Cancel.

Durch den System Fileopen-Dialog ist die Option vorgegeben, eine Bilddatei hinzuzufügen. Entsprechend dem Windows 8 UI Patterns erfolgt dies Chromeless[67] und damit in Vollbildansicht.

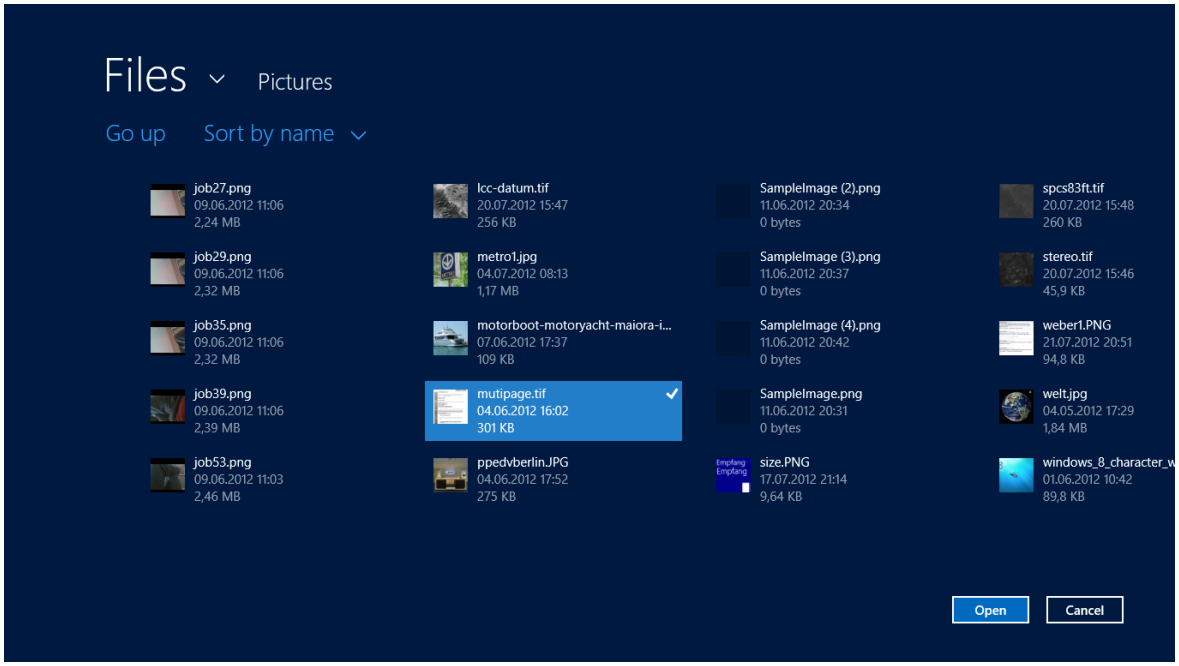


Abbildung 29 Windows 8 File-Dialog

Letztendlich wird der Benutzer auf Senden klicken und den oder die Benutzer auswählen, die das Fax erhalten sollen. Entsprechend kommt der Contact Picker zum Einsatz, der zunächst alle Anwendungen auflistet, die den Contact Contract als solchen implementiert haben. In diesem Beispiel wird die Windows 8 Standard Contact-Anwendung ausgewählt.

[67] Vgl. 6.3.1

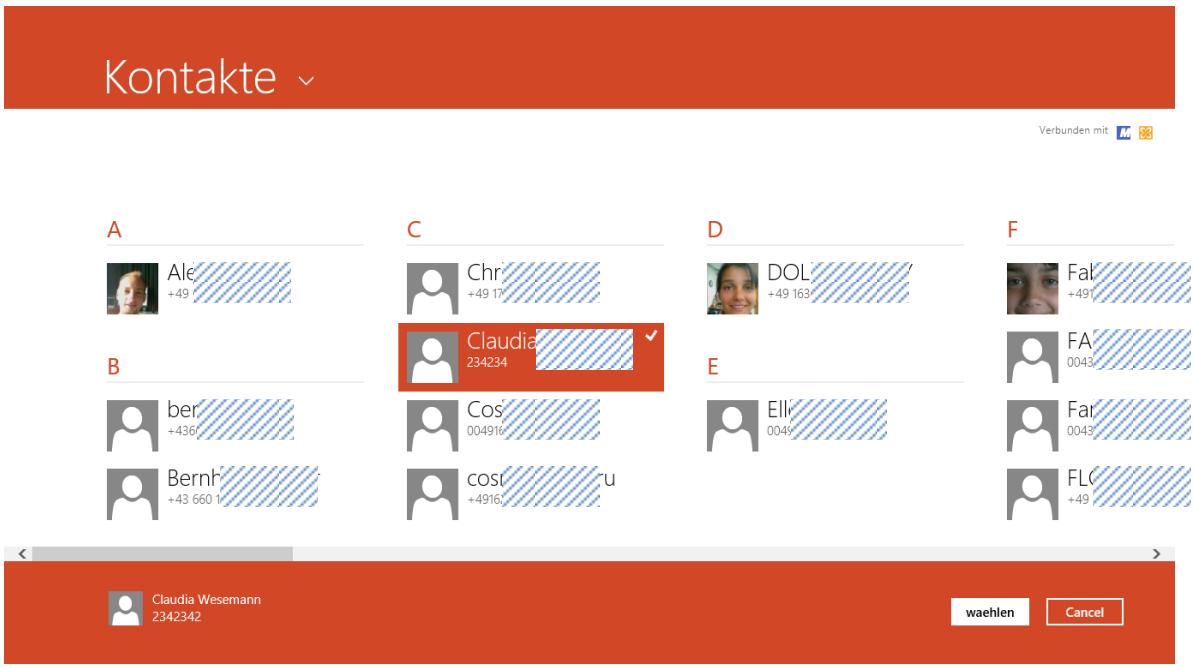


Abbildung 30 Windows 8 Contact Application

Obwohl dieser Contact Manager einige Limitierungen aufweist, wird er verwendet um doppelte Datenpflege zu vermeiden. Konkret kann man nicht vorfiltern nach Kontakten, die auch eine Faxnummer hinterlegt haben. Eine weitere Funktion wäre die Implementierung einer fax:// -Protokollerweiterung, die es erlauben würde aus einem Kontakt ein Fax zu verschicken. Da dies für tel:// funktioniert, um einen Anruf zu initiieren, ist zu erwarten, dass es auch diesbezüglich von Seiten Microsofts noch Änderungen geben wird.

Letztendlich erhält der Benutzer einen Standard Message-Dialog, um die Aufnahme in die Sendequeue des Services zu bestätigen.

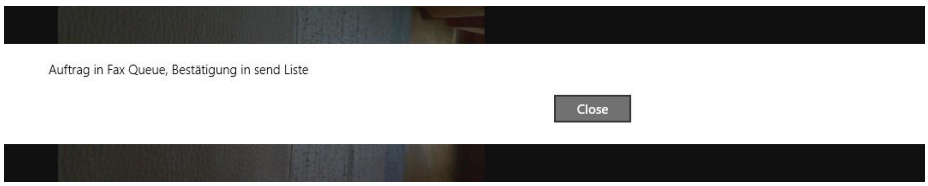


Abbildung 31 Bestätigungsdialog

Dieser Dialog bestätigt nicht den eigentlichen Versand des Dokumentes durch den Faxserver. In der Regel ist der Benutzer auch eher am erfolgreichen Empfang des Dokumentes durch die Gegenstelle interessiert. Die Kommunikation dazu muss Windows 8-typisch asynchron erfolgen.

