Hochschule für Technik und Wirtschaft Dresden (FH)
Fachbereich Informatik/Mathematik

Diplomarbeit

im Studiengang Medieninformatik

Thema:

Prototypischer Technologievergleich von Microsoft Silverlight und Adobe Flash mit Fokus Richmedia-Integration – Qualität, Codecs, Streaming, Steuerungsmöglichkeiten

eingereicht von: Peter Rauschenbach
eingereicht am: 30.09.2008
Betreuerin: Prof. Dr. Teresa Merino

Inhaltsverzeichnis

Einleitung

Das Internet ist mittlerweile aufgrund internetfähiger Handys, tragbarer PCs und WLAN an jedem Ort zu jeder Zeit zugänglich. Das Internet hat sich von einem für Militär- und Forschungszwecke geschaffenen Netzwerk zum Unterhaltungsmedium und Informationspool entwickelt. Doch mit der zunehmenden Nutzerzahl und Verbreitung sind auch die Anforderungen an das World-Wide-Web gestiegen, statische Seiten aus reinem HTML-Code mit Texten sowie Bildern sind längst überholt, dynamische Inhalte mit Videoclips, eigenen Nutzerkonten oder personalisierten Einstellungen, welche aus Datenbanken abgerufen werden, sind heutzutage längst Standard. Das neue Stichwort lautet Rich-Internet-Applications, Anwendungen mit reichhaltigem Inhalt, welche die Techniken des Internets nutzen und eine intuitive Benutzeroberfläche bieten. Sei es Video, Musik, Spiele oder selbst Office-Programme, so gut wie nichts ist unmöglich. Die Entwicklung schreitet rapide voran, neue Techniken werden jedes Jahr in unüberschaubarer Zahl entwickelt und verschwinden teilweise ebenso schnell wieder aus der Fachpresse sowie den Köpfen der Entwickler und Nutzer. Jedoch gibt es Techniken, welche sich seit einigen Jahren etabliert haben, welche mit der Zeit gehen und in immer neueren und umfangreicheren Versionen versuchen den stetig wachsenden Anforderungen gerecht zu werden. Eines der Entwicklungswerkzeuge, welches als sozusagen als Pionier schon in seinen ersten Versionen Interaktion zwischen Benutzer und Inhalt der Webseite angeboten hat, war das Programm Flash von Adobe. Damit ließen sich nicht nur optisch ansprechende Webseiten gestalten, sondern auch erste Spiele, die ohne Download, ohne Installation für jeden, der den entsprechenden Flash-Player installiert hatte, überall abruf- und spielbar waren. Und kommen neue Techniken auf den Markt, welche das Ziel haben, bewährte Technologien ablösen wollen, indem sie neue Funktionen bieten, welche nach Aussagen der jeweiligen Hersteller das Internet revolutionieren sollen.

Die aktuelle Software für grafische Webinhalte von Microsoft nennt sich Silverlight, liegt zur Zeit in der Version 1.0 vor, als Beta bereits in der Version 2.0, und verspricht die größte Konkurrenz zu Adobe Flash zu werden. Gerade im Bereich des Streaming, der Videosteuerung sowie der Qualität sind die Anforderungen nicht zuletzt durch den Datenübertragungsstandard DSL enorm hoch und können somit durchaus über Erfolg oder Scheitern einer Technologie entscheiden.

Im Rahmen des eCampus Projektes der HTW Dresden, welches sich zum Ziel gesetzt hat, vorhandene Strukturen, Erfahrungen und Potentiale im Bereich eLearning zu bündeln um innovative Lehr- und Lernformen nachhaltig und als integrativen Bestandteil in der akademischen Aus- und Weiterbildung zu etablieren, bot sich für mich die Möglichkeit, eine virtuelle Besichtigungsmöglichkeit der Bibliothek der HTW Dresden unter Verwendung dieser neuen Technologien zu programmieren. Die Projektleitung des eCampus-Projektes teilen sich Frau Prof. Dr. Teresa Merino, Professorin für Multimediale Werkzeuge an der HTW Dresden, Frau Renate Rudat, Leiterin des Sprachzentrums und Lehrkraft für Englisch und Herr Prof. Dr.-Ing. Ivan Panajotov, an der HTW verantwortlich für die Bereiche Reproduktionstechnik, Kartografische Originalherstellung sowie EDV.

Das Ziel der Diplomarbeit ist es, durch die prototypische Implementierung einer Anwendung zur Realisierung eines virtuellen Rundgangs unter Verwendung von Flash bzw. Silverlight die Fähigkeiten der jeweiligen Technologie im Hinblick auf Merkmale wie unterstützte Codecs, Streaming, Videoqualität sowie -steuerung zu untersuchen und vergleichend darzustellen um die geeignete Technik zur finalen Umsetzung dieser Aufgabenstellung auszuwählen.

Diese Aspekte sind von besonderer Bedeutung, da diese Besichtigungsmöglichkeit basierend auf über Streaming bereitgestelltem Videomaterial aufgebaut werden soll.

Ferner wird dabei auf die grundlegende Eignung der Software für ähnliche Projekte sowie die Vor- und Nachteile bei der Anwendung der jeweiligen Technik eingegangen.

Im ersten Kapitel wird eine Überblick über die unterschiedlichen Typen von Webinhalten gegeben.

Das zweite Kapitel zeigt die Neuerungen in Actionscript 3.0, dabei wird auf die Streaming- und Steuerungsmöglichkeiten sowie vorhandene Codecs eingegangen und Vor- und Nachteile der Technik aufgezeigt, welche sich aus der theoretischen Analyse ergeben.

Anschließend wird Silverlight im dritten Kapitel vorgestellt und ebenfalls auf oben genannte Aspekte hin untersucht.

Kapitel vier beschäftigt sich mit der Konzeption der Anwendung, dabei wird die zu erreichende Zielgruppe, den Funktionsumfang sowie die technische Spezifikation untersucht.

Des Weiteren befasst sich die Arbeit in Kapitel Nummer fünf mit der prototypischen Implementierung der Anwendung mit Flash CS3 und Actionscript 3.0, zeigt den Aufbau der XML-Beschreibung sowie der Klassen und Besonderheiten, welche bei der Umsetzung aufgetreten sind.

Die Erklärung zur Implementierung mit Silverlight findet dann im sechsten Kapitel statt, dabei wird ebenfalls auf den Aufbau der Anwendung sowie besondere Aspekte beim praktischen Einsatz eingegangen.

Zum Schluss folgt im siebten Kapitel das Fazit, in welchem die Untersuchungsergebnisse zusammengefasst werden.

1. Webinhalte

1.1 statische Inhalte

Statische Inhalte bilden historisch gesehen die Grundlage des WWW, war es doch in seiner ursprünglichen Form aufgebaut worden, um Forschungsergebnisse über dieses Hypertext-System mit anderen Forschungseinrichtungen, Mitarbeitern, Kollegen und Bekannten zu teilen. Diese Forschungsergebnisse lagen in statischer Form vor, als Übersichten mit Daten und Erklärungen, welche über Links, den so genannten Hyperlinks, miteinander verbunden waren, um eine einfache Navigation zu ermöglichen. Der statische Inhalt setzt sich zum Einen aus den aus ASCII-Zeichen bestehenden HTML-Dokumenten, welche basierend auf der universellen Beschreibungssprache SGML eine auf Tags aufbauende Struktur besitzen, wodurch einfache Textformatierungen und -positionierungen sowie Tabellen etc. möglich sind, zum Anderen aus Bildinhalten, welche über die HTML-Sprache in die Webseite eingebunden werden können, zusammen. Diese Inhaltstypen waren für den ursprünglichen Zweck ausreichend, und trotz fehlender Interaktionsmöglichkeit eine Revolution, man konnte nun auf einen riesigen Wissensschatz zurückgreifen, und selbst Inhalte veröffentlichen, die von jedem Zugangspunkt des WWW erreichbar waren. Dabei funktioniert die Kommunikation zwischen Client und Server nach dem Client-Server-Prinzip, bei dem der Client eine Anfrage, den so genannten Request, an den Server sendet, dieser verarbeitet die Nachricht, lädt die entsprechende Webseite und schickt diese dann als Antwort, auch Response genannt, an den Client zurück, der die Antwort dann im Webbrowser darstellt. Die Aufgabe des Servers besteht damit aus Nachrichtenverarbeitung, Zugriffskontrolle und Auslieferung des statischen Inhaltes, es findet dabei keine Verarbeitung der Inhalte durch den Server statt.

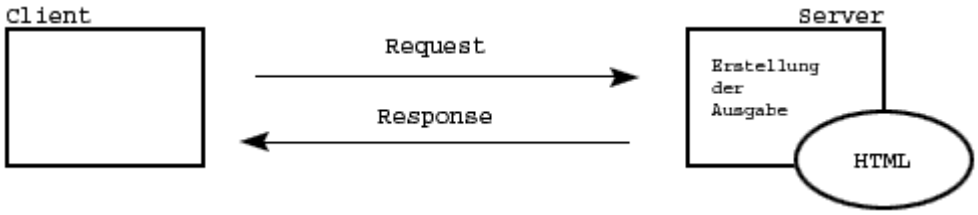

Abbildung 1: Kommunikationsprinzip bei statischen Inhalten [www02]

1.2 Dynamische Inhalte

Mit der zunehmenden Akzeptanz und Verbreitung des World Wide Web stiegen auch die Anforderungen an die Inhalte, die der Nutzer geboten bekommt. Das Bedürfnis nach einfacher Interaktion wuchs, die jeweilige Webseite sollte mit dem Nutzer kommunizieren und sich dementsprechend anpassen lassen. Erste Ansätze wie CGI mit der Skriptsprache PERL bzw. mit vorkompilierten Anwendungen in C setzten dem unflexiblen Inhalten ein Ende, unter anderem zusammen mit den Programmiersprachen JavaScript, JSP, Servlets, ASP.NET, PHP, Flash, etc. Somit konnten einfache Anwendungen geschaffen werden, welche nicht allein auf dem Rechner des Betrachters ausgeführt werden, wie z.B. Webmail-Programme, eine Zusammenarbeit mit serverseitiger Logik erweiterte den Funktionsumfang beträchtlich. Später entwickelten sich gemischte Formen wie bspw. AJAX, bei dem JavaScript Requests versendet, vom Server bearbeitet und anschließend Teile des Seiteninhalts dynamisch geändert werden, damit sich Webseiten zunehmend dem Verhalten von Desktop-Anwendungen annähern. Datenbankinhalte konnten ausgelesen, angezeigt und bearbeitet werden, und selbst Anwendungen wie Online-Versionen einer Textverarbeitung konnten problemlos implementiert werden.

Dabei war und ist für bestimmte Inhaltstypen ein Browser-Plugin notwendig, denn bspw. Flash-Applikationen laufen nur bei entsprechend installiertem Flash-Player-Plugin, Java-Anwendungen funktionieren nur, wenn die Java-Software auch auf dem Endgerät vorhanden ist, wobei JavaScript eine Ausnahme bildet, da es bereits in alle gängigen Browser integriert wurde, allerdings muss es dort aktiviert sein. Unterschieden wird dabei zwischen client- und serverseitiger Dynamik, wobei sich die Erstere mit der Problemlösung von Aufgaben beschäftigt, welche auf dem Rechner des Clients zu erledigen sind und keine Kommunikation mit dem Server voraussetzen, wie z.B. die Überprüfung von Formulardaten. Dies geschieht bei AJAX z.B. durch JavaScript. Die serverseitige Dynamik umfasst die Generierung des eigentlichen Webseiteninhaltes auf Seiten des Servers, dieser setzt nach dem Eintreffen eines Requests Programme oder Skripte ein, um vorhandene HTML-Seiten zu verändern bzw. neue Seiten u.a. durch Verarbeitung von Nutzereingaben und externen Quellen wie Datenbanken etc. zu erstellen. Diese veränderten oder neu erstellten Inhalte werden dann an den Client zurückgeliefert und dort dargestellt [www02].

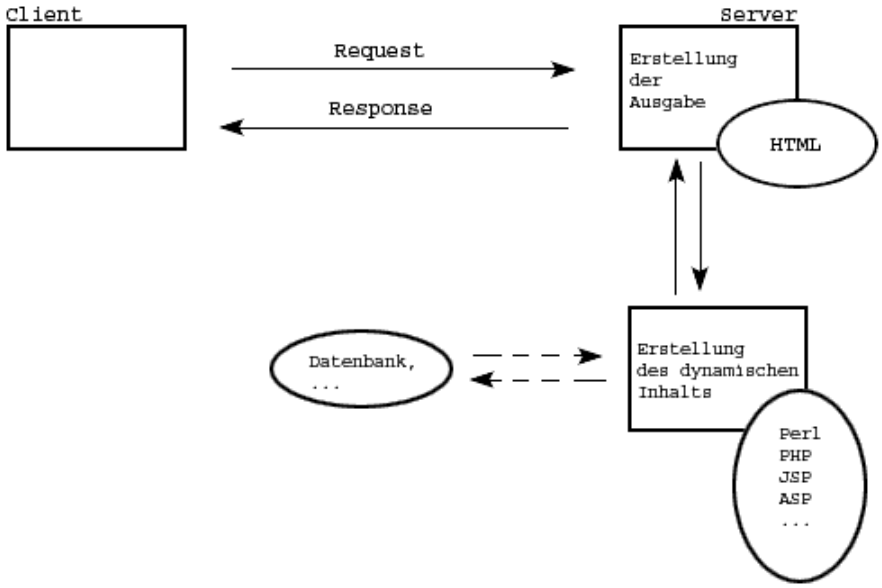

Abbildung 2: Kommunikationsprinzip dynamischer Webseiten [www02]

1.3 Richmedia und Rich-Internet-Applications

Der Begriff Richmedia umfasst alle Medien, die dem Anspruch gerecht werden, durch ihren Einsatz Anwendungen zu bereichern. Richmedia steht für Reichhaltigkeit, nicht unbedingt bezogen auf den Inhalt, sondern auf die dargebotenen Möglichkeiten, so ist dabei der Einsatz von optischen und akustischen Inhalten wie Video, Ton und Animation gemeint. Richmedia steht dabei eher für eine Masse an Inhalten und ein breit gefächertes Angebot an Darstellungsformen, welche dem Nutzer präsentiert werden können und Inhalte, z.B. durch den Einsatz eines Videos, ergänzen. Ein Vorreiter auf dem Gebiet der Richmedia-Integration ist Flash, welches dem Benutzer schon in ersten Versionen Animation und Sound, ab Version 6 auch Video geboten hat, vorausgesetzt, das entsprechende Plugin wurde vorher auf dem Rechner installiert. Nicht nur animierte Webseiten, auch kleinere Spiele mit durchaus ansprechender grafischer Gestaltung forcierten die Verbreitung von Flash und dessen Plugin und zeigten, welche Möglichkeiten im World Wide Web vorhanden waren. Durch die Interoperabilität zwischen den Sprachen, wie z.B. den Einsatz von Flash kombiniert mit PHP-Funktionalität, bieten sich Möglichkeiten, die dynamische Inhalte einer einzelnen Sprache nicht bieten könnten. Dabei ist Richmedia nicht als neue Technologie zu sehen, wie beim Sprung von statischen zu dynamischen Inhalten, vielmehr als Ergänzung der Inhalte um Funktionalität und optische Qualität, an die der Nutzer aus dem täglichen Leben bspw. durch TV und Radio gewohnt ist und deren Einsatz die Grenzen zwischen den genannten Medien und Internet langsam verschwinden lässt. Der Begriff Richmedia ist streng mit den RIAs, den Rich-Internet-Applications, verknüpft. Diese Anwendungen verwenden Techniken des Internets kombiniert mit aus Desktop-Applikationen bekannter Funktionalität, wie Drag-and-Drop und Verwendung von Tastaturkürzeln, und bieten damit einen Mehrwert gegenüber herkömmlichen Webanwendungen. Die Leistungsfähigkeit kann enorm gesteigert werden, indem Berechnungen clientseitig ohne eine Anfrage an einen Server senden zu müssen ausgeführt werden und damit flüssiger bedient werden können.