8.2 Service-Schicht

Schon früh war klar, dass ein SOAP-basierter Ansatz, wie in WCF verfolgt, nicht wirklich passend ist. Argumente gegen SOAP waren die Nachrichtengröße und das Format, die mangelnde Flexibilität und vor allem der RPC-Ansatz, der nicht der Aufgabenstellung entspricht. Folglich wird ein REST-basierter Ansatz verfolgt, der die üblichen http-

Kommandos GET, POST, PUT und DELTE nutzt. Darüber hinaus ist gerade für Bilder Caching eine performance-steigernde Funktionalität. Im Rahmen der fortschreitenden Veröffentlichungszyklen erweist sich der vorhanden WCF Service als unpassend.

Wenn man den Gedanken verfolgt, dass Faxe Dokumente sind, sollte man diese in jeder Hinsicht so behandeln. Metainformation wird direkt im Dokument abgelegt und nicht in separaten Beschreibungsdaten. Der Vorteil ist, dass das Dokument losgelöst von der Anwendung transportiert und behandelt werden kann.

Grafikformate bieten diese Information schon lange an. Es gibt Standardfelder und erweiterte Formate, wie Exif[68].

Folglich muss der Service schon beim Erstellen der Faxdokumente diese mit den nötigen Attributen versehen. Das tut der bisherige Service nicht.

Letztendlich wurde im Dialog mit den Entwicklern des Visendo Fax Service entschieden, dass eine Prototyp Implementierung basierend auf Pseudodaten erfolgt, weil es schlichtweg in der kurzen Zeit nicht möglich war, das Problem anders zu lösen. In einem späteren Stadium wird das Visendo Team den Service nachimplementieren.

8.2.1 Formate und Vereinbarungen

Da die existierende Service-Architektur des Visendo Fax Servers auf RPC basiert, ist es wirtschaftlich nicht ratsam diesen für dieses Projekt neu zu implementieren. In Zusammenarbeit mit den Entwicklern wurde das Model dazu entworfen. Die dabei erstellten Vereinbarungen müssen in der Anwendungspraxis noch nachgebessert werden.

Als Grafikformat wird TIFF festgelegt. Dies erzeugt sehr kleine Dateien für Anwendungstypische Schwarz/Weiß Dokumente und unterstützt Multipage. Die Kommunikation mit dem Web Service erfolgt soweit sinnvoll per HTTP-Upload und -Download. Zusätzlich existieren spezifische Service Calls, die strukturierte Daten übermitteln.

Mit der Autodiscover-Funktion wird eine XML-Datei vom Webserver geladen, die dem Benutzer Konfigurationshilfen bietet.

```
<?xml version="1.0" encoding="utf-8" ?>
<Autodiscover>
  <Server>http://localhost:49506/</Server>
  <Url>http://localhost:49506/pullstatus.aspx</Url>
  <Account>0049-8677 9889 44 ppedv AG</Account>
</Autodiscover>
```

Listing 14 VisendoDiscover.xml

Die TIFF-Dateien werden in einer Auflösung von 1728x2276 erstellt und gelesen. Die Metainformationen werden entsprechend Exif-Format wie folgt zugeordnet:

[68] Exif - Exchangeable Image File Format ist ein Standard der JEITA.http://www.jeita.or.jp/english/

Tabelle 5 Verwendete Exif Attribute

Zweck	Property	Hinweis
Hersteller	System.Photo.CameraManufacturer	
Version	System.Photo.CameraModel	
Sender, Kennung	System.Author	
Empfänger	System.Copyright	
Datum	System.Photo.DateTaken	Datum, Uhrzeit, Empfang
Subject	System.Subject	empfangen oder gesendet
Tags	System.Keywords	

8.2.2 Service-Vereinbarungen

Für den Abruf auf neue Faxe wird ein Service-Aufruf erzeugt, der direkt den passenden XML-Code für die Live Tiles zurück liefert. Diese Calls können von Windows zum Beispiel in Abständen von 30 Minutenim Hintergrund ausgeführt werden ohne dass die Anwendung laufen muss. Das geht vergleichsweise einfach mit dem TileupdateManager.

```
TileUpdateManager.CreateTileUpdaterForApplication().StartPeriodicUpdateBatch({New
Uri("http://localhost:55108/tile1.aspx")}, PeriodicUpdateRecurrence.HalfHour)
```

Listing 15 TileUpdateManager starten

Dieser TileupdateManager läuft allerdings nur, wenn das Tablet an einer Stromversorgung hängt und auch mit einer minimalen Zeitspanne von 30 Minuten. Alternativ kann man eine Backgroundtask erstellen. Mit erheblichem Aufwand und auch nur, wenn der Benutzer das als LockedScreen Anwendung erlaubt, läuft dann minimal alle 15 Minuten eine Task,allerdings mit weiteren Einschränkungen.

Wenn man umgehend von eintreffenden Faxen informiert werden muss, bleibt nur das Windows Push Notification-System. Dies basiert auf Windows Azure und wurde in der Anwendung nicht fertig implementiert, weil aktuell zuverlässige Testmöglichkeiten fehlen.

Wenn die Kachel neue Faxe anzeigt, heißt das noch lange nicht, dass diese auch lokal vorhanden sind. Entsprechend muss beim Start der Faxanwendung ein Service Call initiiert werden, der zurück liefert

- welche Faxe vom Service versendet wurden.
- welche Faxe neu angekommen sind.

Als Serialisierungsformat wird wegen seiner kompakten Form JSON verwendet.

Der eigentliche Download geschieht mit der BackgroundDownloader-Klasse. TIFF Dateien haben zwar ein kompaktes Format, aber auch hier besteht die Möglichkeit, dass der Download abbricht. Der Background Downloader kümmert sich dann anwendungsautark um den Download und speichert diesen in das gewählte Verzeichnis.

Als Anwendungsverzeichnis für alle Faxbilder wird per Konvention der Ordner Pictures Library definiert und dort das Unterverzeichnis VisendoFax, also zum Beispiel. C:\Users\pre\Pictures\VisendoFax. Darin enthalten sind die Ordner Ausgang und Eingang.

Für die Fax-Queue wird der Ordner Temp definiert (C:\Users\pre\AppData\Local\Packages\81cf6aee-d9da-45dd-a9d5-9dc29d43d8af_89gf582k2a27c).

Generell folgt der Anwendungsentwurf dem Paradigma „Konvention vor Konfiguration“, weil dadurch die Fehlersuche vereinfacht wird.

Für den Upload eines neu erzeugten Faxes wird auf dem IIS Web Server ein ASHX Handler aus ASP.NET[69] verwendet.

8.3 Logik-Implementierung

Als Programmiersprache kommt VB.NET in Kombination mit XAML zum Einsatz. Der komplette Quellcode findet sich als Visual Studio 2012-Projekt auf der beiliegenden CD ROM.

Die folgenden Listings beschränken sich auf die jeweilig relevanten Stellen unter Berücksichtigung voriger Ausführungen.

8.3.1 Dateien Binden an Gridview

Um Daten deklarativ binden zu können, benötigt XAML eine generische Liste, die dem Itemssource-Attribut zugewiesen wird. Um diese Liste jederzeit aktualisierbar zu halten, muss das Interface InotitfyPropertychanged implementiert werden. Da der Zugriff auf das Dateisystem, in dem die Faxe liegen, asynchron erfolgen muss, wird die Liste erst gefüllt, nachdem das User Interface fertig gerendert ist. Mit anderen Worten, der Dialog wird aktualisiert, wenn sich die Daten geändert haben. Der Typ ObservableCollection erfüllt diese Eigenschaften.

```
Public Async Function LoadSend() As Task(Of ObservableCollection(Of FaxDocument))
        Dim pics = Await
Windows.Storage.KnownFolders.PicturesLibrary.GetFolderAsync("VisendoFax\Ausgang")
        Dim liste As New ObservableCollection(Of FaxDocument)
        Dim files As IReadOnlyList(Of StorageFile) = Await
pics.GetFilesAsync(CommonFileQuery.OrderByDate, 0, 8)
        For Each f In files
            Dim d As New FaxDocument
            Dim ra = Await f.OpenAsync(FileAccessMode.Read)
            Dim bmp = New BitmapImage()
            bmp.SetSource(ra)
            d.bild = bmp
            d.SenderKennung = f.Name
            d.name = f.Name
```

[69] Der Internet Information Server IIS ist mit .NET programmierbar. Die Kombination aus HTML, Steuerelementen und Server Code wird von Microsoft als ASP.NET bezeichnet.

```
            liste.Add(d)
        Next
 'Alles anzeigen Element
        Dim n As New FaxDocument
        n.SenderKennung = "...ALLE..."
        liste.Add(n)
        Return liste
End Function
```

Listing 16 Typisierte Liste füllen

Üblicherweise wird ein Basistyp für die Listeneinträge erstellt, der alle nötigen Eigenschaften beinhaltet. In diesem Beispiel FaxDocument. Alternativ ist es auch denkbar, eine anonyme Liste zu erstellen. Mit dieser gibt es allerdings erfahrungsgemäß manchmal Probleme beim Binden der Daten. Im folgenden Beispiel ist nur das Property IsSent vollständig auskodiert. Die übrigen Properties müssen dieser Logik analog folgen. Die Methode SetProperty synchronisiert die Daten mit dem User Interface. Dabei handelt es sich nur um eine Abstraktion des Events INotifyPropertychanged aus der BindableBase-Klasse, die sich in jedem Metro-Projekt-Template findet.

```
Public Class FaxDocument
    Inherits BindableBase
    Private _isSent As Boolean
    Public Property isSent() As Boolean
        Get
            Return _isSent
        End Get
        Set(ByVal value As Boolean)
            SetProperty(_isSent, value)
        End Set
    End Property
    Public Property Datum() As Date
    Public Property SenderKennung() As String
    Public Property Tags() As String
    Public Property Empfaenger() As String
    Public Property faxHersteller() As String
    Public Property Version() As String
    Public Property bild() As BitmapImage
    Public Property name() As String
End Class
```

Listing 17 Klassendefinition für Fax Objekt Container

Die Visualisierung wird später deklarativ im XAML-Teil durchgeführt. Basis für die Listendarstellung ist das GridView-Steuerelement (grdSent), das per Code an das grdSent gebunden wurde. Weil das gebundene Objekt komplexer Natur ist, benötigt das Gridview detaillierte Anweisungen, um jedes einzelne Item wie gewünscht darstellen zu können. Das geschieht im Datatemplate. Dort wird. ein Image-Steuerelement mit der Bildquelle aus dem faxDocument-Objekt mit Hilfe des Schlüsselwortes Binding verknüpft. Außerdem wird der Bildname im Tag Attribut hinterlegt. Für die Anzeige der Sanduhr für ein Dokument, das sich noch im Sendevorgang befindet, wird ein Textblock verwendet, dessen Sichtbarkeit über den boolschen Wert isSent gesteuert wird. Da allerdings die gültigen Werte für Visibilty nicht true oder false sind, wird ein Converter zwischengeschaltet. Con-

verter sind ein Standardconcept in XAML und im wesentlich eine Klasse, die das Interface IConvertible implementiert.

```
<GridView x:Name="gridSent" Grid.Column="1" HorizontalAlignment="Left" Mar-
gin="31,1,0,0" Grid.Row="1" VerticalAlignment="Top" Width="365" Height="550"
Grid.ColumnSpan="2">
   <GridView.ItemTemplate>
       <DataTemplate>
          <StackPanel >
             <Grid Width="100" Height="150">
              <Image Width="100" Height="150" Source="{Binding bild}" Tag="{Binding
name}"/>
             <TextBlock Text="⌛"  FontSize="20"
Visibility="{Binding isSent, Converter={StaticResource BooleanToVisibilityConvert-
er}}" Foreground="Red" HorizontalAlignment="Right"  VerticalAlignment="Bottom"/>
             </Grid>
             <TextBlock Text="{Binding SenderKennung}" Width="100" />
          </StackPanel>
       </DataTemplate>
 </GridView.ItemTemplate>
</GridView>
```

Listing 18 XAML Template für Fax Liste

8.3.2 Settings

Wenn der Benutzer den Settings Charm öffnen will, bekommt die aktuelle Anwendung ein Event und damit die Chance sich in den Dialog einzuklinken. In der app.xml.vb wird dann ein oder mehrere Commands in den seitlichen Charme eingefügt und diesem ein Callback Event (hier Settingsfunktion) übergeben.

```
Protected Sub OnSettingsRequest(sender As SettingsPane, args As SettingsPaneCom-
mandsRequestedEventArgs)
        Dim h As New SettingsCommand("SetupID", "Einstellungen", New UICommandIn-
vokedHandler(AddressOf SettingsFunktion))
        args.Request.ApplicationCommands.Add(h)
End Sub
```

Der eigentliche Settings-Dialog der Anwendung wird dann extra gehandhabt. In der Praxis ist es am einfachsten den Dialog als User Control zu erstellen, weil dann der visuelle Designer genutzt werden kann. Alternativ kann man den Dialog auch komplett per Code erstellen. Wenn der Benutzer neben den Settings-Dialog klickt, soll dieser wieder verschwinden. Diese Funktion bietet nur das Pop-Up-Steuerelement mit dem Attribut isLightdismissEnabled. Deswegen wird dieses als Containercontrol verwendet und der Settings-Dialog dort reingeladen. Die Standardweite des Settingsdialog ist 346 virtuelle Pixel.

```
Private Sub SettingsFunktion()
  Dim p As New Popup
  p.IsLightDismissEnabled = True
  p.Width = 346
  p.Height = Window.Current.Bounds.Height
  p.Child = New Settings1 With {.Height = Window.Current.Bounds.Height}
  p.SetValue(Canvas.LeftProperty, Window.Current.Bounds.Width - 346)
  p.SetValue(Canvas.TopProperty, 0)
  p.Margin = New Thickness(0)
  p.IsOpen = True
```

```
End Sub
```

Listing 19 Settings Dialog instanzieren und einblenden

Wesentliches Merkmal einfacher User Interfaces ist, dass Benutzer möglichst wenig Fehler machen können. Deswegen ist es hier nötig, dass die Einstellungen automatisch gesichert werden und im Bedarf bei Neustart wiederhergestellt werden.

Die Settings-Daten werden in einem eigenen Objekt gehalten. Das Metro-Projekt-Template liefert eine vordefinierte Klasse SuspensionManager. Dieser kann der Entwickler Objekte per Key übergeben und lesen lassen. Diese Klasse kümmert sich auch um das Speichern der Werte in der Datei _SessionState.xml. Allerdings schlägt die Serialisierung für ein komplexes Objekt, wie hier die Settings, fehl.