Erkennbar sind RIAs daran, dass sie ohne Installation unter Verwendung von Internettechnologien ausgeführt werden können und Interaktionsmöglichkeiten mit dem Benutzer bieten. Die Anforderungen an die clientseitige Leistungsfähigkeit sowie die Größe der herunterzuladenden Anwendung steigen beim Einsatz von RIAs zwar an, durch den Einsatz von Streaming und Komprimierungstechniken wird dieser Nachteil jedoch ausgeglichen, zusätzlich wird die Netzwerkbelastung herabgesetzt und der Zugriff auf lokale Daten ist möglich. Auch im Bereich der Mobile Devices, vor allem der Mobiltelefone, und Embedded Devices wie Navigationssysteme besteht Bedarf an einfachen sowie intuitiven Bedienoberflächen, welche zudem nicht installiert werden müssen und deren Größe trotzdem durch das Nachladen von Daten bei Bedarf gering gehalten werden kann [www01].

2. Actionscript 3.0

Actionscript als Programmiersprache ist eng Verknüpft mit dem Produkt Flash von Adobe, dem zweitgrößten Softwarehersteller der Welt, welcher die Firma Macromedia, den bisherigen Entwickler von Flash und somit auch Actionscript, im Jahr 2005 aufgekauft hat. Gedacht war Flash in seiner ursprünglichen Version als Grafikprogramm für einen Tablet PC, durch das Scheitern des Projektes entstand die Idee für das Vektoranimationsprogramm FutureSplash der Firma FutureWave Software, welches durch Übernahme von Macromedia im Jahr 1995 und ein daraufhin kreiertes Webbrowser-Plugin für Flash 1.0, welches im November 1996 veröffentlicht wurde, große Beachtung und Verbreitung in aller Welt erlangte.
Im August 2000 erschien dann Actionscript 1.0, welches auf ECMAScript basierte und somit ähnlich JavaScript aufgebaut war. Flash wurde dadurch zu einem ernst zu nehmenden Entwicklungswerkzeug. Schon 2003 erfolgte die nächste Evolutionsstufe der Software, als im September Flash MX 2004 vorgestellt wurde, welches durch Actionscript 2.0 erstmals eine objektorientierte Programmierung ermöglichte. Zwischendurch erschienen mehrere Versionen des Produktes, welche den Umfang und die Einsatzmöglichkeiten ständig erweiterten, wobei diese jeweils die vorhergehende Actionscript-Version nutzten und somit von programmiertechnischer Sicht nicht ins Gewicht fallen.

Mit der Version CS3 liegt Flash in der neunten Generation vor und bietet durch die Einbettung von Actionscript 3.0 neue Funktionen und Ansätze, welche in diesem Kapitel behandelt werden. Dabei wird vor allem neben den allgemeinen Neuerungen AS3 hinsichtlich Streaming, Steuerungsmöglichkeiten und Codecs betrachtet, da diese Faktoren ausschlaggebend für die Eignung zur Implementierung des virtuellen Bibliotheksrundgang im Rahmen des eCampus-Projektes sind.

2.1 Neuerungen in Actionscript 3.0

Einen so großen Schritt wie von Actionscript 2.0 zur Version 3.0 gab es bei der Entwicklung von AS1 zu AS2 nicht, es wurde damals zwar die sehr wichtige Objektorientierung implementiert, jedoch war der Actionscript-Interpreter in seiner Form lediglich weiterentwickelt und somit, in gewissem Maß, abwärtskompatibel zur Vorgängerversion belassen worden. Der Flashplayer 9 besitzt aus technischer Sicht einen völlig neuen Interpreter, welcher lediglich AS3 unterstützt. Zwar können mit dem Player auch Projekte basierend auf Actionscript 1 und 2 abgespielt werden, jedoch wird dafür ein anderer Interpreter genutzt. AS3 ist in gewisser Hinsicht eine vollkommen neue Programmiersprache, wodurch eine Umwandlung einer AS2-Anwendung in eine AS3-Applikation nicht ohne größere Mühen vorgenommen werden kann, hat sich doch nicht allein der Elementzugriff und das Ereignismodell enorm verändert.

Variablen und Typisierung

Die strikte Typisierung, welche mit AS2 eingeführt wurde, wird nun bei der Kompilierung nicht einfach verworfen, sondern bleibt komplett erhalten, wodurch der Einsatz bestimmter und sorgfältig gewählter Datentypen eine Notwendigkeit darstellt.
In AS2 war es möglich über

```
var i=0;                //Erzeugung Variable i, setzen der Variable auf 0
i="meinText";           //Zuweisung von Text auf Zahlenvariable
```

eine Variable mit einem Zahlenwert zu erzeugen und diesen anschließend mit Werten vom Typ Text zu belegen, da AS2 die lose Typisierung weiter unterstützte, obwohl es auch möglich war, Variablen eines bestimmten Typs anzulegen, welcher dann aber zur Kompilierungszeit nicht übernommen sondern ignoriert wurde.

Damit kam ein weiterer Vorteil der strikten Typisierung, die Reduzierung des Speicherplatzbedarf durch Auswahl von Datentypen mit möglichst geringer Größe, nicht zum Tragen, da für die Variablen Speicherplatz von einheitlicher Größe reserviert wurde. Actionscript 3.0 verlangt nun aber explizit die strikte Typisierung, Variablen müssen über

```
var i:Number=0;              //Erzeugung einer Variable vom Typ Number
var str:String="meinText";   //Anlegen einer String-Variable
i=str;                       //Fehler, Number kann kein Text zugewiesen werden
```

unter Angabe eines konkreten Typs erzeugt werden und behalten diesen Typ, so lang sie existieren. Eine Variable vom Typ Boolean benötigt nun wirklich nur den Speicherplatz um die Werte 1 und 0 darzustellen, über eine durchdachte Wahl des jeweiligen Datentyps kann der Speicherbedarf minimiert werden.

Eine weitere Neuerung betreffend dem Sprachkern ist das Setzen undeklarierter Variablen auf den Wert "null", das vorher benutzte "undefined" wurde ersetzt, zumal damit nicht erkennbar war, ob eine Variable nicht existierte oder nur nicht zugewiesen war.

Auch der Zugriff auf die Elemente wurde überarbeitet, so wurde "_root" durch "root" ersetzt, auch die Benennung verschiedener Attribute wie z.B. "element._x" wurde durch weglassen der Unterstriche zu "element.x".

Neue Funktionen und Sprachelemente

Fehlerbehandlung ist nun auch in Actionscript mit try-catch-Anweisungen umfassend realisierbar.

Laufzeitfehler können somit über die Einbindung der Codezeilen

```
try
{
    //Anweisung, welche auf Fehler überprüft werden soll
}
catch(error:Error)
{
    //Fehlerbehandlung
}
```

abgefangen und behandelt werden.

Des Weiteren wurde AS3 um eine for-each-Schleife bereichert, Arrays können nun auf das Vorhandensein von Variablen eines bestimmten Datentyps durchsucht und mittels der Anweisung

```
var myArray:Array={"ersterString","zweiterString","dritterString");
for each(var str:String in myArray)
{
    trace(str);
}
```

ausgelesen bzw. ausgegeben werden, ohne Einsatz von Indexvariablen und Typvergleichen.

Eine wichtige Erweiterung ist auch die Möglichkeit, reguläre Ausdrücke zu verwenden, über welche Gültigkeitsüberprüfungen von Eingaben einfach und zuverlässig vorgenommen werden können.

Soll eine Variable auf Gültigkeit entsprechend einer Regel untersucht werden kann dies unter Verwendung eines Objekts der RegExp-Klasse über

```
var meineBanane:String="banane";
var meinBananenbaum="bananenbaum";
var myReg:RegExp=/b(an)+e/;                 //Festlegung des regulären Ausdrucks
trace(myReg.test(meineBanane));             //Ausgabe "true"
trace(myReg.test(meinBananenbaum);          //Ausgabe "false"
```

geschehen.

Auch der Zugriff auf XML-Dateien wurde unter Verwendung von E4X objektorientierter gestaltet, die Navigation erfolgt nun mittels

```
xmlSet=new XML(xmlLoader.data);    //XML mit den Daten des URLLoaders anlegen
xmlList=new XMLList(xmlSet.children());      //Kindelement aufrufen
var s:String=xmlList[i].attribute("name");       //Attribut "name" auslesen
```

über die Kindelemente bzw. durch die einzelnen Elemente über die Indizierung nach dem Vorbild des Array-Objektes.

Objektorientierter präsentiert sich nun auch das Klassenkonzept, was sich z.B. durch die Möglichkeit der Paket-Bildung und die Einführung von Namensräumen, so genannte Namespaces, zeigt.

Eine Klasse kann nun über den generellen Aufbau

```
package myPackage
{
      public namespace Deutsch;
      public namespace English;
      class myClass
     {
          English function HelloWorld() {trace("Hello World");}
          Deutsch function HelloWorld() {trace("Hallo Welt");}
          function myClass() {}
      }
}
```

erstellt und einem Paket zugewiesen werden. Eine Koexistenz gleichnamiger Funktionen innerhalb einer Klasse ist durch Verwendung von Namensräumen möglich, durch Anlegen einer Instanz dieser Klasse und Angabe des zu verwendenden Namensraums wie z.B.

```
import myPackage;                          //Importieren des Paketes
var myVar:myClass=new myClass();   //Anlegen einer neuen Instanz
use namespace Deutsch;                //Festlegung des Namensraums
myVar.HelloWorld();                        //Ausgabe "Hallo Welt"
use namespace English;                //Umschalten des Namensraums
myVar.HelloWorld();                        //Ausgabe "Hello World"
```

kann nun auf die zum Namensraum gehörende Funktion zugegriffen werden.
Des Weiteren wurde die Ereignisbehandlung dahingehend vereinheitlicht, dass es nur noch über Eventlistener möglich ist, Ereignisse abzugreifen. Bereits in AS2 wurde das Eventlistener-Konzept für Klassen und Komponenten eingeführt, nun muss es den Elementen explizit zugewiesen werden. Dies geschieht mit der Funktion addEventListener(Eventtyp, Eventhandler), wodurch jede Komponente auf jedes Ereignis reagieren kann, für welches ihr ein entsprechender Eventlistener zugewiesen wurde.

Es kann z.B. dem Klick-Ereignis eines MovieClip über den Aufruf

```
function myFunction(event:MouseEvent)
{
	trace("KLICK!");
}
myMovieClip.addEventListener(MouseEvent.CLICK, myFunction);
```

die entsprechende Funktion zugewiesen werden.

Alle Klassen wurden komplett überarbeitet, nun bildet das DisplayObject den Ausgangspunkt, nicht wie in vorherigen Versionen die Klasse MovieClip. Die Regelung der Tiefen hinzugefügter Elemente übernimmt sich nun der Interpreter, jedes hinzugefügte Objekt wird eine Ebene über dem zuvor hinzugefügten Element platziert, über Funktionen wie swapChildren(...) können untergeordnete Objekte jedoch getauscht werden. Weitere neue Funktionen bieten sich vor allem beim Zugriff auf serverseitige Daten, so werden Bilder und SWFs nicht länger über die Funktion load(...) sondern über einen Loader geladen. Um z.B. eine XML zu laden wird dann der URLLoader eingesetzt, über die Codezeilen

```
var xmlLoader:URLLoader=new URLLoader();
xmlLoader.load(new URLRequest("meineXML.xml"));
var xmlSet:XML=new XML(xmlLoader.data);
```

wird ein URLLoader-Objekt erzeugt, über eine URLRequest-Instanz die XML-Datei geladen und anschließend dem XML-Objekt zugeordnet.

Im Allgemeinen ist Actionscript 3 umfangreicher und reifer geworden, lassen sich doch z.B. Rohdaten aus Sounddateien auslesen, in Videoclips Haltemarken, so genannte Cuepoints, setzen und behandeln, Bitmaps filtern und überblenden sowie Tweens per Actionscript erstellen.

Letztendlich bietet auch die Timer-Klasse das objektorientierte Äquivalent zur setInterval-Methode, ein Timer-Objekt wird jetzt mittels

```
var myTimer:Timer=new Timer(100);    //Erzeugung, Intervall=100ms
myTimer.addEventListener(TimerEvent.TIMER, myFunction);
myTimer.start();
```

erzeugt, bekommt eine entsprechende Funktion zur Ereignisbehandlung zugewiesen und wird gestartet.

Weitere Neuerungen

Durch die konsequente Durchsetzung des objektorientierten Konzeptes funktionieren auch Zuweisungen und damit Referenzen ohne Probleme. Jetzt ist es möglich, z.B. auch undeklarierten Movieclips einen bestehenden Movieclips zuzuweisen, was noch in AS2 nicht möglich war. Auch taucht in AS3 das Scoping-Problem nicht länger auf, welches sich daraus ergab, dass Attribute erst beim Auslösen eines Events interpretiert und damit als Bestandteil des Auslöserobjektes angesehen wurden.

Wenn einem Objekt A die Eventbehandlungsfunktion eines anderen Objektes B zugeordnet wurde, um diese zu verknüpfen, z.B. über

```
class MYCLASS
{
      private var myName:String;
      function MYCLASS(objektA: MovieClip,theName:String)
      {
            myName=theName;
            objektA.onClick=showName(); //Zuweisung Objekt-Funktion zu MovieClip
       }



       public  function showName()
       {
            trace(this.myName);            //Ausgabe der Objekteigenschaft myName
       }
}
...
var objektB:MYCLASS=new MYCLASS(objektA, "MeinNameIstObjektB");
```

wurde beim Klick auf den MovieClip die Funktion showName() aufgerufen, jedoch wurde sie innerhalb des MovieClip-Objektes interpretiert, welches kein Attribut namens myName besitzt, da dieses Bestandteil des Objektes B ist. Somit wurde zur Laufzeit der Wert "undefined" ausgegeben bzw. Fehler produziert, unter AS3 funktioniert dies nun problemlos, da dem Objekt direkt die Funktion der jeweiligen Instanz zugeordnet wird.

2.2 Streaming

In Zeiten immer schneller werdender Internetverbindungen bietet sich die Möglichkeit, Filme in ansprechender Qualität über das Internet auf dem heimischen PC anzuschauen. Das Stichwort lautet Video-On-Demand, dabei entscheidet der Nutzer, wann er welchen Film anschauen möchte. Streaming bietet dabei die Möglichkeit, auch in diesen Videodateien, welche sich nicht auf dem heimischen PC befinden sondern auf dem Server des Video-On-Demand-Anbieters, in gewohnter Weise zu spulen und an beliebige Stellen zu springen. Der Vorteil von Streaming ist dabei, dass sich der Videodatenstrom genau wie eine auf dem PC befindliche Datei verhält, ohne dass der komplette Film vorher heruntergeladen werden muss, was wiederum bei Vorschaufunktionen bzw. beim Verschaffen eines Überblickes über den Inhalt von großer Bedeutung ist. Für die Umsetzung des "Virtuellen Rundgangs" wird genau diese Funktionalität benötigt, um innerhalb des Videomaterials navigieren zu können. Actionscript 3.0 bietet für diese Zwecke Klassen, welche für das Streaming zuständig sind.