Die in WinRT vorgesehene Methode dafür ist, eine Composite-Gruppe für das Objekt zu erstellen und Eigenschaft für Eigenschaft dort als Schlüssel abzulegen. Unter Zuhilfenahme von .NET Reflection lässt sich der nötige Code verkürzen und flexibilisieren. Folgender Code wird aufgerufen, wenn der Settings-Dialog entladen wird.

```
Private Sub Settings1_Unloaded(sender As Object, e As RoutedEventArgs) Handles
Me.Unloaded
   Dim composite As New Windows.Storage.ApplicationDataCompositeValue
   Dim properties As PropertyInfo() = GetType(SettingsModel).GetRuntimeProperties()
   For Each pi As PropertyInfo In properties
     If TypeOf pi.GetValue(einstellungen) Is System.Uri Then
        composite(pi.Name) = CType(pi.GetValue(einstellungen), System.Uri).ToString
     Else
        composite(pi.Name) = pi.GetValue(einstellungen)
     End If
   Next
  localSettings.Values("settings") = composite
End Sub
```

Listing 20 Settings-Eigenschaften in Composite-Gruppe speichern

8.3.3 Alle Faxe virtuell verwalten

Es wurde bereits beschrieben wie neue Faxdokumente als Liste angezeigt werden können. Was passiert aber, wenn nun hunderte oder gar tausende Dokument im Verzeichnis liegen? Um diesem Gesichtspunkt zu begegnen, gibt es zwei Ansatzpunkte. Der eine liegt im Rendering des User Interface. Sozusagen out-of-the-box wird ein Gridview nämlich nur den sichtbaren Bereich rendern. Dies verdankt das Control seinem internen Template, das ein VirtualizingStackPanel verwendet. Man kann dieses Verhalten im Bedarfsfall auch ändern.

Der zweite Teil kümmert sich um die Daten und das Filesystem direkt. Auch eine Liste mit tausenden von binären Bilddaten wird den Arbeitsspeicher, die Festplatte und den Prozessor fordern, was sich negativ auf die Akkulaufzeit auswirkt. Um dieses Problem zu umgehen, kann man die FileinformationFactory verwenden und dann nur den Virtualized-Vector zuweisen.

```
Protected Overrides Async Sub LoadState(navigationParameter As Object, pageState As
Dictionary(Of String, Object))
  Dim ft = {".tif"}
  Dim pics = Await
Windows.Storage.KnownFolders.PicturesLibrary.GetFolderAsync("VisendoFax\Eingang")
  Dim qo = New QueryOptions(CommonFileQuery.OrderByDate, ft)
  qo.FolderDepth = FolderDepth.Shallow
  Dim fg = pics.CreateFileQueryWithOptions(qo)
  Dim fi = New FileInformationFactory(fg, ThumbnailMode.SingleItem, 190, Thumb-
nailOptions.ResizeThumbnail, True)
  itemGridView.ItemsSource = fi.GetVirtualizedFilesVector
  itemListView.ItemsSource = fi.GetVirtualizedFilesVector
End Sub
```

Listing 21 Dateizugriff mit der FileInformationFactory

In diesem Beispiel ist auch die Möglichkeit erkennbar, Queryoptions zu verwenden, um die Menge der gefundenen Dateien zu reduzieren. Diese Queryoptions lassen sich auch auf den Index des Filesystems anwenden, statt auf eine Volltextsuche. Damit reduziert sich der Dateizugriff erheblich und die Reaktionszeit wird schneller.

Der Vollständigkeit halber ist noch der recht einfache Trick erklärt, mit dem die Detailansicht geöffnet wird, wenn der Benutzer das Bild anklickt. Zunächst fällt auf, dass es kein Maus-Event ist, da Windows 8 nur mehr Touch Events kennt, die aber auch mit der Maus ausgelöst werden. Dann ist die grundlegende Art, eine neues Fenster zu öffnen, immer eine Navigation, ähnlich dem Browser. Und letztendlich wird der Name der zu öffnenden Datei als Parameter übergeben, der sich aus der Bindung des Tag-Attributes des Images ergibt.

```
Private Sub Image_Tapped_1(sender As Object, e As TappedRoutedEventArgs)
   Frame.Navigate(GetType(faxDetails1), e.OriginalSource.tag)
End Sub
```

Listing 22 Fax Detailansicht öffnen

8.3.4 Fax Details und Sharing

Der Benutzer navigiert auf die Seite FaxDetail. Dabei wird der Name im ersten Parameter übergeben. Im folgenden Code-Ausschnitt wird die Bilddatei geöffnet, die sich in der PicturesLibrary befinden muss. Um nicht komplette Images im Speicher zu laden, arbeitet man in der Regel mit Streams. Dieser Stream wird einem Bitmap Image zugewiesen, das wiederum als Quelle für das UI-Element vom Typ Image dient.

Wenn der Stream ein weiteres Mal verwendet werden soll, wie hier um die Anzahl der Frames im Bild auszulesen, muss der Stream-Zeiger per Seek auf Anfang gesetzt werden.

Im mittleren Teil des Listing 23 Bild- und Meta-Eigenschaften auslesen Listings, wird ein eigener Lesezugriff ausgeführt, um spezielle Attribute aus dem Exif-Bereich auszulesen. Die Namen werden dabei als String Array übergeben. Eine Liste in der Form Key und

Objekt dient als Datenquelle für das XAML-Formular. Anders als im vorigen Code-Ausschnitt 8.3.1 wird nun das Attribut DataContext verwendet. Dies erlaubt beliebige Objekte als Quellen für die Datenbindung.

```
Protected Overrides Async Sub LoadState(navigationParameter As Object, pageState As
Dictionary(Of String, Object))
  picSF = Await Win-
dows.Storage.KnownFolders.PicturesLibrary.GetFileAsync("VisendoFax\Eingang\" + navi-
gationParameter)
  Dim r As String() = {"System.Photo.DateTaken", "System.Copyright",
       "System.Photo.CameraManufacturer", "System.Photo.CameraModel",
       "System.Author", "System.Subject", "System.Keywords"}
  Dim iProp = Await picSF.Properties.GetImagePropertiesAsync()
  Dim rProp As IDictionary(Of String, Object) = Await
iProp.RetrievePropertiesAsync(r)
  Me.DataContext = rProp
  Dim b As IRandomAccessStream = Await picSF.OpenAsync(FileAccessMode.Read)
  Dim previewBI As BitmapImage = New BitmapImage()
  previewBI.SetSource(b)
  img1.Source = previewBI
  b.Seek(0)
  Dim imgDecode = Await BitmapDecoder.CreateAsync(BitmapDecoder.TiffDecoderId, b)
  pageNumber.Text = "1/" + imgDecode.FrameCount.ToString
End Sub
```

Listing 23 Bild- und Meta-Eigenschaften auslesen

Im XAML-Code erfolgt das Binding an das Attribut des Bildes über den Key des Dictionaries.

```
<TextBox Text="{Binding [System.Photo.DateTaken]}" MinWidth="150" />
…Text="{Binding [System.Autor]}" MinWidth="150" />
```

Figure 1 Binden per Key

Da nun das Dokument in der zentralen Ansicht ist, kann der Benutzer es weitergeben, also sharen. Dazu wird das DataTransfermanager Object verwendet und das Event registriert, welches die Daten letztendlich zur Verfügung stellen wird.

```
Dim dtm As DataTransferManager = DataTransferManager.GetForCurrentView()
Private Sub faxDetails1_Loaded(sender As Object, e As RoutedEventArgs) Handles
Me.Loaded
   AddHandler dtm.DataRequested, AddressOf reqHandler
End Sub
```

Listing 24 Datatransfermanager Event registrieren

Der Austausch findet als Datapackage statt. Dieses wird mit dem Bild gefüllt und kann dann von anderen Anwendungen, dem Share Target, ausgelesen werden.

```
Private Async Sub reqHandler(sender As DataTransferManager, args As DataRe-
questedEventArgs)
  Dim reqDP As DataPackage = args.Request.Data
  reqDP.Properties.Title = "Fax "
  reqDP.Properties.Description = "Fax Details"
  Dim l As List(Of IStorageItem) = New List(Of IStorageItem)
  l.Add(picSF)
```

```
  reqDP.SetStorageItems(1)
End Sub
```

Listing 25 Datapackage füllen

8.3.5 Bilder erstellen, speichern und lesen

Das Erstellen eines Bildes gestaltet sich sehr einfach. Benötigt wird ein MediaCapture-Steuerelement (hier mc) und ein Encoder-Objekt, um im TIFF-Format zu speichern.

```
Private Async Sub Foto_Click(sender As Object, e As RoutedEventArgs) Handles Fo-
to.Click
  Dim img As ImageEncodingProperties = New ImageEncodingProperties()
  Dim sf As StorageFile = Await ApplicationDa-
ta.Current.LocalFolder.CreateFileAsync("job.tiff", CreationCollision-
Option.ReplaceExisting)
  img.Subtype = "TIFF"
  img.Width = 1536
  img.Height = 2048
  Await mc.CapturePhotoToStorageFileAsync(img, sf)
  Await mc.StopPreviewAsync()
  Frame.Navigate(GetType(faxPreview))
End Sub
```

Listing 26 Foto vom Dokument erstellen und speichern

Es hat sich in den Tests herausgestellt, dass es notwendig ist, Bildoptimierung durchzuführen um beispielsweise den Kontrast weiß/schwarz herauszuarbeiten. In WinRT gibt es keine API dafür. Die Evaluierung von Open Source-Komponenten ist nicht Bestandteil dieser Arbeit, aber noch essentiell um eine Marktreife der Lösung zu erlangen.

Das Lesen und Schreiben von Bildern geschieht über Streams. Da WinRT und seine Dateizugriffsmethoden eigene Streamtypen verwenden, müssen diese immer in die .NET-Typen konvertiert werden. Anders als System.IO aus .NET liefert WinRT einen IRandomAccessStream. Der Code-Ausschnitt zeigt, wie ein TIFF geladen wird und per Decoder Frame per Frame gelesen wird.

```
Dim fileStream As IRandomAccessStream = Await
f.OpenAsync(Windows.Storage.FileAccessMode.Read)
Dim imgDecode = Await BitmapDecoder.CreateAsync(BitmapDecoder.TiffDecoderId,
fileStream)
Dim b1 As BitmapFrame = Await imgDecode.GetFrameAsync(pageCount)
Dim pData = Await b1.GetPixelDataAsync
Dim ims = New InMemoryRandomAccessStream()
Dim imgEncode = Await BitmapEncoder.CreateAsync(BitmapEncoder.TiffEncoderId, ims)
   imgEncode.SetPixelData(imgDecode.BitmapPixelFormat,
  imgDecode.BitmapAlphaMode,
  imgDecode.PixelWidth,
  imgDecode.PixelHeight,
  imgDecode.DpiX,
  imgDecode.DpiY,
  pData.DetachPixelData()
)
Await imgEncode.FlushAsync
ims.Seek(0)
```

```
Dim bi As New BitmapImage()
bi.SetSource(ims)
bild1.Source = bi

pageCount += 1
```

Listing 27 Multipage TIFF in seine Frames zerlegen

Das Konvertierungsproblem kann entstehen, wenn das Bild per Upload an den Service übermittelt werden soll. Die dafür notwendige Klasse HttpClient ist in .NET implementiert und erwartet als StreamContent kein WinRT-Objekt. Microsoft liefert seit kurzem Klassen mit, die per Extension Syntax verschiedene Konvertierungsmethoden anbieten, hier As-Stream.

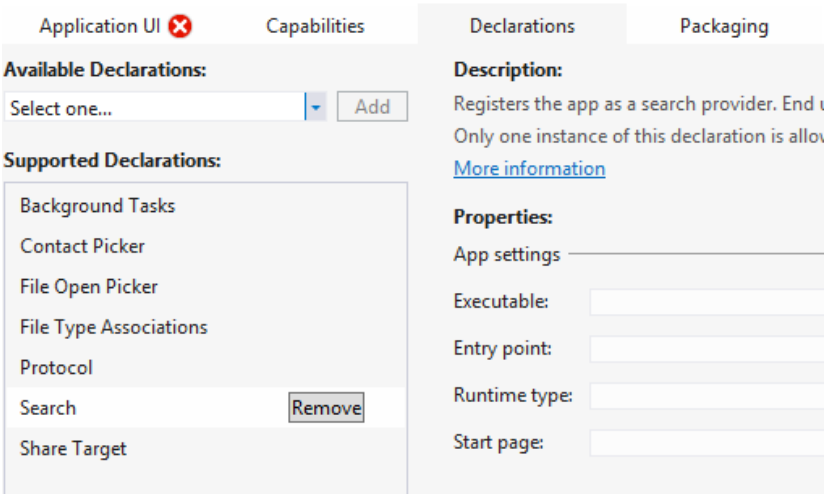
```
Dim url As New Uri("http://../upload1.ashx?datei=" & imgName)
Dim fs = Await Win-
dows.Storage.ApplicationData.Current.TemporaryFolder.GetFileAsync("job.tiff")
Dim b As IBuffer = Await FileIO.ReadBufferAsync(fs)
Dim sc As StreamContent = New StreamContent(b.AsStream)
'asstream extension Methode aus System.Runtime.InteropServices.WindowsRuntime
Dim http As New HttpClient()
Dim req As HttpRequestMessage = New HttpRequestMessage(HttpMethod.Post, url)
req.Content = sc
Dim resp As HttpResponseMessage = Await http.SendAsync(req)
Dim msg As MessageDialog = New MessageDialog(Await resp.Content.ReadAsStringAsync)

Await msg.ShowAsync()
```

Listing 28 Bild-Upload per httpClient

8.3.6 Suche per Charm

In einer ruhenden Anwendung zu arbeiten, ist schon im Ansatz außergewöhnlich. Dabei ist das genau das, was der Benutzer tut, wenn er die Funktion Search im Charm aufruft. Der Anwendung muss im Manifest die Fähigkeit erst vergeben werden.