Audiostreaming

Zunächst wäre da die Sound-Klasse zu erwähnen, welche die Sounddateien verarbeitet, den Datenstrom liefert dabei die Klasse URLRequest. Über die load()-Funktion des Sound-Objektes wird mittels URLRequest-Objekt die externe Datei geladen. Mittels Steuerungsfunktionen wie z.B. play() bzw. pause() kann nun Einfluss auf den Abspielvorgang genommen werden. Leider unterstützt AS3 als externes Dateiformat lediglich MP3, andere Formate wie WAV, AIFF und QuickTime können lediglich über Flash importiert und in den fertigen Film integriert werden, Streaming ist nicht möglich.

Videostreaming

Für Videos hält AS3 die FLVPlayback-Komponente bereit, welche grundsätzlich Streaming unterstützt, dabei muss lediglich dem source-Parameter der FLVPlayback-Instanz ein Wert vom Typ String mit dem Pfad und entsprechenden Dateinamen zugewiesen werden. Vorweg ist zu sagen, dass bei der Auslieferungsversion von Flash CS3 für das Videostreaming lediglich das eigene FLV-Format Unterstützung findet, andere Formate wurden zuerst ignoriert, nach einem Update des Flash Players auf die Version 9.0.115.0 lassen sich dann aber auch Videos nach dem H.264 Standard abspielen. Trotzdem sind die Einsatzmöglichkeiten begrenzt, da andere weit verbreitete Formate wie z.B. WMA vor dem Einsatz in ein entsprechend unterstütztes Format umgewandelt werden müssen, jedoch hat sich das FLV-Format nicht zuletzt durch Webseiten wie YouTube, welche das Format nutzen, bereits fest etabliert und auch die Einsatzmöglichkeit von MP4 gleicht diesen Nachteil wieder aus.

Die Steuerung des mit der FLVPlayback-Komponente realisierten Datenstroms gestaltet sich recht unproblematisch, durch Steuerungsfunktionen, auf welche später eingegangen wird, kann der Abspielvorgang beeinflusst und über Spulfunktionen an eine gewünschte Stelle bzw. zu einem angegebenen Zeitpunkt gespult werden. Über eingebaute Suchfunktionen können bestimmte Zeitmarken,, so genannte Navigations-Cuepoints, welche zuvor in die FLV-Datei eingebunden wurden, angesteuert werden. Wenn zum Zeitpunkt des Erstellens der FLV-Datei keine Haltemarken gesetzt wurden, kann dies über das hinzufügen von Actionscript-Cuepoints zur Laufzeit erfolgen, welche z.B. aus einer Datenbank oder einer XML-Datei ausgelesen werden könnten. Diese werden als Instanz der dynamischen Klasse Object angelegt und mit Werten für die time- und name-Parameter belegt, wobei weitere Parameter nach Belieben hinzugefügt, sowie über die Funktion addASCuePoint(...) an die bestehende FLVPlayback-Instanz gebunden werden können.

Da diese Cuepoints, anders als die Navigations-Cuepoints, nicht in den Metadaten des Films hinterlegt werden, kann es durchaus zu Ungenauigkeiten beim Auslösen des entsprechenden Cuepoint-Ereignisses kommen, Cuepoints werden oftmals etwas zu früh oder zu spät aufgerufen. Ein Cuepoint an der Stelle 2 Sekunden wird damit manchmal bei 1,8 Sekunden und beim nächsten Mal bei 2,1 Sekunden ausgelöst. Diese so klein erscheinende Diskrepanz wird zu einem größeren Problem, wenn es das Spulen, ausgelöst bspw. durch Auswahl eines Cuepoints in einer Liste, zu jener Stelle betrifft, denn AS-Cuepoints werden dabei regelmäßig übergangen und unverarbeitet zurückgelassen, denn ein Cuepoint kann früher oder später verarbeitet werden und wird somit oftmals, auch bedingt durch die Verarbeitung mit und durch Actionscript sowie der Kodierung des Videomaterials und Verbindungsabbrüchen, übersprungen. Auch durch Spulen an einen Zeitpunkt, der kurz vor dem Haltepunkt liegt, bringt keine Lösung, Actionscript arbeitet oftmals so ungenau, dass der Cuepoint trotzdem ignoriert wird, in anderen Fällen funktioniert es fast punktgenau. Letztendlich muss mit nach dem Standard kodiertem Videomaterial ein Vorlauf von 2 Sekunden eingeräumt werden, um eine konsistente Lösung zu schaffen, es sei denn, man arbeitet mit Navigations-Cuepoints anstelle von AS-Cuepoints und bindet diese an die Videodatei, was aber bei Änderungen eine Neukodierung des Videomaterials erfordert.

Als Alternative zur FLVPlayback-Klasse kann der Datenstrom durch die Klassen NetConnect und NetStream gesteuert werden. NetConnect beschränkt sich dabei auf die Möglichkeit, Datenströme vom gleichen Server abzurufen, anschließend wird diese NetConnection an eine neue Instanz der Klasse NetStream übergeben, welche dann über die bereits erwähnten play()-, stop()- und pause()-Befehle gesteuert werden kann. Im Anschluss daran wird der NetStream an eine Instanz vom Typ Video übergeben und kann so dargestellt werden.

Diese Komponenten bieten jedoch keinen wirklichen Vorteil zur FLVPlayback-Komponente, solang es um die Integration von Videomaterial geht, da sie zum Einen komplizierter einzusetzen sind, schon aufgrund der Tatsache, dass man für die Verwendung 3 Klassen benötigt, zum Anderen weil z.B. auch die Steuerungsmöglichkeit mittels Cuepoints nachträglich über den client-Parameter des NetStreams nachgerüstet werden muss, da Haltepunktbehandlung standardmäßig nicht vorhanden ist. Dafür wird ein Listener-Objekt vom abstrakten Typ Object angelegt, diesem wird wiederum ein Attribut onCuePoint hinzugefügt und mit der Behandlungsfunktion gleichgesetzt, anschließend erfolgt die Anbindung an das NetStream-Objekt, welches dann wie folgt an das Video-Objekt gebunden wird:

```
var listenerClient:Object = new Object();           //Listener-Objekt erzeugen
listenerClient.onCuePoint = cuePointHandler;   //Eventhandler zuweisen

var myVideo:Video = new Video();                    //Anlegen des Videoobjektes
var nc:NetConnection = new NetConnection();   //Anlegen der Verbindung
nc.connect(null);
var ns:NetStream = new NetStream(nc);      //Anlegen des Datenstroms
ns.client = listenerClient;                         //Einbinden des Cuepointlisteners
myVideo.attachNetStream(ns);                        //Datenstrom ans Videoobjekt binden
ns.play("myMovie.flv");                             //Laden und Abspielen der Datei
```

2.3 Codecs

Im Gegensatz zu vorhergehenden Versionen Unterstützt der neue Flashplayer auch den Codec H.264, auch MPEG-4/AVC genannt, welcher die erforderliche Datenrate bei gleichbleibender Qualität mindestens halbiert und dessen Anwendungsgebiet sich vom portablen Einsatz bis hin zur Verwendung von HDTV erstrecken soll. Im Gegensatz zur MPEG2-Kodierung wird die Datenrate bei gleicher Qualität sogar auf ein Drittel reduziert, die Kodierungszeit verlängert sich allerdings auf die zwei- bis dreifache Zeitspanne.

Während der Entwicklung von H.264 ergaben sich durch Abspaltung auf dem gleichen Ansatz basierende aber untereinander inkompatible Formate, wie der Sorensen-Video-3-Codec und das Windows-Media-Video9, welches das zur Zeit bevorzugte Format vom Entwickler Microsoft ist und somit Einzug in die neueren Windows-Mediaplayer Versionen gefunden hat. Änderungen von H.264 zu den Vorgängern des MPEG-Formates sind unter anderem eine Integertransformation von 4x4 Blöcken, welche im Gegensatz zu der vorher verwendeten diskreten Kosinustransformation von 8x8 Blöcken steht. Subtraktionen, Additionen und binäre Verschiebungen stellen die Grundlage der Integertransformation dar, was durch die einfache Möglichkeit der Hardwareimplementierung einen enormen Geschwindigkeitsgewinn bedeutet. Durch den Einsatz der 4x4 Blöcke wird die Ringing-Artefaktbildung, auch als Gibbssches Phänomen bekannt, deutlich reduziert, was wiederum einen Qualitätsgewinn bedeutet. Auch die Entropiekodierung wurde auf die neue Situation eingestellt und unterstützt nun auch eine leistungsfähigere arithmetische Kodierung, im Gegensatz des bisher verwendeten und auf der Huffmann-Kodierung aufbauenden VLC-Coding. Die Möglichkeit der Makroblock-Partitionierung von 16x16 auf 4x4 Pixel bedeutet präzisere Bewegungsvektoren, welche auch komplexes Verhalten und damit Bewegungen, die vorher missgedeutet wurden und somit Fehler generierten, erfassen können. Des Weiteren wurde die Bewegungsvorhersage, bekannt als Prediction, mit Intra-Prediction und Long-Term-Prediction erweitert, dabei steht die Intra-Prädiktion für die Vorhersage in so genannten I-Frames, wobei Pixelwerte eines Blockes aus seiner Umgebung abgeschätzt und lediglich der Fehler gespeichert wird. Durch Einsatz der Langzeitvorhersage, englisch Long-Term-Prediction, können P- und B-Bilder Referenzen auf beliebig viele vorhergehende P- und I-Frames enthalten, was besonders bei periodischen Bildinhalten eine deutliche Verbesserung darstellt.

Aus- und Überblendungen von Videoinhalten lassen sich durch die gewichtete Vorhersage (Weighted Prediction) zudem mit wenig Aufwand realisieren, dabei werden Referenz-Bilder nach Belieben gewichtet und bilden so einen Übergang.

Eine Qualitätsverbesserung entsteht ebenfalls durch den Einsatz eines Deblocking-Filters, wobei bereits dekodierte Bilder in Verbindung mit schon gefilterten Bildern gesetzt werden, was wiederum eine deutliche Verbesserung der subjektiven Qualität mit sich bringt. Unter Verwendung verschiedener Profile können Merkmale wie die flexible Makroblockanordnung, unterschiedliche Chrominanz-Auflösungen sowie verlustlose Kodierung je nach Bedarf gewählt werden. Diese Profile entscheiden zusammen mit den Qualitätsstufen über die erforderliche Bandbreite und das Endergebnis. Unterteilt werden die Qualitätsstufen in Levels, Stufe 1 steht für eine Auflösung von 128x96 Pixeln bei 30,9 Bildern pro Sekunde, die Datenraten reichen je nach gewähltem Profil von 64 kbit/s bis 256 kbit/s. Die maximal verwendbare Einstellung für den Level-Parameter ist das Level 5.1, welches je nach Profil Datenraten von 240 Mbit/s bis 960 Mbit/s und Auflösungen von 1920x1080 bei 120 Bildern pro Sekunde bzw. 4096x2048 bei 30 Bildern pro Sekunde generiert. Somit bietet der H.264-Codec eine breite Palette von Anwendungsmöglichkeiten sowie subjektiv (in gewissem Maße auch objektiv) gute Qualität bei niedrigen Übertragungsraten. Natürlich unterstützt Flash auch weiterhin die bisher eingesetzten Codecs

- Sorensen, eine Variante des als H.263 bekannten Standards
- und VP6.

Dabei ist Sorenson jenes Format, welches auch schon in der Apple QuickTime Software Verwendung fand und 2002 seinen Einzug in die Flash Version MX vollzog, VP6, entwickelt von On2, wurde dann im August 2005 zum Standardformat sowohl für Flashplayer als auch für das FLV-Format.

Als Audiocodec kam bisher das MP3-Format zum Einsatz, nicht zuletzt durch seine weite Verbreitung, den relativ niedrigen Bitraten sowie der Verfügbarkeit entsprechender Kodiersoftware.

Als Soundformat für den seit Flashplayer 9.0.115.0 enthaltenen Codec H.264 bietet sich nun die Möglichkeit, neben MP3 auch den MPEG-4 High Efficiency Advanced Audio Codec, kurz HE-AAC, zu verwenden, welches auf MPEG-4 Advanced Audio Codec basiert, jedoch um die Funktion der Spektralband-Replikation erweitert wurde. Bei der SBR werden Frequenzanteile, welche über einer bestimmten Frequenz liegen, nicht kodiert, vielmehr werden diese Frequenzen aus tieferen und mittleren Signalen über Steuersignale aufgebaut, da vor allem die Hüllkurve für den Eindruck eines Signals von Bedeutung ist und diese sich bei tiefen, mittleren und hohen Frequenzen stark ähneln. Weiterhin nutzt der Codec die Tatsache, dass hohe Frequenzen vom menschlichen Gehör schlechter wahrgenommen werden und Sinuswellen nicht von schmalbandigem Rauschen unterschieden werden können. Die Grenzfrequenz richtet sich nach der gewählten Qualität, SBR bringt vor allem bei niedrigen bis mittleren Datenraten und niedriger Qualität einen enormen subjektiven Qualitätsgewinn. Bei HE-AAC liegt die Grenzfrequenz bei 5 kHz, zwischen 5 und 15 kHz wird die SBR angewendet. Somit kann auch über schmalbandige Verbindungen oder portable Anwendungen eine gute Qualität gewährleistet werden, bei Signalen mit hohen Datenraten bzw. hoher Qualität bringt der Einsatz von HE-AAC keine nennenswerten Vorteile gegenüber dem Einsatz des MP3 Formats.

2.4 Steuerungsmöglichkeiten

Wie bereits im Kapitel 2.2 über Streaming erwähnt, bietet die FLVPlayback-Komponente von AS3 viele Möglichkeiten der Videosteuerung. Einfache Videosteuerung mittels play()-, pause()-, und stop()-Funktionen der Klasse kann mit nur wenigen Codezeilen implementiert werden und bedürfen aufgrund der eindeutigen Bezeichnung wohl keiner näheren Erklärung, mit Parametern wie autoPlay und autoRewind lässt sich das Verhalten des Videos verändern. Über die Suchfunktionen seek(...) und seekSeconds(...) lässt sich bequem an die in Sekunden angegebene Zeit springen, seekPercent() dagegen spult das Video auf die gewünschte Prozentzahl der Gesamtlänge. Zudem gibt es weitere Methoden, welche für das Auffinden von Haltepunkten, den bereits erwähnten Cuepoints, verantwortlich sind. Da wären zuerst die Methoden

- seekToNavCuePoint()
- seekToNextNavCuePoint()
- und seekToPrevNavCuePoint()

durch deren Einsatz zu einem bestimmten bzw. dem nachfolgenden und dem davor befindlichen Navigations-Haltepunkt gesprungen werden kann. Mit den Funktionen

- findCuePoint(...)
- findNearestCuePoint(...)
- und findNextCuePointWithName(...)

können Haltpunkte über ihren Namen, ihre Zeit oder den zeitlich naheliegendsten Haltepunkt aufgefunden werden.

Wie bereits erwähnt, gibt es in AS3 mehrere Arten von Haltepunkten:

- Navigations-Cuepoints
- Actionscript-Cuepoints
- und Ereignis-Cuepoints.

Die Navigations-Cuepoints werden beim Erstellen der FLV-Videodatei eingebettet, dienen wie der Name schon sagt der Navigation und werden in den Metadaten der Datei hinterlegt, was wiederum bedeutet, dass sie nicht nachträglich per Actionscript eingefügt oder geändert werden können, somit bei Änderungen eine komplette Neukodierung des Videomaterials erforderlich ist. Dafür wurden die Actionscript-Cuepoints vorgesehen, welche als einfache Instanzen vom Typ Object mit den Parametern name, type und time erstellt und mittels addASCuePoint(...) zum geladenen Video hinzugefügt werden.

Leider sind für diese Cuepoints Funktionen wie seekToNavCuePoint(...) und andere, direkt für Navigations-Cuepoints entwickelte Prozeduren nicht einsetzbar. Doch das größere Problem stellen die bereits erwähnten Ungenauigkeiten bei der Behandlung der Actionscript-Cuepoints dar, welche bild- bzw. sekundengenaue Sprünge unmöglich machen. Dies kommt bei Navigations-Cuepoints nicht vor, was durch die direkte Einbettung in das Video zu erklären ist. Ereignis-Cuepoints bieten die gleiche Funktionalität wie die Actionscript-Cuepoints, werden jedoch, wie die Navigations-Haltepunkte, bereits beim Erstellen des Films in die Metadaten integriert, ein Ereignis-Haltepunkt ist die im Video hinterlegte Variante des Actionscript-Haltepunktes. Beide Cuepoints lösen beim Erreichen der jeweiligen Zeitmarke das Ereignis CUE_POINT aus, welches sich durch Eventhandler behandeln lässt. AS3 bietet damit viele, sofort ohne großen Aufwand einsetzbare Steuerungsmöglichkeiten.

Durch den Einsatz von Eventhandlern kann zudem eine vollautomatisierte Ereignisbehandlung aufgebaut werden, welche sich zwar in der Entwicklung zu befinden scheint, aber z.B. auch Sprünge im Video und somit Interaktionsmöglichkeiten mit dem Film und dem zu steuernden Datenstrom bietet.

2.5 Vorteile des Einsatzes von Actionscript 3.0

Durch die vielen Jahre, die sich Flash im Einsatz befindet und die daraus resultierende Akzeptanz in der Anwenderschaft kann man Flash als Plugin schon fast voraussetzen, nach der Veröffentlichung einer neuen Flash-Player Version liegt die Verbreitung bereits nach wenigen Monaten laut Wikipedia bei 90% aller Anwender, Adobe spricht sogar von 99%. Man kann sagen, Flash und sein SWF-Format haben sich etabliert und zu einer Art Standard für interaktive Webanwendungen entwickelt. Ein Grund dafür kann durchaus in der guten Performance liegen, welche aus der Konzeption als Compiler-Sprache resultiert und der erzeugte Binärcode nicht zur Laufzeit übersetzt bzw. interpretiert werden muss. Die einfache Einbindung in bestehende Webseiten lässt auch die Möglichkeit offen, betagte Webseiten mit dynamischen Inhalten zu bereichern, die für jeden zugänglich sind, der den Flash-Player installiert hat. Des Weiteren bietet Actionscript und somit Flash gute Streaming-Möglichkeiten, zum einen des weit verbreiteten MP3 Formats und zum anderen des hauseigenen FLV-Formats, welches durch den H.264-Codec erweitert und in seiner Leistungsfähigkeit enorm gesteigert wurde. Weitere vorgefertigte Klassen zur Arbeit mit XML und die einfache Arbeit mit dem vorhandenen Dateisystem erleichtern dem Implementierenden bei der Actionscript-Programmierung die Arbeit. AS3 bietet dem Benutzer eine große Menge an vorgefertigten Klassen und Funktionen, welche aufeinander abgestimmt sind und von geometrischen Objekten auf Vektorbasis bis hin zur fertigen Videoplayer-Komponente, welche mit wenigen Zeilen Code integriert werden kann, reichen.

Doch der grundlegende Vorteil von AS3 / Flash ist die einheitliche Technologie, beide kommen vom gleichen Entwickler, sind aufeinander abgestimmt und kein Technologien-Mix wie z.B. AJAX, bei dem unterschiedlichste Techniken wie JavaScript, HTML und XMLRequest-Objekt zum Einsatz kommen und verschieden Sprachen beherrscht und aufeinander abgestimmt werden müssen.

2.6 Nachteile des Einsatzes von AS3

Trotz der vielen genannten Vorteile sollen auch die Nachteile von AS3 und damit auch die von Flash erläutert werden. Problematisch gestaltet sich für erste Schritte und das Testen von AS3, dass Actionscript an Flash gebunden ist, man kann keine reine Actionscript-Applikation implementieren und übersetzen, ohne Flash installiert zu haben. Selbst vollkommen autark agierender AS3-Code muss zumindest im ersten Bild des Flash-Movies instanziert werden, ohne Instanzierung besteht keine Möglichkeit die angelegten Actionscript-Klassen zu kompilieren. Wie jede vorkompilierte Programmiersprache zeigt sich neben dem Geschwindigkeitsvorteil natürlich auch die Unflexibilität gegenüber Änderungen, denn das übersetzte Programm, bestehend aus Binärcode, kann im Nachhinein nicht schnell z.B. direkt auf dem Server geändert werden. Man benötigt den Quellcode, welcher bei jeder Änderung neu kompiliert werden muss, denn das Design und die Logik werden in der gleichen Datei unveränderbar festgelegt, es sei denn es werden Vorbereitungen zur Auslagerung getroffen. Auch sind die Möglichkeiten der Vereinheitlichung des Designs wie bspw. mittels CSS begrenzt, da nur einzelne Komponenten wie das TextField überhaupt Cascading-Style-Sheets verstehen, dies dann nur in begrenzter Form, da nicht alle Attribute verstanden werden.

Somit ist der Programmierer/Designer bei Änderungen im Design der Webseite oder des Flash-Films darauf angewiesen, selbst zentrale Möglichkeiten zu schaffen, mit denen sich Designvorlagen mit Schriftgröße, Form, Farbe etc. anlegen lassen, dies könnte z.B. eine XML-Datei oder eine eigene AS-Klasse sein. Die Ungenauigkeit bei der Einhaltung der Cuepoints bzw. die Tatsache, dass wenn die Abspielzeit nicht mindestens 2 Sekunden vor das gewollte Ereignis gesetzt wird, der Haltepunkt einfach übergangen wird, stellt eine Ungenauigkeit dar, die es bei heute mit so hohen Taktraten arbeitenden Prozessoren nicht mehr geben sollte. Dies liegt jedoch nicht an der Verarbeitungsgeschwindigkeit von Flash oder Actionscript, vielmehr wird bei der Kodierung des Videomaterials nur alle 2 Sekunden ein Schlüsselbild gesetzt, das Setzen eines Schlüsselbildes aller 5 Bilder würde den notwendigen Vorlauf auf 0,2 Sekunden verringern, jedoch steigt damit auch die Dateigröße enorm.

Ärgerlich ist weiterhin, dass das objektorientierte Konzept nicht bis ins Detail durchgesetzt wurde und dass somit z.B. die Cuepoints keine eigene erweiterbare Klasse besitzen sondern als abstrakter Typ Object verwendet werden und somit der Aufbau von selbstkontrollierenden und im Funktionsumfang angereicherten Haltepunkten unmöglich ist. Selbst Klassen, die Object erweitern, werden nicht wie Klassen behandelt, sondern von der FLVPlayback-Komponente intern umgewandelt und auch nur als Instanz der Klasse Object zurückgegeben, angehängte eigene Klassen werden ebenfalls in den gleichen Typ umgewandelt.

3. Silverlight 1.0

Seit kurzem bietet Microsoft auf seinen Download-Seiten die Option, anstatt der bisherigen Variante nun auch die Silverlight-Version der Internetpräsenz zu nutzen, und stellt bei Bedarf gleich den passenden Player dafür bereit. Nach dem Herunterladen des Browser-Plugins zeigen sich nun Seiten, die etwas an mit Flash entworfenen Seiten erinnern, es wurde zusätzliches Augenmerk auf das Design gelegt, Elemente öffnen sich und können geschlossen werden, alles wirkt dynamischer. Die Veröffentlichung von Microsofts neuer Technologie, liegt bereits über ein Jahr zurück, schon im April 2007 wurde Silverlight der Öffentlichkeit vorgestellt, ein browser-, plattform- und geräteübergreifendes Plugin, welches die nächste Generation der .NET-basierten und Richmedia-verarbeitenden Web-Anwendungen mit sich bringen soll. Vorher wurde das neue Microsoft-Kind unter dem Name "WPF/E" entwickelt, was wiederum für "Windows Presentation Foundation/Everywhere" steht. Silverlight ist daher der Versuch, das WPF-Konzept überall verfügbar zu machen und ist damit eine stark im Funktionsumfang reduzierte Version von WPF, welches für die grafische Oberflächengestaltung sowie Medien und deren Darstellung in Browsern entwickelt wurde. Sowohl WPF als auch WPF/E basieren auf dem unter dem Namen XAML bekannten Beschreibungsformat. Microsoft versucht somit, WPF auch auf anderen Plattformen als der Windows-Produktreihe zu etablieren, begibt sich nun auf das Terrain von Flash und dessen Derivaten, welche bereits seit Jahren erfolgreich mittels Browser-Plugin Interaktion, grafische Oberflächen sowie Animationsmöglichkeiten in Webseiten einbinden und heute fast auf jedem Internet-PC installiert sind. Zur Entwicklung von Silverlight-Anwendungen stellt Microsoft das Silverlight SDK bereit, welches kostenlos heruntergeladen werden kann. Um die Entwicklung zu erleichtern ist eine Integration in Visual Studio 2005 mittels mitgelieferten Template möglich, alternativ bietet Microsoft auch Entwicklungswerkzeuge wie die Software Expression Blend an, welche stark an Flash erinnert.

3.1 Das Konzept von Silverlight

Beim genaueren Hinsehen zeigt sich, dass Silverlight nicht so neu ist, wie es vorgibt zu sein. Es ist keineswegs eine reine Neuentwicklung, vielmehr stellt Silverlight eine Weiterentwicklung dar, welche neu verpackt und unter neuem Namen präsentiert wird. Das Grundkonzept ist bereits einige Zeit bekannt, es handelt sich hierbei, wie bereits erwähnt, um WPF, die "Windows Presentation Foundation", diese benutzt XAML um den Aufbau und das Design der interaktiven Web-Anwendung zu beschreiben. Die Integration erfolgt über JavaScript-Code, welcher in die HTML-Seite eingebunden wird und mittels Silverlight-Plugin im Browser interpretiert wird, welches für Safari, Internet-Explorer 7 und Mozilla Firefox bereits verfügbar ist. JavaScript regelt in diesem Fall die Logik und die dynamischen Aspekte der Anwendung, XAML beschreibt den Aufbau, vernetzt über Attribute wie z.B. "MouseLeftButtonDown" das jeweilige Objekt mit der dazugehörigen Verarbeitungsfunktion und knüpft damit die Verbindung zwischen Design und Programmlogik. Grundsätzlich besteht eine Silverlight-Anwendung also aus der im Silverlight SDK befindlichen Datei "Silverlight.js", die für die Einbindung in die Webseite verantwortlich ist, der HTML-Seite "Default.htm", welche einen "<DIV>"-Bereich vom Typ "silverlightControlHost" enthält und den Platzhalter für den SL-Inhalt darstellt. Weiterhin benötigt werden zum einen die "Page.xaml", welche die XAML-Beschreibung des Inhalts in sich trägt, und zum anderen die "Page.xaml.js", welche den dazugehörigen JavaScript-Code für die "Page.xaml" bereitstellt. In ihr werden Eventhandler, Variablen und selbstentworfene Funktionen zur Steuerung des mit XAML definierten Inhalts hinterlegt, hier erfolgt die eigentliche Interaktion zwischen Nutzer und Silverlight-Applikation. Das Document-Object-Model sorgt dafür, dass über document.getElementByID("SilverlightControl") auf das Silverlight-Objekt der Seite zugegriffen werden kann und über dieses dann mittels der Abfrage content.findName(<Elementname>) der Zugriff auf das entsprechende SL-Element erfolgt.

3.2 XAML

XAML steht für "eXtensible Application Markup Language" und wurde von Microsoft entwickelt um eine Beschreibungssprache für grafische und interaktive Oberflächen bereitzustellen. Basierend auf dem XML-Format werden Design und Attribute der einzelnen Elemente definiert, gruppiert und Transformationen wie z.B. Skalierungen und Rotationen ausgeführt. Im Gegensatz zu MXML, welches 2004 von Macromedia eingeführt wurde, beherrscht XAML nur die Beschreibung des Designs, über MXML können hingegen auch Strukturen wie Arrays oder Anwendungslogik formuliert werden. Verfügbare Kontrollelemente in Silverlight 1.0 sind

- TextBlock
- Canvas
- Image
- MediaElement
- und Glyphs.

Dazu kommen so genannte Shapes, also die Zeichenobjekte, bestehend aus

- Line
- Rectangle
- Ellipse
- Path
- Polygon

welche benutzt werden können, um komplexere Objekte zu kreieren.

Um diese Formen zu füllen existieren wiederum Brushes, wie die

- SolidColorBrush für einfache Farben
- LinearGradientBrush für lineare Farbverläufe
- RadialGradientBrush für kreisförmig Farbverläufe
- ImageBrush zum Füllen der Objekte mit Bildern
- VideoBrush für Füllungen mit Videoinhalt

welche wiederum über ihre Attribute angepasst werden können. Für jedes Objekt können die Opazität (Opacity), die Erscheinungsform des Mauszeigers beim Überfahren (Cursor) und die Umrandung (Stroke) festgelegt werden. Weiterhin bieten Elemente wie das Image die Möglichkeit, Clipping mittels eines angehängten Clip-Elementes, in diesem Fall durch hinzufügen von "<Image.Clip>" und setzen einer geometrischen Form wie z.B. eines Kreises als Clipping-Vorlage, zu realisieren, dabei wird das Bild-Objekt in Form der geometrischen Vorlage zugeschnitten. Transformationen lassen sich nach dem gleichen Prinzip hinzufügen, einfach indem in einen RenderTransform-Abschnitt, welcher sich innerhalb des Abschnittes des gewünschten Objektes befindet, die gewünschte Transformation definiert wird. Hier bietet XAML folgende Transformationen an:

- RotateTransform für Rotationen
- ScaleTransform für Größenveränderungen
- TranslateTransform für Verschiebungen
- SkewTransform um ein Objekt gleichmäßig entlang einer Achse zu verzerren
- MatrixTransform zur Definition eigener Transformationsmatrizen

Diese können beliebig kombiniert werden, um bspw. eine 3D-Perspektive zu simulieren.

Über den Einsatz von Triggern und Eventhandlern können mit Hilfe der Transformationen relativ leicht Animationen geschaffen werden, indem man im Objekt-Abschnitt den entsprechenden Trigger und dessen Auslöser definiert, diesem ein Storyboard zuweist, welches wiederum die Animationsvorlage enthält.
Animationen können vom Typ

- DoubleAnimation zur Animation von Double-Werten wie z.B. der Opazität
- PointAnimation zur Animation von Kurven- oder Linienpunkten
- ColorAnimation zur Änderung von Farbwerten

sein und gelten immer für ein benanntes Objekt. Verhalten, Dauer, der zu ändernde Parameter und alles weitere werden innerhalb der Animation festgelegt. Weitere Features wie bspw. der Einsatz von "Ink", einer Art Zeichenstift, mit der der Benutzer auf der Silverlight-Oberfläche zeichnen kann, runden das Gesamtpaket ab und vermitteln den Eindruck eines abgerundeten Definitionskonzepts.