Listing 29 Projekteinstellungen Application Manifest

Der Aufruf der eigentlichen Suchseite wird in der Application-Klasse und dem zugehörigen Event durchgeführt.

```
Protected Overrides Sub OnShareTargetActivated(ByVal e As Win-
dows.ApplicationModel.Activation.ShareTargetActivatedEventArgs)
   Dim shareTargetPage = New ShareTargetPage1
   shareTargetPage.Activate(e)
End Sub
```

Listing 30 App.xaml.vb SearchTarget Event

Die eigentliche Arbeit beginnt dann in der Suchseite. Wegen der zu erwartenden großen Ergebnismengen ist der erste Ansatz, beim Dateizugriff mit Virtualisierung und dem Index zu arbeiten. Allerdings stimmen die Ergebnisse nicht. Selbiger Code direkt in der Anwendung liefert allerdings die korrekten Werte. Auch bei Microsoft ist keine Antwort zu bekommen, so dass dies der Autor als möglichen Bug einstuft.

Als alternativer Ansatz kann man die Dateiobjekte in eine Liste laden. Parallel dazu wird eine Liste von Tags aufgebaut und die Anzahl des Vorkommens gezählt. Das Search Template aus Visual Studio liefert eine Page mit vorgegebenen Funktionsblöcken und der etwas ungewohnten Handhabung eines Viewmodels in den letzten Zeilen. Um hier das Rad nicht neu zu erfinden, wird der Ansatz übernommen.

```
Dim queryText
Protected Overrides Async Sub LoadState(navigationParameter As Object, pageState As
Dictionary(Of String, Object))
  queryText = DirectCast(navigationParameter, String)
  Dim filterList = New List(Of Filter)()
  filterList.Add(New Filter("All", 0, True))
  Dim pics = Await Win-
dows.Storage.KnownFolders.PicturesLibrary.GetFolderAsync("VisendoFax\Eingang")
  Dim files As IReadOnlyList(Of StorageFile) = Await
pics.GetFilesAsync(CommonFileQuery.OrderByDate, 0, 8)
  For Each f In files
  Dim d As New FaxDocument
  Dim ra = Await f.OpenAsync(FileAccessMode.Read)
  Dim bmp = New BitmapImage()
  bmp.SetSource(ra)
  d.isSent = False
  d.bild = bmp
  d.SenderKennung = f.Name
  d.name = f.Name
  Dim ep As IDictionary(Of String, Object) = Await
f.Properties.RetrievePropertiesAsync({"System.Photo.TagViewAggregate"})
  d.Tags = ep.First.Value(0)
  If d.Tags <> "" Then
     Dim i = 0
     Dim found = False
    For Each fi As Filter In filterList
        If fi.Name = d.Tags Then
           filterList(i).Count = fi.Count + 1
           found = True
           Exit For
        End If
        i += 1
    Next
    If found = False Then filterList.Add(New Filter(d.Tags, 1, False))
  End If
  liste.Add(d)
Next
…selbiges für “Asugang”
  filterList.Item(0).Count = liste.Count
```

```
  If liste.Count = 0 Then
    ' Error1.Text = "Keine übereinstimmung gefunden"
  End If
  Dim bindableProperties As New PropertySet()
  Me.DefaultViewModel("QueryText") = ChrW(&H201C) + queryText + ChrW(&H201D)
  Me.DefaultViewModel("CanGoBack") = Me._previousContent IsNot Nothing
  Me.DefaultViewModel("Filters") = filterList
  Me.DefaultViewModel("ShowFilters") = filterList.Count > 1
End Sub
```

Wenn der Benutzer auf einen der Tags klickt, um die Ergebnismenge zu filtern (Bilder in 8.1.4) kann mit einer LINQ-Abfrage in der vorhandenen Liste gesucht werden. Da die Liste vom Typ ObservableCollection ist, wird das User Interface auch automatisch aktualisiert.

```
Protected Sub Filter_Checked(sender As Object, e As RoutedEventArgs)
  If filtersViewSource.View IsNot Nothing Then
     Dim F = DirectCast(sender, FrameworkElement).DataContext
     filtersViewSource.View.MoveCurrentTo(F)
     Dim selectedFilter = CType(F, Filter)
     Dim q
     If selectedFilter.Name = "All" Then
        q = From x In (From p In liste
        Where p.name.Contains(queryText)
        Select p)
        Group x By key = x.isSent Into Group
        Select New With {.key = key, .Items = Group}
     Else
        selectedFilter.Active = True
        q = From x In (From p In liste
        Where p.name.Contains(queryText) And p.Tags.Contains(selectedFilter.Name)
        Select p)
        Group x By key = x.isSent Into Group
      Select New With {.key = key, .Items = Group}
    End If
    resultsViewSource.Source = q
  End If
End Sub
```

Listing 31 Filtern der Ergebnismenge per LINQ

Um in der Darstellung (Abbildung 22 Filtern der Ergebnismenge) die Bereichsüberschriften erstellen zu können, werden die eigentlichen Listendaten mit Group By gruppiert. Dies ist nötig, um die Gruppierung in XAML binden zu können. Auch das SemanticZoom-Steuerlement benötigt die gebundenen Daten, aufbereitet in dieser Form.

```
<GridView.GroupStyle>
  <GroupStyle>
    <GroupStyle.HeaderTemplate>
       <DataTemplate>
          <Grid Margin="7,7,0,0">
            <TextBlock  Text="{Binding key,Converter={StaticResource booleanToXCon-
verter}}" Margin="0" FontSize="21"/>
          </Grid>
       </DataTemplate>
    </GroupStyle.HeaderTemplate>
    <GroupStyle.Panel>
      <ItemsPanelTemplate>
         <VariableSizedWrapGrid Orientation="Vertical" Margin="0,0,80,0"/>
```

```
      </ItemsPanelTemplate>
    </GroupStyle.Panel>
  </GroupStyle>
</GridView.GroupStyle>
```

Listing 32 XAML GridView-Gruppierungsbereich

9 Ergebnisanalyse und Auswirkungen

Zusammenfassend muss der Autor einräumen, dass das Ziel, einen lauffähiges Software-Prototyp abzuliefern, gescheitert ist. Nachfolgend werden die Ursachen kurz zusammengefasst und mögliche Maßnahmen vorgeschlagen.

9.1 Planung und Know-how

Ein häufiger Fehler in Software-Projekten ist eine verfehlte Aufwandsschätzung. Das begründet sich meist in der fehlenden Expertise. Habe ich so etwas ähnliches schon einmal gemacht? Der Autor muss diese Frage zu häufig nachträglich mit Nein beantworten.

Um überhaupt einen effizienten Einstieg in die Materie zu schaffen, werden in folgender Tabelle die nötigen Skills definiert.

Tabelle 6 Skill-Anforderungen für schnellen Einstieg

XAML Skills; WPF oder besser Silverlight	300
Design-Kenntnisse	200
.NET-Programmiersprache C# oder VB.NET	400
HTTP und Services	200
Windows 8 Metro-Benutzererfahrung	500

Der Level der Expertise reicht von 100 für Anfänger bis 500 für Experten.

Ausgehend von identer Funktion ist der Aufwand für die Implementierung eines Projektes auf Windows 8 im Vergleich zu Windows 7 deutlich höher. Noch aufwändiger wird es, Windows 8 richtig und komplett zu nutzen. Eine tatsächliche Aufwandsschätzung im eigentlichen Sinne ist zum jetzigen Zeitpunkt nicht seriös möglich.

Offensichtlich Unterschiede bezüglich des Entwicklungszweiges XAML vs. HTML kommen erschwerend hinzu. Dies war nicht Gegenstand der Untersuchung, ist jedoch bei der Recherche aufgefallen. Beispielhaft seien das Flyout und das TimePicker-Steuerelement aus HTML erwähnt, zu denen es kein XAML-Pendant gibt. Daraus entwickelt wurde die Checkliste Anlage 1 UI Designplanung.