Ein einfaches Beispiel für die Kombination eines Elementes mit einer Füllung und einer Transformation wäre der folgende XAML-Abschnitt in welchem ein Rechteck angelegt, dem Füllungsparameter ein ImageBrush-Objekt zugewiesen und das Objekt anschließend gedreht wird.

```
<Rectangle Width="243" Height="178" Stroke="#FF000000" Canvas.Left="65"
Canvas.Top="74" OpacityMask="#FF000000" RenderTransformOrigin="0.5,0.5">
   <Rectangle.Fill>
         <ImageBrush ImageSource="poi.png" Stretch="Uniform"/>
    </Rectangle.Fill>
    <Rectangle.RenderTransform>
          <TransformGroup>
                <RotateTransform Angle="16"/>
         </TransformGroup>
   </Rectangle.RenderTransform>
</Rectangle>
```

Das Ergebnis präsentiert sich dann in folgender Form:

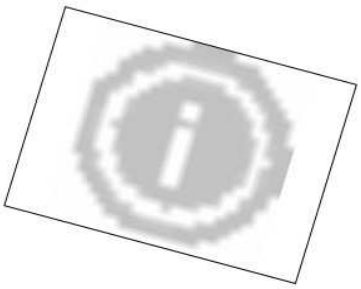

Abbildung 3: Kombination von XAML-Elementen

3.3 Streaming

Für Streaming-Anwendungen stellt Microsoft auf ihrer Internetplattform unter "streaming.live.com" eine kostenlose Hosting-Möglichkeit bereit, auf welcher Videos für den Einsatz abgestellt und mit einer Datenrate von bis zu 700 KB/s als Datenstrom in die Silverlight-Applikation geladen werden können.

Für den Einsatz von Streaming steht unter Silverlight das MediaElement bereit, es wird verwendet um jegliche Art von Ton- sowie Videomaterial wiederzugeben und unterstützt On-Demand-Streaming von Medien, welche auf einem Windows Media Server vorliegen. Dafür muss die entsprechende URI das MMS-Protokoll verwenden, andernfalls werden die Medien progressiv heruntergeladen und wiedergegeben, sobald der Puffer über genügend Material bzw. Vorlauf verfügt, während der Rest der Datei nachgeladen wird. Wird der Download über das HTTP- oder HTTPS-Protokoll durchgeführt, wird zuerst versucht, die Datei progressiv herunterzuladen, sollte das scheitern, wird auf Streaming umgestellt. Streaming lässt sich nicht forcieren, es gibt keinerlei Optionen oder Attribute, um ein solchen Verhalten zu erzwingen.

Jedoch bietet Microsoft, wie bereits erwähnt, einen kostenlosen eigenen Service an, über welchen ausschließlich Streaming möglich ist und man sich somit bei einfachem Einsatz des Silverlight Streaming Service keine Gedanken über diese Problematik machen muss. Da die Anwendung jedoch auf dem Server der HTW Dresden laufen soll, ist diese Möglichkeit nicht praktikabel, auch eine Einrichtung eines zusätzlichen Microsoft Medienservers ist nicht möglich, somit muss für die prototypische Implementierung zwar dieser Service von Microsoft genutzt werden, stellt aber für die finale Anwendung "Virtuelle Bibliothek" keine Lösung dar. Andere Komponenten von Silverlight für diesen Einsatz gibt es nicht, das MediaElement-Objekt stellt Microsofts Universallösung für Streaming dar, ist für den einfachen Einsatz von Streaming vollkommen ausreichend sowie einfach implementierbar. Über Attribute wie AutoPlay kann das Verhalten beeinflusst, über die Größenparameter Width und Height geändert, über Source die Quelle und über Opacity die Durchsichtigkeit eingestellt werden. Leider bietet das Objekt nicht ausreichend viele Parameter, um es auf die persönlichen Bedürfnisse anzupassen, dafür ist es beliebig mit Transformationen und ebenfalls mit Clipping kombinierbar und kann so zumindest optisch verändert und beeinflusst werden [Mor08].

3.4 Codecs

Genauso wie Adobe setzt Microsoft bei den Codecs auf die für die Windows-Plattform entworfenen hauseigenen Video- und Audioformate. Für den Audioeinsatz unterstützt das MediaElement

- Windows Media Audio 7, kurz WMA7
- WMA8
- WMA9
- und MP3

auf welches aufgrund der großen Verbreitung natürlich nicht verzichtet werden konnte. Die WMA-Formate sind, wie auch MP3, verlustbehaftet und mit dem MP3-Verfahren vergleichbar, es unterstützt Datenraten von bis zu 320 KB/s [Mor08]. Bei der Kodierung werden Töne, die von anderen Tönen maskiert werden oder unterhalb der Hörgrenze liegen, somit sowieso keine Relevanz haben, weggelassen und damit Daten gespart, es entsteht eine Audiodatei, welche zwar verlustbehaftet, aber bis zu einem bestimmten Maß akustisch nur schwer von der Originaldatei unterscheidbar ist [www02]. Somit kann relativ gute Qualität bei geringen Datenraten geboten werden, besondere Vorteile bietet dieses Format für den Nutzer letztendlich aber nicht. Bei den Videoformaten zeigt sich die Auswahl noch geringer, denn als Dateiformat werden nur die verschiedenen Variationen des Windows Media Video Formats, kurz WMV, unterstützt. Dies sind grundsätzlich ähnlich dem MPEG-4 Format aufgebaut [Mor08]:

- WMV1
- WMV2
- WMV3
- WMVA, Windows Media Video Advanced Profile (non VC-1)
- und WMVC1, Windows Media Video Advanced Profile VC-1

Die VC-1-Version wurde für den Einsatz auf Blu-Ray-Disc und HD DVD entwickelt [www01]. WMV9 bietet neben einer 720p-HD-Auflösung im Advanced Profile vor allem eine Verbesserung der Qualität zwischen 15 und 50 Prozent im Gegensatz zu Windows Media Video 8, das bedeutet für den Nutzer leichte Qualitätsverbesserungen bei niedrigeren Bitraten und starke Verbesserungen bei hohen Datenraten [www04], doch gerade bei niedrigen Bitraten wäre ein höheres Maß an Verbesserung wichtiger, da gerade dort, im Gegensatz zu der ausreichenden Qualität bei höheren Datenraten, eine Qualitätsverbesserung unter Umständen nötig wäre um den optischen Eindruck zu verbessern. Im Vergleich zu MPEG-2 liegen die Datenraten bei gleicher Auflösung bei ungefähr einem Drittel wobei die Qualität als gleich bleibend bezeichnet werden kann. Des Weiteren besteht die Möglichkeit, einzelne Bilder mittels Zoom- und Schwenkeffekten sowie Übergängen in Videosequenzen zu verwandeln, durch Einsatz dieser Techniken kann die Datenrate weiter gesenkt werden, schon Datenraten ab 20 KBit/s reichen aus, um Datenströme dieses als Windows Media Video 9 Image betitelten Formates zu übertragen. Die Auflösung liegt hier laut Standard bei 640x480 Bildpunkten und gilt als Vollbildmodus. Laut Angaben von Microsoft soll es im Advanced Profile möglich sein, aufgrund der niedrigen Datenraten einen kompletten HD-Film in 720p-Auflösung auf nur einer herkömmlichen CD unterzubringen. Aufgrund der starken Ähnlichkeit zum MPEG-4 Format sind auch die Qualitäten des kodierten Materials ähnlich, WMV bietet keine nennenswerten technischen und optischen Verbesserungen gegenüber jenem Format.

3.5 Steuerungsmöglichkeiten

Die Steuerung der Videodateien und somit auch des Datenstroms erfolgt über JavaScript, da XAML für die Umsetzung des Designs und JavaScript für die Programmlogik verantwortlich ist. Silverlight hält für das MediaElement die Funktionen play(), pause() und stop() bereit, damit lässt sich das Video abspielen, pausieren sowie anhalten und auf den Anfang zurück setzen [Mor08]. Funktionen zur Ansteuerung bestimmter Zeitpunkte wie z.B. das seek() der Actionscript-FLVPlayback-Komponente besitzt das MediaElement nicht, allerdings kann hier über die Eigenschaft Position die Abspielposition gesetzt werden. Allerdings lässt sich die Position nicht einfach bspw. über zuweisen eines Zahlenwertes setzen, denn lediglich das Unterattribut Position.Seconds kann geändert werden, und auch dieses nicht in der Form

```
Video.Position.Seconds=10;
```

da das MediaElement Änderungen ignoriert, so lange dem Position-Attribut nicht direkt ein neuer Wert zugewiesen wird. Die Position muss in einer Variable gespeichert, anschließend das Unterattribut Seconds geändert und die ganze Positionsvariable dem MediaElement-Parameter Position zugewiesen werden [www03]. Silverlight erkennt in diesem Fall die Wertänderung und spult auf den gewünschten Zeitpunkt. Ebenso wie Actionscript bietet Silverlight die Möglichkeit, Zeitlinien-Markierungen einzufügen und auf diese zu reagieren, ähnlich den AS3 Cuepoints. Um auf diese Markierungen zu reagieren muss im MediaElement der Parameter MarkerReached mit der entsprechenden JavaScript-Funktion verbunden werden. Solche TimelineMarker werden in der XAML-Datei innerhalb des MediaElement-Tag gesetzt und besitzen die Eigenschaften Time, Type, und Text. Wie auch bei den FLV-Dateien können diese Markierungen mittels Programmen wie Expression-Media in den Metadaten der Videodatei hinterlegt werden, wenn die alternative Methode des Setzens in der XAML-Datei nicht erwünscht ist [Mor08].

Allerdings ergibt sich daraus wieder das Problem, dass das Videomaterial bei einer Änderung von Zeitlinienmarkierungen komplett neu kodiert werden muss, wobei bei der Speicherung innerhalb der XAML-Datei die Änderungen unproblematisch und sofort in kurzer Zeit vollziehbar sind.

Weiterhin gibt es noch die schlecht dokumentierte Möglichkeit, mittels JavaScript neue Marker zu setzen, allerdings nur beim Öffnen einer Videodatei. Somit besteht die Option des Nachladens aus Ressourcen, welche sich außerhalb des Videomaterials und des Silverlight-Programms befinden, wie XML Dateien von einem fremden Server, aus einer Datenbank ausgelesene Werte für das Setzen der Markierungen sowie das Erstellen eines TimelineMarker durch Nutzereingaben [www04]. Dabei unterscheidet Silverlight nicht zwischen eingebetteten Zeitmarkierungen und nachträglich hinzugefügten bzw. über XAML definierten, alle Markierungen werden als Marker des jeweiligen MediaElements gespeichert und interpretiert. Für genügend Steuerungsmöglichkeiten bezüglich des Datenstroms ist somit gesorgt, wenn auch einige Besonderheiten beachtet werden müssen. Problematisch gestaltet sich auch hier die Genauigkeit der zu setzenden Haltepunkte, denn das TimelineMarker-Objekt unterstützt lediglich die Angabe von Sekunden, eine Eingabe von Millisekunden oder Werten mit Kommastellen ist nicht möglich, daher müssten die Wert zu Ungunsten der Genauigkeit gerundet werden.

3.6 Vorteile des Einsatzes von Silverlight

Ein entscheidender Vorteil von Silverlight gegenüber Flash zeigt sich schon auf den ersten Blick, bereits ohne einen genauen Blick auf den Aufbau und die Technik zu werfen, denn Silverlight stammt aus den Entwicklungslaboren des Weltkonzerns Microsoft. Abgesehen von der Marktdominanz des Softwaregiganten lässt es erahnen, wie schnell die Verbreitung von Silverlight voranschreiten wird, denn in künftigen Versionen der Windows-Betriebssysteme und des Internet-Explorers wird Silverlight sicherlich bereits bei der Auslieferung integriert sein, wie es zuvor mit Microsoft-Technologien wie DirectX geschehen ist, bereits heute wird SL als optionaler Download über den Windows-Update-Dienst angeboten [www01]. Somit ist Silverlight sicher eine ernst zu nehmende Option für die Zukunft, bereits jetzt hat die neue Technologie augenscheinlich eine große Anhängerschaft gewinnen können.

Bei der Entwicklung von SL-Anwendungen fällt vor allem die einfache Änderbarkeit aufgrund des Aufbaus als Interpreter-Sprache auf, denn Änderungen können am Quellcode sofort durchgeführt sowie getestet werden, ohne die Applikation neu übersetzen und hochladen zu müssen. Der Quellcode ist somit immer verfügbar und notfalls direkt auf dem Server änderbar, denn man benötigt keine zusätzlichen Programme außer einem einfachen Editor, um Design bzw. JavaScript-Logik zu ändern.

Für Entwickler könnte Silverlight vor allem interessant sein, da das Silverlight-Development-Kit, kurz Silverlight SDK, bei Microsoft gratis verfügbar ist und alle zum ersten Aufbau einer Anwendung nötigen Dateien enthält, durch Bearbeitung z.B. mit dem Windows-Editor Notepad können schnell und ohne weitere Kosten erste Programme entwickelt werden, allein die im SDK enthaltenen xaml-, js- und html-Dateien reichen aus, um erste Testläufe zu starten.

Des Weiteren zeigt sich Silverlight umfangreich sowie durch seine umfassenden Formatierungsmöglichkeiten flexibel einsetzbar, da die verschiedenen Komponenten und Transformationen miteinander kombiniert werden können. Durch die Trennung von Design und Logik lässt sich die Arbeit von Designer und Programmierer sehr gut verbinden, ohne dass der Programmcode bei der Entwicklung des Designs stört oder umgekehrt. Beides kann getrennt voneinander entworfen und implementiert werden, anschließend werden die Anwendungsteile über die entsprechenden Eventhandler miteinander verbunden.

3.7 Nachteile des Einsatzes von Silverlight

Aspekte, die auf den ersten Blick Vorteile zu sein scheinen, können natürlich auch negative Aspekte mit sich bringen, wie bspw. die Konzeption von Silverlight als interpretative Sprache, denn zugunsten der Änderbarkeit wird dabei auf Performance verzichtet. Zwar wird dieser Unterschied bei einfachen, kleineren Anwendungen schon aufgrund der heutigen Prozessorgeschwindigkeiten nicht zum Tragen kommen, er bleibt dennoch existent und könnte bei größeren Projekten einen leichten Nachteil für die Microsoft-Technologie bedeuten.

Ein weiterer Nachteil ist die zur Zeit noch geringe Verbreitung und Akzeptanz des Silverlight-Plugins, zwar gibt es schon einige Entwickler, die Silverlight einsetzen und damit zur Verbreitung beitragen wollen, jedoch existieren im WWW zur Zeit nicht viele Anwendungen, welche den Einsatz und damit den Download des Plugins voraussetzen.

Ein Nachteil ist auch die Mixtur aus verschiedenen Sprachen und Techniken, denn der Entwickler muss sich mit XAML, JavaScript und HTML auseinander setzen, was das gesamte Silverlight-Konzept nicht wie eine einheitliche Technik erscheinen lässt und von den Entwicklern ein ständiges Umdenken abverlangt.

Vor allem fehlt die konsequente Durchsetzung der Objektorientierung, denn Objekte können nicht innerhalb der Programmlogik eingefügt werden, vielmehr müssen über JavaScript-Funktionen wie createFromXaml Objekte im XAML-Bereich erzeugt und anschließend dem Objekt hinzugefügt werden, was wiederum eine Kenntnis des Aufbaus bzw. des Designs voraussetzt und somit den Vorteil der Trennung zwischen Design und Logik ein wenig geringer erscheinen lässt. Objekte werden durch XAML erzeugt, die Behandlung der Ereignisse wird wiederum durch JavaScript vollzogen, die Erzeugung des Objektes findet damit außerhalb der JavaScript-Programmlogik statt. Es fehlt auf jeden Fall die in Programmiersprachen wie Actionscript vorhandene Möglichkeit der Vererbung, bedingt durch die strenge XAML-Typisierung und die vorgegebenen Elemente-Klassen ist die Schaffung komplexerer, eigener Typen zur Zeit nicht möglich.

Sobald das Projekt komplexer wird empfiehlt sich der Einsatz von Visual Studio in Verbindung mit dem Silverlight-Template oder Expression Blend, welches für die neue Technologie entworfen wurde und eine komfortable Bearbeitung sowohl des Designs über Drag-and-Drop als auch des Codes über die Aufteilungsmöglichkeit des Bildschirms und umschaltbarer Oberfläche für Design- bzw. Programmieraufgaben bietet. Ansonsten wird die Entwicklung sehr mühsam, für größere Projekte ist die alleinige Arbeit mit Notepad einfach zu komplex und aufwendig.

Der zuvor erwähnte Nachteil der Misch-Technologie zeigt sich auch in der Fehlersuche, denn selbst mit dem für den Silverlight-Einsatz empfohlenen Expression Blend von Microsoft gestaltet sich die Entwicklung äußerst schwierig. Zwar analysiert Blend die XAML-Syntax und gibt Hinweise zu Fehlern, jedoch bleibt der JavaScript-Code ungeprüft, lediglich kryptische Fehlermeldungen des JS-Interpreters im Browser können Aufschluss über eventuelle Probleme geben, sind aber meist zu ungenau, berichten teilweise von Fehlern an vollkommen anderen Stellen, als sie tatsächlich auftreten.

Ein weiteres Problem, welches jedoch nicht als technologiebedingt zu sehen ist, sind die teilweise extrem voneinander abweichenden Angaben zur Programmierung der Silverlight-Elemente mittels JavaScript, nicht nur Unterschiede in Groß- und Kleinschreibung sind zu finden, auch komplett andere Attributbezeichnungen wie z.B. beim Einfügen der TimelineMarker, abhängig davon, wann die Silverlight-Anwendung programmiert oder der Artikel darüber geschrieben wurde, fordern vom Entwickler viel Geduld und andauernde Recherche. Dies kommt zustande, da Silverlight in seiner Version 1.0 noch in den Kinderschuhen steckt und selbst die veröffentlichte Version sowohl von Silverlight selbst als auch der Entwicklungswerkzeuge wie Blend des öfteren geändert wurde und somit in Codebeispielen Attribute verwendet werden, welche umbenannt wurden bzw. die gezeigte Funktionalität nicht länger unterstützen.

3.8 Ausblick auf Silverlight 2.0

Zur Zeit bietet Microsoft die Möglichkeit, die neue Betaversion des Silverlight 2.0 SDK herunterzuladen, auch Expression Blend 2.5 steht für die Entwicklergemeinde schon als Vorab-Version bereit. Eine Alpha-Version von Silverlight 2.0 gab es nie, denn im Dezember 2007 entschied sich Microsoft, für die bisher als Silverlight 1.1 geplante Nachfolgeversion aufgrund der überarbeiteten und erweiterten Funktionalität die Versionsnummer 2.0 zu vergeben. In der Tat bietet SL 2.0 einiges mehr an Funktionalität, welche vor allem den Programmierern zugute kommt. Die wichtigste Neuerung ist wohl die Möglichkeit, C#- und VB-Code in die Anwendung zu integrieren, Eventhandler und Funktionen lassen sich nun nicht länger nur in JavaScript definieren, auch die Entwicklung von eigenen Klassen ist nun in einfacher Form möglich. Die Palette der Anzeige-Elemente wurde erweitert, um sich nach und nach der Flash-Konkurrenz anzunähern, auch die Verarbeitung von XML soll bedeutend erleichtert werden. Es wird sich zeigen, in welchem Maß die Neuerungen die Position von Silverlight gegenüber Flash sichern, eventuell verbessern können und wie gravierend die Änderungen und damit die Vorteile gegenüber der Version 1.0 ausfallen werden.

4. Konzeption

4.1 Inhalt und Funktionalität

Die grundlegende Aufgabe des Projektes bestand darin, eine virtuelle Besichtigungsmöglichkeit für die zur HTW Dresden gehörende Bibliothek im Rahmen des eCampus-Projektes bereitzustellen. Die dabei zu vermittelnden Inhalte entsprechen neben der Visualisierung der Bibliothek den Besonderheiten der jeweiligen Etage sowie Nutzungshinweisen und Orientierungshilfen.

Darzustellende Inhalte der Ebene 1 wären demnach:

- Eingang mit Informationen zu Kontaktdaten und Öffnungszeiten
- Garderobe
- Aufzug
- Ausleihe / Information
- Selbstverbuchung
- Rückgabeautomat
- OPAC
- Magazin

Der Eingang der Bibliothek bildet den Ausgangspunkt des Rundganges.

Für die zweite Ebene ergeben sich dann die Inhalte

- Treppenhaus 1 / 2
- Aufzug
- Terrasse
- Datenbankrecherche / Internet
- Kopierraum
- Lesesaal
- WC
- Zeitschriften
- sowie Bücher in Freihandaufstellung

welche entsprechend der Übersichtstafeln erklärt werden sollen.

Auf der dritten Ebene sind die Besonderheiten

- Treppenhaus 1 / 2
- Aufzug
- Gruppenräume / Veranstaltungsraum
- Lesesaal
- WC
- und Bücher in der Freihandaufstellung

darzustellen.

Inhalte der vierten Etage sind:

- Treppenhaus 1 / 2
- Aufzug
- Carells
- WC
- Räume der Bibliotheksmitarbeiter
- und Freihandaufstellung

Die Darstellung soll in bewegten Bildern mit einem möglichst hohen Grad an Realismus erfolgen, daher bietet es sich an, mit aufgezeichnetem Videomaterial zu arbeiten. Der Benutzer soll selbstständig die Steuerung des Videomaterials übernehmen, somit kann die Erfahrung eines realen Rundgangs durch die Bibliothek vermittelt bzw. simuliert werden. Entsprechend dem architektonischen Aufbau der Etagen wird jeweils ein Video verwendet, welches vom Ausgangspunkt aus ohne Unterbrechung und über die rechte Seite beginnend den Hauptweg um das Atrium bis zum erneuten Erreichen des Ausgangspunktes darstellt. Abzweige werden als Videos mit dem darzustellenden Inhalt des Abzweiges einzeln abgedreht. Vor Beginn des Videodrehs wurde die Kamerapositionierung während des Rundgangs an den einzelnen Haltepunkten festgelegt, damit im fertigen Material Informations- und Navigationselemente richtig positioniert werden können und das Blickfeld den zu verdeutlichenden Inhalten entspricht. Die Ansteuerung der Inhalte soll über die Ebenenkarten möglich sein, die Steuerung des Videos erfolgt zudem über grafische Elemente im Video selbst sowie über eine Suchfunktion. Dies ist nötig, damit die Anwendung für mehrere Aufgaben, entsprechend der nicht klar abgrenzbaren Zielgruppe, gleichzeitig genutzt werden kann:

- Schnelles Suchen und Finden von Inhalten
- Vorabinformation
- Unterhaltungswert / Vorstellung der Bibliothek

Somit gestaltet sich der Rundgang interessant, sowohl für Interessenten am Gebäude selbst als auch für Studenten, die herausfinden wollen, wo sich eine bestimmte Sache befindet, z.B. der Kopierraum, und welche Voraussetzungen für die Nutzung gegeben sein müssen.

Das Team

Um ein möglichst schnelles Vorankommen bei der Entwicklung zu gewährleisten und qualitativ hochwertige Ergebnisse zu erzielen wurde eine Arbeitsgruppe im Rahmen des eCampus-Projektes mit den Aufgabenbereichen

- Medienproduktion
- Design
- und Programmierung / Inhaltsumsetzung

gebildet, wobei jeweils eine Person des Teams eine Aufgabe bearbeitete.

Zuständig für den Bereich Medienproduktion und damit der Umsetzung der Videokonzeption in verwendungsfähiges Material war Herr Maik Vlcek, Student der HTW-Dresden und studentische Hilfskraft des eCampus. Neben der Aufnahme des Materials bestand seine Aufgabe darin, Störungen wie Verwackelungen, verursacht durch Unebenheiten des Bodens, zu beheben, dafür fand Adobe After Effects Verwendung, da es die dafür notwendigen Funktionen bereithält. Außerdem wurden Abweichungen der Farben korrigiert und die Helligkeiten der einzelnen Videos angeglichen, da gleich bleibende Aufnahmebedingungen durch Tageslichtunterschiede nicht gegeben waren.

Für die Umsetzung des Designs konnte Frau Dipl.-Inf. (FH) Jana Halgasch gewonnen werden, Mitarbeiterin der HTW Dresden und Mitarbeiterin des eCampus mit dem Aufgabengebiet Contententwicklung und technische Beratung.

Ihre Aufgabe war die Umsetzung der funktionalen Anforderungen in ein Design für das Benutzerinterface sowie die Gestaltung der Interaktionselemente. Dabei fanden die Farbvorgaben der Bibliothekswebseite Verwendung sowie klare Formen, welche die einzelnen Bereiche abgrenzen und auf unnötige Ablenkung durch irrelevante Elemente verzichten. Zudem war Frau Halgasch neben Frau Prof. Dr. Teresa Merino für die Koordination sowie Projektleitung zuständig.

Die Programmierung und den Zusammenbau der Anwendung aus Designvorlage, Videomaterial und Textvorgaben sowie die Kodierung und Anpassung des Videomaterials an die Anwendungsvorgaben bildeten den Aufgabenbereich, für den ich verantwortlich war. Unter Verwendung von durch Frau Dipl.-Bibl.(FH) Petra-Sybille Stenzel, Leiterin der Bibliothek, bereitgestelltem Material sowohl in Textform als auch in Form von Ebenenkarten wurden die Inhalte gestaltet und in eine für die Applikation verwendbare Form gebracht.

4.2 Zielgruppe

Da die "Virtuelle Bibliothek" nicht nur HTW-intern abrufbar sein soll, besteht für jede Person mit Zugang zum Internet und installiertem Browser-Plugin die Möglichkeit, am virtuellen Rundgang teilzunehmen, daher kann der Kreis der Nutzer nur schwer eingeschränkt werden. Jedoch entstand das Projekt aus der Idee heraus, Studenten der HTW Dresden die Nutzung und den Aufbau der Bibliothek sowie Besonderheiten dieser näher zu bringen, damit stellen Personen im studierfähigen Alter ab 18 Jahren die Hauptzielgruppe dar. Die Nutzung des Internets sowie der Umgang mit Anwendungen sollte dieser Gruppe bekannt sein, dennoch sollte der Aufbau der Anwendung sowie Navigations- und Infosymbole im Sinne der intuitiven Bedienbarkeit möglichst einfach gestaltet und klar erkennbar bzw. den Funktionen zuzuordnen sein.

Damit wird eine längere Einarbeitungszeit des Nutzers vermieden und eventuelle Nachteile, resultierend aus eventuell fehlender Erfahrung im Umgang mit diesen Medien, kommen nicht zum Tragen. Somit werden auch alle anderen Nutzer, welche nicht zur Hauptzielgruppe gehören, berücksichtigt, welche sich aus Angehörigen der Studenten und sonstigen Interessenten zusammensetzen.

Vor allem Personen mit wenig Interneterfahrung kommt eine unkomplizierte, intuitive Bedienung zugute, welche auf unnütze Funktionalität verzichtet und sich auf das Wesentliche beschränkt, somit ohne Vorkenntnisse den Einstieg und auch die Nutzung der Anwendung für diesen Personenkreis gewährleistet.

4.3 Design und Navigation

Das Design der Anwendung wurde unter Berücksichtigung der funktionellen Anforderungen von Frau Dipl.-Inf. (FH) Jana Halgasch erstellt. Zielstellung war es dabei, optisch ansprechend und dem Design der Bibliotheks-Internetseite folgend Hintergründe und Platzhalter für die Elemente

- Video
- Informationstext
- Etagen-Navigation
- Haltepunkt-Navigation
- sowie Suchfunktion

zu schaffen.

Dabei teilen sich Etagen- und Haltepunkt-Navigation sowie Suchfunktion den Platz für ihre Darstellung, da im Anwendungsverlauf immer nur eine Komponente sichtbar sein muss, die Navigationskarten liegen minimiert unterhalb des Darstellungsbereiches, beim Überfahren mit der Maus vergrößern sie sich selbstständig, beim Verlassen werden diese wieder minimiert und geben die Sicht auf die Suchfunktion frei.

Abbildung 4: Interface mit minimierter Navigationskarte

Abbildung 5: Interface mit maximierter Navigationskarte

Für die direkte Navigation im Videomaterial wurden Pfeil-Symbole entworfen, welche die Richtung der Bewegung wiedergeben, außerdem zeigen Info-Symbole das Vorhandensein weiterführender Informationen an.

Abbildung 6: Symbole Direktnavigation

Die Navigationssymbole der Ebenenkarten bzw. Stellvertreter der Haltepunkte folgen dem Design des Info-Symbols, somit wird dem Nutzer angezeigt, dass an diesen Stellen Informationen vorhanden sind.

Abbildung 7:
Stellvertreter Haltepunkte

Die Steuerung der Anwendung erfolgt dreistufig, zum einen über im Videomaterial eingeblendete Interaktionselemente, zum anderen über Übersichtskarten der jeweiligen Etage sowie als dritte Möglichkeit über die Suchfunktion.

Die direkte Steuerung über die Interaktionselemente soll an bestimmten Haltepunkten erfolgen, welche Punkte besonderen Interesses darstellen und an denen auf funktionale und architektonische Besonderheiten sowie zusätzliche Informationen zur Einrichtung und Nutzung der Bibliothek hingewiesen werden soll.

Außerdem sollen diese Informationen über Hyperlinks mit Seiten des Inter- bzw. Intranets verknüpft werden können, um weiterführende Informationen für den Nutzer bereitzustellen.

Die Interaktionselemente dienen dabei sowohl der Informationsbereitstellung als auch der Navigation bzw. der Steuerung und werden an den Haltepunkten über das Videomaterial gelegt, durch die Überlagerung können u.a. Informationselemente direkt über das betreffende Objekt gelegt werden, wodurch die Zugehörigkeit der Information zum Objekt hervorgehoben wird.

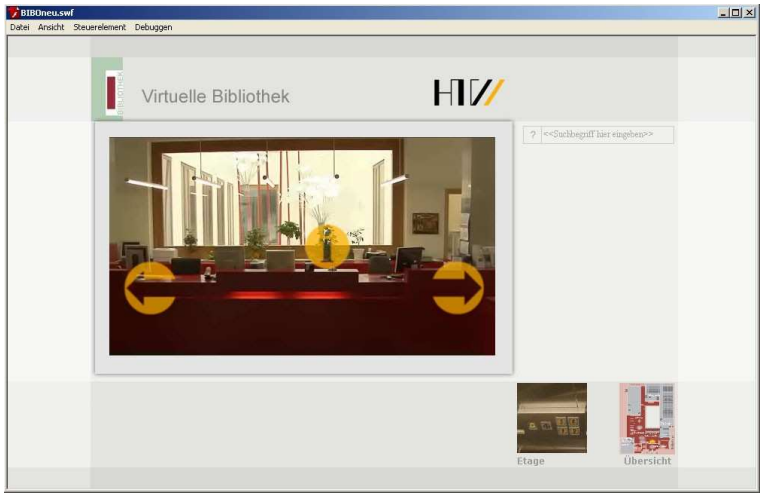

Abbildung 8: Überlagerung durch Info-Symbol

Nach erfolgter Navigation werden die eingeblendeten Elemente wieder ausgeblendet um die Sicht auf den Rundgang nicht unnötig zu verdecken.