In Anlage 2 werden bekannte Steuerelemente den WinRT-Steuerelementen funktionell gegenübergestellt.

Als Vorgehensweise empfiehlt der Autor, ein Ziel und ein Ende zu definieren. Dazwischen sollte man Meilensteine in kurzen Zyklen von wenigen Tagen oder maximal Wochen planen. An diesen Terminen muss es möglich sein, wesentliche Teile des Projektes neu zu planen, basierend auf dem Gelernten des vorangegangenen Entwicklungszyklus.

9.2 Benutzererfahrung

Der Konzeptentwurf erfolgt am einfachstem auf Papier mit Storyboards, mangels anderen Werkzeugen. Dabei geht es natürlich um die Oberflächen, aber auch um die Führung des Benutzers.

Im ersten Schritt wird die Aufgabe analysiert, anhand von Benutzeraussagen, Beobachtungen des üblichen Benutzerverhaltens oder Expertenmeinungen. Darüber hinaus geht es um weiche Attribute, wie Effizienz, Effektivität oder Zufriedenheit. Als problematisch stellt sich dabei die Messbarkeit heraus. Unter der Oberfläche lässt sich dies am besten mit automatischem Usertracking realisieren, um beispielsweise die Fehlerquote zu ermitteln. In der Praxis existieren allerdings datenschutzrechliche Beschränkungen.

Der Benutzer wird nicht in den Entwurfs- und Designprozess eingebunden und ist damit kein Projektbeteiligter.

Bisher arbeitet ein Anwender typischerweise mit wenigen Programmen regelmäßig. Diese werden relativ lange, meist über Jahre in identen Versionen eingesetzt. Zur Unterstützung der Benutzer gibt es Hilfe per F1, ein Handbuch, Schulungen und das Helpdesk.

Apps hingegen werden sehr kurz verwendet. In Kombination mit kurzen Updatezyklen muss eine sehr kurze Lernkurve realisierbar sein. Die Anwendung muss sich deshalb zu jeder Zeit selbst erklären. Das ist nur möglich, mit reduzierter Funktionalität.

Folglich ist die Oberfläche das strategisch wichtigste Produktfeature. Das Interface muss genau die Zielgruppe ansprechen. Dabei geht der Benutzer aufgrund seiner persönlichen Erfahrungen mit einer gewissen Erwartungshaltung an Windows 8 heran. Da das Metro-Design[70] mit der Vergangenheit von Windows bricht, ist eine Migration von Oberflächen zum Scheitern verurteilt. Es fehlen Steuerelemente, wie Treeview, Datagrid oder Menu, die heute in nahezu jeder Desktop-Anwendung verwendet werden. Dafür gibt es in Windows 8 Steuerelement, wie Semanticzoom, Appbar oder das Charm- Konzept, die ungewohnte touch-orientierte Bedienung erlauben. Auf Drag& Drop mit Daten und Listen sollte man weitestgehend verzichten, da bei deren Verwendung mit Touch die Fehlerquote höher ist. Selbst wenn der Benutzer eine Fehleingabe gemacht hat, wird durch die Style Guides gefordert, dass Aktivitäten jederzeit vom Benutzer rückgängig gemacht werden können.

[70] Der Name Metro könnte von Microsoft kurzfristig geändert werden, wegen Marken-Kollision mit dem deutschen Handelskonzern Metro.

Interface Design muss auf Geräte und Einsatzumgebung Rücksicht nehmen. Einfachste aber notwendige Darstellungen sind Portrait, Full oder Flipped. Wenn Entwickler sich an die Style Guides halten, reicht es für eine solide App. Als empfehlenswerte Option kann ein Usability Engineer das Design verbessern. Um optisch eine hochwertige Anwendung zu erhalten, kann letztlich noch ein Designer zum Einsatz kommen. Dank des Template-Konzeptes von XAML kann jedes Control in seiner äußeren Erscheinung komplett verändert werden, so dass aus einem eckigen Button ein runder Zurück-Pfeil wird. Standard Styles finden sich der StandardStyles.xml.

Design meint vor allem Benutzerorientierung im Sinne von Industriedesign und weniger die schicke Oberfläche. Jede Nutzersituation ist einzeln zu bewerten und zu berücksichtigen. Dieselbe Aufgabenstellung für die Arbeit auf der Couch oder im Büro wird ein unterschiedliches Interface erfordern. Dabei sind die letztlichen Lösungsansätze meist sehr einfach und wenig variantenreich. Die Software Industrie lässt sich die Konzepte allerdings[71] schützen. Dies begrenzt den Einsatz und erzwingt eventuell eine Alternative, die auch schlechter sein kann.

Ganz generell finden sich in den Style Guides noch weitere Regeln, die erheblichen Einfluss auf das Design haben. So sollen vom Benutzer erfasste Informationen automatisch und auch dauerhaft wieder aufrufbar sein. Einen Speichern-Buttons oder ein Message-Dialog zu bestätigen, ist dabei nur selten sinnvoll. Die Schwierigkeit besteht darin, herauszufinden, wann man auf bekannte Konzepte zurückgreifen kann. Für den Autor hat es sich als zielführend erweisen, beim Erfassen eines Datensatzes explizit ein Symbol Speichern anzubieten. Viele Metro-Anwendungen tun das nicht.

9.3 Windows 8

Die spezifischen Eigenschaften von Windows, wie Energieverwaltung, Netzwerkstack, Sensorik oder Appbars werden in den aktuellen Metro-Anwendungen oft nicht genutzt. Da der Benutzer sie nicht kennt, wird er sie auch nicht vermissen. Folglich für sind dies Funktionen, die oftmals einfach zu streichen sind um z.B. einen Termin zu halten.

Um die Möglichkeiten von Windows 8 speziell auf verschiedenen Geräten zu erforschen, müssen über einen längeren Zeitraum der Microsoft Store und seine darin befindlichen Metro Apps verwendet werden. Wichtige Quelle zur Recherche über das Benutzerverhalten sind die User Ratings und Kommentare im Microsoft Store, auch wenn diese andere Anwendungen behandeln.

[71] http://www.heise.de/newsticker/meldung/Apple-laesst-sich-ein-und-ausblendbaren-Scrollbalken-schuetzen-1644543.html

9.4 WinRT API

Obwohl an die Entwicklung mit .NET erinnernd, sind die Unterschiede erheblich. Nur ganz wenige Anwendungen werden sich in wenigen Stunden von WPF oder Silverlight nach WinRT portieren lassen. Das sind Anwendungen, die mit einem WCF Service Daten anzeigen und speichern.

Es ist insofern illusorisch von einer gemeinsamen Codebasis auszugehen. Trotzdem betrachtet man eine Momentaufnahme. Technisch denkbar und möglich ist, das Microsoft WinRT in kürzeren Zyklen als Windows weiterentwickelt wird.

Als problematisch hat sich die Fehlersuche und das daraus resultierende Debuggen herausgestellt. Manche Fehler passieren im COM-Teil der WinRT und werden als altbekannte kryptische HResult-Fehlercodes angezeigt. Hier sind dann oft deutlich tiefere .NET Kenntnisse nötig, um die Fehler zu finden. Durch den asynchronen Ablauf kommen Fehlermeldungen oft nicht zum Vorschein oder treten während des Debugging gar nicht auf. Wie aus XAML grundsätzlich bekannt, werden bei der Datenbindung keine Exceptions geworfen. Leider fehlt das XAML Debugging, das man aus Silverlight und WPF kennt.