Die Steuerung über die Übersichtskarten der jeweiligen Etage soll dem Nutzer eine weitere, der Steuerung über die Bildinhalte untergeordnete Möglichkeit bieten, sich in der virtuellen Bibliothek zu bewegen.

Dabei werden auf der Übersichtskarte der jeweiligen Etage Informationssymbole für jeden existierenden Haltepunkt eingefügt, der jeweils aktuelle Haltepunkt wird dabei farblich hervorgehoben. Durch den Mausklick auf einen Haltepunkt besteht für den Nutzer die Möglichkeit, sich auf der Etage zu der ausgewählten Position zu bewegen. Danach wird das dem Haltepunkt zugeordnete Video geladen und innerhalb dessen auf die jeweilige Haltepunktposition gespult. Somit muss der Nutzer nicht dem Rundgang in seinem vorgegebenen Ablauf folgen sondern kann Videopassagen überspringen und sofort an Haltepunkte gelangen, welche für ihn von besonderem Interesse sind.

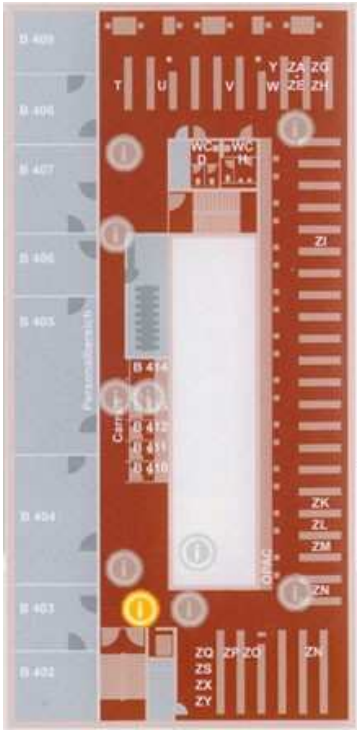

Abbildung 9: Ebenenkarte mit Haltepunktsymbolen

Die Umschaltung der Etagen erfolgt über eine Darstellung der Aufzugtastatur, durch Klick auf die Tastaturknöpfe wird die jeweilige Etage ausgewählt und durch eine farbliche Hervorhebung der Bedienflächen wird die aktuelle Etage angezeigt.

Abbildung 10: Etagenumschaltung

Um dem Nutzer eine Hilfestellung bei der Orientierung sowie dem Auffinden bestimmten Inhaltes zu geben soll die Anwendung über eine Suchfunktion verfügen, welche die dritte Stufe der Navigation bildet. Diese Funktion durchsucht alle Haltepunkte sowie Interaktionselemente auf das Vorkommen der eingegebenen Zeichenkette und liefert nach erfolgreicher Suche Textauszüge der betreffenden Komponente. Durch Klicken auf den Textauszug besteht die Möglichkeit, direkt an die Stelle zu springen, welche mit dem Textauszug verknüpft wurde, dadurch können vom Nutzer gesuchte, nicht aus der Navigationskarte ersichtliche Informationen schneller gefunden und eingeblendet werden.

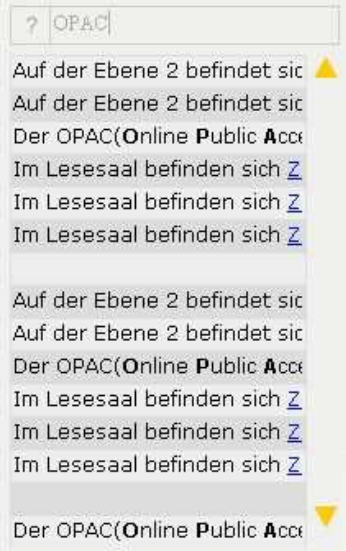

Abbildung 11: Suchergebnis

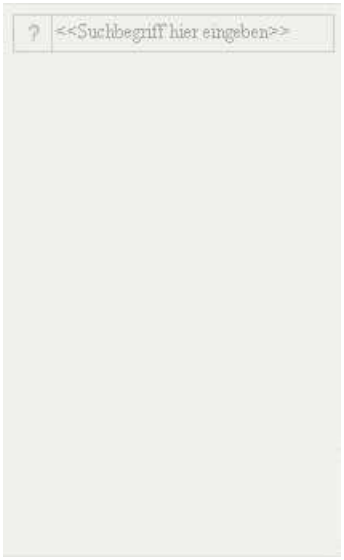

Abbildung 12: Suchfunktion

4.4 Technische Konzeption

Die Steuerung des Videomaterials, die Koordination der Haltepunkte und die Implementierung der Anwendungslogik erfolgt über die durch die jeweilige Programmiersprache bereitgestellten Funktionen, wobei objektorientierte Programmierung in Form von Klassen zum Einsatz kommt. Auf den Klassenaufbau wird später in den zur Programmiersprache zugehörigen Kapiteln 5 und 6 näher eingegangen.

Für die Beschreibung des Inhaltes sowie die Platzierung der Interaktionselemente, das Setzen der Haltepunkte, die Verwaltung der Etagen und die Verknüpfung der Videos untereinander bietet sich die Verwendung von XML an, da XML als universelles Sprachkonstrukt sehr flexibel und mächtig ist.

Elemente und Attribute können weitestgehend beliebig benannt und angeordnet werden, Schachtelungen von Elementen bzw. die Zuordnung von Unterelementen ist ebenfalls ohne Probleme möglich. Weiterhin bieten aktuelle Programmiersprachen wie Actionscript 3.0 oder Silverlight 1.0 mehr oder weniger umfangreiche Möglichkeiten zur Verarbeitung von XML. Ein weiterer Vorteil ist die Möglichkeit Änderungen direkt mit jedem beliebigen Editor vornehmen zu können, da XML im ASCII-Textformat vorliegt. Auch die Erweiterbarkeit der Anwendung ist über XML gegeben, neue Elemente, Videos und Haltepunkte können jederzeit über die XML-Definition eingebunden werden. Der Aufbau der XML-Datei wird später in den Kapiteln 5 und 6 zur prototypischen Implementierung erläutert.

Eine weitere Anforderung an die Anwendung ist die einfache Pflegbarkeit die ohne tiefgreifende Kenntnisse der Programmiersprache sowie des internen Aufbaus der Anwendung möglich ist. Dabei sollen die HTML-Inhalte möglichst nach dem WYSIWYG-Prinzip bearbeitet, aktualisiert und angepasst werden können um einerseits den zeitlichen Aufwand für die Pflege gering zu halten und andererseits für den Bearbeiter eine geringe Einarbeitungszeit zu gewährleisten. Um dies zu realisieren werden die HTML-Texte des Inhaltes von der XML-Definition getrennt in einer Ordnerstruktur gehalten, welche ausgehend vom XML-Aufbau gegliedert ist und deren Inhalte in der XML-Definition mit den jeweiligen Elementen verknüpft wird.

Für die Einbindung neuer Haltepunkte, Videos etc. sind dennoch die XML-Kenntnisse nötig, da die Platzierung der Elemente im Video, die zeitliche Einordnung der Haltepunkte sowie die Verknüpfung der Videos innerhalb der XML-Definition stattfindet, deshalb nur umständlich zu visualisieren wäre.

4.5 Technische Spezifikation

Die Einbindung der Anwendung in die Internetpräsenz der Bibliothek soll über einen Link erfolgen, welcher den "Virtuellen Rundgang" in einem neuen Fenster startet. Somit kann die Größe der Anwendung ohne Rücksichtnahme auf die Platzverhältnisse der Webseite gewählt werden, bei einer derzeit standardmäßigen Auflösung von 1024x768 Bildpunkten bietet sich eine fast bildschirmfüllende Anwendungsauflösung von 1000x600 Pixeln an. Das Videomaterial wird auf 528x288 Bildpunkte verkleinert um die Datenrate und damit die erforderliche Bandbreite der Internetverbindung zu reduzieren aber dennoch ein optisch und qualitativ gutes Ergebnis für den Nutzer zu schaffen. Aufgrund des heute vorherrschenden DSL-Standards wird eine Datenrate von 400Kbit/s angesetzt, welche damit etwas weniger als der Hälfte der DSL1000-Geschwindigkeit von 1000 KBit/s entspricht und somit auch bei Verlustraten von 50 Prozent der Pakete ausreichend sein müsste. Für veraltete Verbindungen wie 56k-Modem und ISDN wäre eine Realisierung aufgrund der geringen Datenrate von 56 bzw. 64 Kbit/s nur in unbefriedigender Qualität möglich und ist deshalb nicht vorgesehen. Aufgrund der Fülle des Videomaterials und der damit verbundenen großen Menge an Daten bietet sich die Möglichkeit an, diese per Streaming für die Anwendung bereit zu stellen, somit die Ladezeiten zu verringern und nur benötigtes Datenmaterial vom Server abzurufen. Die Bereitstellung der Dateien erfolgt direkt auf den Servern der Bibliothek, um den direkten Zugriff auf die Daten gewährleisten und die Anwendung zwecks Pflegbarkeit geschlossen halten zu können.

Die technischen Daten des Webservers, auf dem die Bibliotheksseiten laufen lauten wie folgt:

- DELL-Server
- 2x Intel Xenon Prozessor je 2,8 GHz
- 2 GB Hauptspeicher
- Betriebssystem: Suse Linux 9.3

Da die Installation neuer Software bzw. Abänderung der Konfiguration nicht möglich bzw. erwünscht ist, muss mit der vorhandenen Konfiguration vorlieb genommen werden und entsprechend dieser entwickelt werden. Für die Umsetzung mit Silverlight wäre ein Microsoft Mediaserver notwendig, welcher zur Zeit nur für die aktuellen Windows Plattformen vorhanden ist.

5. Implementation mit Flash CS3 / Actionscript 3.0

In diesem Kapitel wird die konkrete Implementation der Anwendung "Virtuelle Bibliothek" mittels Actionscript 3.0 und Flash CS3 behandelt. Dabei wird auf den grundlegenden Aufbau sowie verschiedene Aspekte und Besonderheiten der Programmierung, welche während der Arbeit am Prototypen aufgetreten sind und näherer Erklärung bedürfen, eingegangen.

5.1 Aufbau der Klassen

Da Actionscript 3.0 objektorientiert aufgebaut ist, bietet sich eine Unterteilung bzw. Kapselung der logischen Einheiten mit ihren Funktionen und Attributen in Form von Klassen an, welche in der fertiggestellten Anwendung miteinander interagieren sollen. Als grobe Unterteilung bietet sich in diesem Fall der Aufbau des Bibliotheksgebäudes an, denn dieser dient zum einen als abstraktes Denkmodell für die Navigation und sorgt zum anderen für ein grundlegendes Verständnis des Klassenaufbaus aufgrund der physischen Gegebenheiten des Gebäudes. Es werden somit Klassen benötigt welche die Elemente

- Gebäude
- Etage
- sowie Haltepunkt

kapseln und über entsprechende Funktionen miteinander verbinden, um das physische Gefüge der Bibliothek darzustellen und in logische Einheiten aufzuteilen.

Zusätzlich sind aufgrund des Aufbaus als Video-Applikation Klassen nötig, welche

- Videodaten
- und Video-Haltepunkte

symbolisieren und die jeweilige Funktionalität bereitstellen.

Des Weiteren werden für die Gewährleistung der Ansprüche hinsichtlich Bedienbarkeit und Hilfestellung Klassen für

- Navigations- bzw. Übersichtskarten
- Suchfunktion
- und "intelligente" Textfelder zur Darstellung der Suchergebnisse

benötigt, welche bei Klick-Ereignissen selbstständig für das Anwählen der gewünschten Sprungmarke verantwortlich sind.

Die TUTOR-Klasse

Die Einheit "Gebäude" wird durch die abstrakte Klasse "TUTOR" beschrieben, welche als oberste Ebene die Anwendung und damit den Lernfilm, englisch Tutorial, umschließt, somit gleichfalls die Schnittstelle zwischen Flash-Film mit den Design-Elementen und der Actionscript-Funktionalität der einzelnen Klassen darstellt. Die Aufgaben der TUTOR-Klasse sind die eben erwähnte Herstellung der Schnittstelle zwischen Design und Funktionalität, die Verwaltung der Etagen sowie die Herstellung der Verbindung zur XML-Datei, welche die Beschreibung des Rundgangs enthält. Die Verbindung zwischen Flash-Design und der TUTOR-Klasse wird über den Konstruktor hergestellt, das heißt, ein Objekt vom Typ TUTOR wird im Aktionen-Bereich des gewünschten Bildes angelegt und die benötigten Referenzen übergeben.

Benötigte Referenzen um den Kontakt zwischen grafischem Aufbau und Anwendungslogik herzustellen sind:

- einzulesende XML-Datei (Pfad der gewünschten Szenenbeschreibung als String)
- eine Referenz auf die Bühne (die Bühne muss als Typ "main" deklariert werden)
- eine Referenz des MovieClips, in welchen der Videoinhalt gezeichnet werden soll (dient vor allem der Positionierung des Videos auf der Oberfläche)
- und eine Referenz auf das TextField, welches später die Informationen enthalten soll

Im konkreten Fall wäre es damit der Aufruf

```
var myTutor:TUTOR=new TUTOR( "Test2.xml", this, getChildByName("videoClip") as
MovieClip, getChildByName("infodisplay") as TextField);
```

welcher nach dem Importieren der benötigten Klassen mittels

```
import classes.*;
```

in das erste und einzige Bild der FLA-Datei im Aktionen-Bereich platziert wird. Der Rundgang wird damit an der Position des MovieClip "videoClip" angezeigt bzw. an dessen Größe angepasst, die einzublendenden Informationen werden im TextField "infodisplay", welches sich ebenfalls auf der Bühne befindet, dem Nutzer präsentiert. Dies erleichtert die Positionierung der Elemente auf der Bühne und sorgt für die sinnvolle Trennung zwischen Design und Logik, Änderungen im Design müssen damit nicht über Parameteränderung oder Bearbeiten der Klassen mit dem dazugehörigem Mehraufwand dem Actionscript-Teil der Applikation mitgeteilt werden.

Nach dem einmaligen Entwurf des grafischen Aufbaus und dem Verbinden mit der TUTOR-Klasse kann das Design beliebig geändert werden, so lang es den Anforderungen der TUTOR-Klasse entspricht, das Vorhandensein des TextFields und des Ziel-MovieClips ist sicherzustellen.