9.5 Architektur

Viele gewohnte Funktionen sind in WinRT schlicht nicht vorhanden, wie beispielsweise ADO.NET. Folglich gibt es keine SQL Server-Unterstützung und damit auch keine Client Server-Architektur. Wobei ein Datenbankservice auf einem mobilen Gerät ein Widerspruch in sich ist.

Der Einstieg fällt dem Entwickler leichter, wenn er ohnehin schon mit asynchronen Web-Szenarien gearbeitet hat, wie bei Silverlight. Pferdefuß dabei ist, dass die dort vorhandene und beliebte Service-Schicht RIA Services unter WinRT nicht unterstützt wird.

Am einfachsten und damit am wenigsten aufwändig ist, die Realisierung der Service-Schicht mit WCF. Man nimmt sich dabei aber, wie diese Arbeit zeigt, die modernen Möglichkeiten, die REST und HTTP bieten.

Ergänzend muss noch erwähnt werden, dass SQLLite[72] als alternative Datenbank als Open Source für WinRT zur Verfügung steht.

Letztendlich wird eine mehrschichtige Architektur zum Einsatz kommen. Datenbank und Service werden extra gehostet. Auf dem Device wird per http REST Call oder WCF Proxy der Service aufgerufen, die Daten deserialisiert und ggf lokal am Device im Datei System gespeichert um auch offline Szenarien bewältigen zu können.

[72] http://sqlWinRT.codeplex.com/

9.6 Offene Punkte

Der Testprozess von Anwendungen wird heute mit Unit Tests gerne automatisiert. In der Zwischenzeit lassen sich dank MVC oder MVVM auch User Interfaces ganz leidlich mit Unit Test und Mocking Frameworks testen. Das gilt grundsätzlich auch für WinRT Metro Apps. Allerdings machen zahlreiche Szenarien, die an den verwendeten Geräten und spezifischen Nutzersituationen hängen, das Testen zukünftig komplexer. Weitere Betrachtungen wurden in dieser Arbeit nicht getroffen.

Der going-to-market-Prozess über den Microsoft Store wurde ebenfalls nicht betrachtet, weil die nötige Infrastruktur schlicht noch nicht zur Verfügung steht. Aktuell verfügbar ist das App Certification Kit [73], mit dem sich Apps selbst formal testen lassen auf Konformität mit den Windows Store-Richtlinien.

9.7 Problempunkte

Auch mit dem Markteinstieg von Microsoft mit dem Surface Tablet ist aktuell die Palette an verfügbarer passender Hardware gering. Die Recherche ergab, dass es keinen Monitor gibt, der Touch-Funktion ohne erhobenen Rahmen realisiert. Eine Grundvoraussetzung für die seitlichen Wischgesten. Von den angekündigten Windows Phone 8-Geräten ist aktuell nichts bekannt. Der Erfolg von Windows 8 hängt aber an der Hardware. Software Projekte werden, speziell im Geschäftsbereich, nur gestartet, wenn eine ausreichende User Basis vorhanden ist. Auch die strikten Bedingungen des Windows Stores können den Erfolg von Windows 8 aus IT-Entscheidersicht bremsen, aber letztendlich nicht verhindern.

Für den Software-Entwickler ist die Perspektive verlockend, einfachen Zugang zu einem potentiellen Milliardenmarkt zu erhalten. Dazu sind im Windows 8-Konzept Anti Piracy-Maßnahmen enthalten.

Einzig der nötige Aufwand, um eine Lösung zu erstellen, ist definitiv höher als mit anderen Microsoft-Plattformen. Das gleiche gilt aber auch für Android oder iOS

[73] ACK http://msdn.microsoft.com/en-us/library/windows/apps/hh694081.aspx

Literatur

Dorau 211	Dorau, Rainer: Emotionales Interaktionsdesign, Heidelberg Dordrecht London New York, 2011.
Grässle, Baumann, Baumann 2000	Grässle Patrik, Baumann Henriette und Philip, UML projektorientiert, Bonn, Galileo 2000.
Lidwell, Holden, Butler 2003	Lidwell William, Holden, Kristina, Buttler Jill, Universal Principles of Design, Massachusetts, Rockpot Publishers 2003.
Microsoft UX Design Guidelines	http://www.microsoft.com/en-us/download/details.aspx?id=2695, am 22.10.2010.
UsabilityProfessionals2011	http://de.scribd.com/doc/64407104/Usability-Professionals-2011-Tagungsband , 201.
MSDN WinRT Dokumentation	http://msdn.microsoft.com/library/windows/apps/ Juni 2012

Anlagen

Anlage I, Checkliste UI Design Planung

Einsatzzweck	nötig	Maßnahme
Mobiler Einsatz		Virtuelle Tastatur, 90° Ansicht definieren
Single Page		Dynamisches Laden von Usercontrols
Hierarchische Struktur		Navigation Frame
Datengetrieben		GridView, SemanticZoom
Mehr als 50 Datensätze		Gruppierung der Daten, Hierarchische Führung
Dauerhafte Einstellungen		SettingsPane
Bilder, Videos, Medien		Share Source
Offline fähig		Lokale Daten
Fehlermeldungen		Kopierbar, Klartext, Logging
Eingabefehler		Im Kontext direkt anzeigen
Benachrichtigung des Benutzers		Kachel wenn dauerhaft, Toast wenn umgehend (am besten mit Sound) Bei ausgeschaltetem Gerät Lockscreen
Suche in Anwendung		Kontextbezogen auf akuelle Liste Mit Detailoption
Mehrfache Optionen zb in Appbar		Popup mit isLightdismiss
Mehrsprachig		
Aktivierung durch fremde Anwendungen		Protokollaktivierung File Typ Aktivierung
Interaktion mit Desktop		Zwischenablage

Wandernder Benutzer		Roaming Settings (Skydrive)

Anlage II, Steuerelement in WinRT Vergleich

Desktop	Windows 8	Hinweis
Treeview	SemanticZoom	Nur sinnvoll 2-3 Ebenen
Menü	Appbar	Eine Hierarchie (Submenü) per Popup
Label	Textblock	
Checkbox	Checkbox	besser ToggleSwitch
Trackbar	Slider	
TextBox	TextBox	Inhaltsspezifische OnScreen Tastatur
PictureBox	Image	
Datetimepicker		Keine Entsprechung (auch Kalender) in XAML
Progressbar	Progressbar	Auch möglich Progressring
Tooltip	TooltipService	
Webbrowser	WebView	
GroupBox	Canvas, Grid, Stack-Panel	Gruppierungen werden durch Design vorge-nommen
Timer	DispatcherTimer	DispatcherTimer ist eine Klasse
OpenFileDialog	OpenFileDialog	
NotfiyIcon	Toast, Tile	Tray Icons wurden entfernt. Status wird auf der Tile abgebildet.
DataGridView	Gridview, Listview, FlipView	Die Funktion und Aussehen unterscheidet sich erheblich
Dialog	MessageDialog	Verwendung sehr unterschiedlich. Beides kei-ne Steuerelemente

Es werden nur stark unterschiedliche Elemente aufgeführt und Empfehlungen ausgesprochen mit welchem Windows 8 Controls man die Problemstellung lösen kann.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Burghausen, den 06.08.2012

Johannes Preishuber-Pflügl