Im Konstruktor der Klasse werden diese Referenzen den klasseneigenen Variablen zugewiesen, um den Zugriff auf die Objekte so unkompliziert wie möglich zu gestalten. Über einen Loader wird die XML-Szenenbeschreibung geladen und nach Abschluss des Vorgangs werden die für die Klasse relevanten Informationen ausgelesen und den Variablen zugewiesen. Die zu den Etagen gehörenden XML-Fragmente werden anschließend den Etagen-Objekten übergeben, welche anschließend dem Etagen-Array der TUTOR-Klasse hinzugefügt werden. Danach wird das Video in den auf der Bühne befindlichen MovieClip eingefügt und der Abspielvorgang gestartet, gefolgt vom Hinzufügen der Suchfunktion sowie der Navigation. Die Klasse TUTOR enthält einige öffentliche Funktionen zum

- Setzen eines Videos als aktives Video (setActiveVideo(...))
- Finden und Setzen eines Videos über die Bezeichnung (findActiveVideo(...))
- Wiederaufnehmen des Abspielvorgangs (resumeVideo())
- Anhalten des Abspielvorgangs (stopVideo())
- Setzen der Abspielposition (seekPosTime(...))
- Abrufen des Pfades des aktuellen Videos (getActiveVideo(...))
- Setzen der aktuellen Etage (setActiveFloor(...))
- sowie Starten einer rekursiven Suche nach einem Begriff (findString(...))

und weiteren, zum Teil privaten Funktionen, welche lediglich für die Funktionalität der Klasse sowie Ereignisbehandlung von Bedeutung sind.

Obwohl die TUTOR-Klasse eigentlich nicht für die Behandlung und Steuerung der Videodaten verantwortlich ist, mussten trotzdem Funktionen geschaffen werden, welche Nicht-Kind-Objekten, wie z.B. der Navigation eine Steuerungsmöglichkeit bieten, da diese nicht direkt auf Etagen-, Video- und Haltepunktfunktionen zugreifen können. Die TUTOR-Klasse implementiert dabei keine Steuerungsfunktionalität im eigentlichen Sinne sondern dient vielmehr als Interface und reicht die übergebenen Werte an die jeweils zuständigen Funktionen weiter. Die Methoden zum Setzen und Auslesen des aktiven Videos mussten in der TUTOR-Klasse implementiert werden, da die Abspielkomponente ebenfalls in dieser erstellt sowie gehalten wird und eine Auslagerung auf Etagen-Ebene nur einen Mehraufwand sowie eine der Anzahl der Etagen entsprechende Menge an Abspielkomponenten und damit mehr Speicherbedarf verursacht hätte. Das der TUTOR-Klasse untergeordnete und damit von ihr gesteuerte Element ist die für die Etagenbeschreibung zuständige FLOOR-Klasse.

Die FLOOR Klasse

Die Beschreibung der jeweiligen Etage wird über ein Objekt vom Typ FLOOR realisiert, welches in der Eltern-Klasse TUTOR erzeugt und mittels der Funktion

```
floorList[i].initFloor(new XMLList(xmlList[i]),this);
```

initialisiert wird. Dabei wird der jeweilige FLOOR-Knoten des XML-Datensatzes übergeben und vom FLOOR-Objekt in dessen Funktion initFloor(...) ausgewertet und den eigenen Variablen zugewiesen. Die Etage besitzt einen Namen, um neben der zahlenbasierenden Indizierung über den FLOOR-Array der TUTOR-Klasse eine weitere Möglichkeit der Identifikation bereitzustellen, außerdem wird die Navigationskarte der jeweiligen Etage geladen sowie die Kind-Objekte, die eigentlichen Videodaten, erzeugt, initialisiert und in den dafür vorgesehenen Array eingefügt.

Da diese Klasse für die Steuerung der Videodaten verantwortlich ist, bietet sie öffentliche Funktionen zum

- Auffinden eines Videos anhand von dessen Bezeichner (findActiveVideo(...))
- Setzen des aktuellen Videos (setActiveVideo(...))
- Auslesen des aktuellen Videos (getActiveVideo())
- Fortführen der in der TUTOR-Klassen begonnenen rekursiven Suche (findString(...))
- und Auslesen der zur Etage gehörenden Navigationskarte (getNavMap())

welche in der TUTOR-Klasse als eigentliche Hauptanwendung zugewiesen wird. Die FLOOR-Klasse erweitert die Klasse MovieClip um neben der Verwaltungsfunktionalität hinsichtlich der Video-Elemente auch eine Auswahlmöglichkeit bzw. Etagenumschaltung zu ermöglichen. Das ELOOR-Objekt kann somit einer Ebenen-Karte zugewiesen werden und führt dann beim Klick-Ereignis auf die die Ebene symbolisierende Grafik selbstständig die Anwahl der jeweiligen Ebene im elterlichen TUTOR-Objekt durch. Somit wird kein zusätzliches Objekt benötigt, welches die Konsistenz der Anwendung beeinträchtigen könnte.

Die VIDEO Klasse

Da der virtuelle Bibliotheksrundgang hauptsächlich aus Videomaterial besteht ist die VIDEO-Klasse von großer Bedeutung. Sie erweitert die FLVPlayback-Komponente von Actionscript 3.0 und steuert die Videodaten, sowohl deren Download als Datenstrom via Streaming als auch den Abspielvorgang und übernimmt das Haltepunkt-Management.

Nachdem das VIDEO-Objekt in der Eltern-Klasse FLOOR angelegt wurde bekommt es von dieser den das Objekt betreffende XML-Knoten mittels

```
videoList[i].initVideo(new XMLList(xmlSource[i]),callingTut,this);
```

übergeben und liest anschließend die Daten aus. Dabei entspricht der Videoname dem Dateiname, welcher mittels Setzen des source-Parameters des VIDEO-Objektes geladen wird. Somit ist die VIDEO-Komponente abspielbereit, nun müssen lediglich die Haltepunkte in den dafür vorgesehenen Array sowie in die Liste der Actionscript-Haltepunkte des Videos eingefügt werden. Letzteres gestaltet sich etwas schwieriger, da die FLVPlayback-Komponente, von der die VIDEO-Klasse abgeleitet ist, nur Objekte vom abstrakten Typ Object als Haltepunkte akzeptiert. Ein einfaches Hinzufügen von Objekten der CUEPOINT-Klasse führt zu Fehlermeldungen und Funktions-Verweigerung seitens der FLVPlayback-Komponente, selbst wenn die CUEPOINT-Klasse den Typ Object erweitert, somit müssen neue Objekte für den videointernen Gebrauch geschaffen werden, welche über eine Indexnummer mit den eigenen Haltepunkt-Objekten in Verbindung gebracht werden können.
Das Anlegen der Haltepunkte erfolgt innerhalb der insertASCuepoints()-Funktion der VIDEO-Klasse über die Befehle

```
for(i=0;i<cuePointList.length;i++)
{
	var theCuePoint:Object=new Object();
	cuepointTime=cuePointList[i].getCuepointTime();
	theCuePoint.name = "ASpt"+cuepointTime;
	theCuePoint.type = "actionscript";
	theCuePoint.cpt= cuePointList[i].nr;
	theCuePoint.time = cuepointTime;
	addASCuePoint(theCuePoint);
}
```

wobei die Objekteigenschaften type auf "actionscript" gesetzt werden sollte und time den Zeitpunkt des Auslösens des Haltepunkt-Ereignisses angibt und damit unerlässlich ist.

Der selbst gewählte Parameter cpt wird hier verwendet, um dem videointernen Haltepunkt die Indexnummer des intelligenten Haltepunktes der eigenen Haltepunkt-Klasse zuzuordnen, ansonsten ist es beim Eintreten des Haltemarken-Ereignisses beinahe unmöglich, einen schnellen Zugriff auf die Daten des eigentlichen Haltepunkt-Objektes zu bekommen bzw. wäre eine Durchsuchung des Arrays nach Attributen wie z.B. der Zeit nötig, was wiederum aus Performance-Gründen unsinnig ist. Ein Anhängen einer Referenz auf das Haltepunkt-Objekt an das videointerne Pendant ist zwar möglich, jedoch geht die Referenz auf das Objekt bei der Rückgabe verloren, da jegliche Information verworfen wird, welche nicht zum abstrakten Typ Object zugeordnet werden kann. Neben den durch die FLVPlayback-Komponente bereitgestellten Funktionen zur Steuerung des Videos wurden folgende Funktionen geschaffen:

- die Zeit des aktuellen Haltepunktes zurückzugeben (getCuePointTimeOnly())
- die auszuführende Aktion am Haltepunkt auszulesen (getCuepointAction(...))
- das Ziel der auszuführenden Aktion abzufragen (getActionTarget(...))
- zum Haltepunkt gehörende Informationen auszulesen (getCuepointInfoText(...))
- beim Wechsel des Videos die aktuelle Position zu lesen bzw. zu schreiben (setVideoPosition(...)) und (getVideoPosition())
- die stellvertretenden Symbole für die Haltepunkte der Etage auf der Übersichtskarte passiv bzw. aktiv zu setzen (setAllCuepointsPassive() / setCurrentCuepointPassive() / setCurrentCuepointActive())
- einen Haltepunkte als aktuellen Haltepunkt zu setzen (setCurrentCuepoint(...) / setCurrentCuepointNR(...))
- Steuerungssymbole einzublenden (insertCuepointItems())
- eingeblendete Symbole zu entfernen (removeMCs())
- und die rekursive Suche in den untergeordneten Elementen fortzusetzen (findString(...))

Einige Funktionen wären unnötig, wenn es möglich wäre, die eigenen Haltepunkte auch als videointerne Haltemarken zu verwenden. Durch dieses Manko ist neben der Videosteuerung auch die Verwaltung der Haltepunkte eine Aufgabe, welche die VIDEO-Klasse übernehmen muss.

Die CUEPOINT Klasse

Obwohl die FLVPlayback-Klasse Actionscript-Haltepunkte über die abstrakte Klasse Object realisiert, ist eine Verwendung einer eigenen CUEPOINT-Klasse unumgänglich, denn die Funktionalität des Typs Object reicht nicht aus, um das gewünschte Verhalten der Anwendung herzustellen.

Die Haltepunkte des jeweiligen Videos werden in einem separaten Array im jeweiligen VIDEO-Object gehalten, der CUEPOINT wird über

```
cuePointList[i].initCuepoint(new XMLList(xmlSource[i]),callingTutor,this);
cuePointList[i].nr=i;
```

initialisiert und zur eindeutigen Zuordnung anschließend die Indexnummer im Haltepunkt-Array zugewiesen. In der Initialisierungsfunktion wird der übergebene XML-Knoten ausgewertet und Informationen über Haltepunkt-Typ, -Zeit, -Ziel, -Text, und -Position des Platzhalters auf der Navigationskarte ausgelesen.

Die Klasse CUEPOINT erweitert die Actionscript-Klasse MovieClip um neben der eigentlichen Aufgabe der Klasse, der Verwaltung der am Haltepunkt einzublendenden grafischen Objekte, auch die grafische Einbindung des Haltepunktes auf der Navigationskarte so einfach wie möglich zu gestalten. Der Haltepunkt steuert beim Klick-Ereignis auf den repräsentierenden MovieClip auf der Navigationskarte selbst die Anwahl des richtigen Haltepunkt- bzw. VIDEO-Objektes.

Die CUEPOINT-Klasse bietet öffentliche Funktionen zum:

- Setzen des Haltepunktes als aktiv und Änderung des dem Haltepunkt zugeordneten anzuzeigenden MovieClips (setActive())
- Setzen des Haltepunktes als passiv / Änderung des MCs (setPassive())
- Rückgabe der Haltepunkt-Zeit (getCuepointTime())
- Rückgabe des Verhaltens (getCuepointAction())
- Rückgabe des Aktions-Ziels (getCuepointTarget())
- Rückgabe der Position auf der Navigationskarte (getCuepointPosX() / getCuepointPosY())
- Erzeugung eines MovieClips, welcher die dem Haltepunkt zugeordneten Szenen-Elemente enthält (insertItems())
- Fortsetzen der rekursiven Suche (findString(...))

Zudem sind interne Funktionen zur Ereignisbehandlung bei Mausklick auf den MovieClip des Haltepunktes auf der Navigationskarte oder ein Szenenobjekt vorhanden. Die Ereignisbehandlung beim Klick wird nicht direkt im Szenenobjekt vorgenommen, da die Verarbeitung sowieso zum größten Teil in der CUEPOINT-Klasse stattfinden muss, um auf alle Szenenobjekte zugreifen, deren Sichtbarkeit ändern und das CUEPOINT-Verhalten steuern zu können. Eine Auslagerung in das Szenenobjekt hätte lediglich einen zusätzlichen Aufruf der zum CUEPOINT gehörenden Methode zur Folge. Die CUEPOINT-Klasse stellt damit den Endpunkt der Anwendungslogik und des physischen Aufbaus dar, die untergeordnete Klasse dient lediglich visueller Aspekte und deren Steuerungslogik betrifft lediglich die Klasse selbst, sie hat keinerlei Einfluss auf den Programmablauf sondern dient als Grafik- und Informationscontainer.

Die ITEM-Klasse

Genau wie die Klassen CUEPOINT und FLOOR erweitert ITEM die von Actionscript bereitgestellte Klasse MovieClip. Die Hauptaufgabe von ITEM besteht darin, die im Video einzublendenden Steuersymbole, auch Icons genannt, zu visualisieren und ihr Verhalten zu steuern. Die ITEM-Objekte des jeweiligen Haltepunktes werden in einem Array innerhalb des zugehörigen CUEPOINT-Objektes gespeichert. Nach dem Erzeugen des ITEM-Objektes wird es über die Initialisierungsfunktion

```
itemList[i].initItem(new XMLList(xmlSource[i]));
```

mit den Werten des übergebenen XML-Knotens angepasst. Dabei nimmt das ITEM Objekt das vorbestimmte visuelle Erscheinungsbild an, und platziert sich selbstständig an der über die XML-Syntax definierte Position.

Neben der Initialisierungsroutine besitzt die Klasse noch weitere öffentliche Methoden zum

- Einblenden von Informationen zum ITEM (showItemInfo(...))
- Auslesen der auszuführenden Aktion (getItemAction())
- Auslesen des Aktionsziels (getItemTarget())
- Auslesen der Position (getItemPosX() / getItemPosY())
- sowie Fortsetzen der rekursiven Suche im ITEM (findString(...))

sowie private Funktionen zur Ereignisbehandlung z.B. des MouseOver-Ereignisses. Die ITEM-Klasse steht somit ganz unten in der Klassenhierarchie und nimmt keinen Einfluss auf die Steuerung der eigentlichen Anwendung, ist jedoch als grafisches Element zur Applikationssteuerung unverzichtbar.

Die NAVMAP-Klasse

Auch diese Klasse erweitert MovieClip, um so einfach wie möglich eine grafische Repräsentation des Objektes verwirklichen zu können. Die NAVMAP-Klasse dient der Darstellung einer Übersichtskarte, mit deren Hilfe der Nutzer direkt zu bestimmten, auf der Karte über Symbole platzierten Haltepunkten springen kann, ohne dem gesamten Rundgang folgen zu müssen. Das NAVMAP-Objekt ist ein Bestandteil des FLOOR-Objektes der jeweiligen Etage und wird über den Aufruf

```
navMap=new NAVMAP(xmlSource.attribute("navmap"));
```

erstellt und mit dem aus dem XML-Element ausgelesenen Bild-Pfad initialisiert. Anschließend werden alle Haltepunkte der Etage durch die Codezeile

```
navMap.addCuepoints(videoList[i].getCuepointList() as Array);
```

als optische Platzhalter und Sprungmarken zum Erscheinungsbild der Übersichtskarte hinzugefügt. Durch Klick auf diese Symbole gelangt der Benutzer zum gewünschten Punkt des Rundganges. Durch die zweifache Verwendung des CUEPOINT-Objektes, einerseits als Datenobjekt bzw. Haltepunktsymbolverwaltung und andererseits als grafische Repräsentation muss keine neue Verbindung zwischen Kartenfunktionalität und bereits erstellter Programmlogik geknüpft werden, die Hierarchie der Objekte bleibt konsistent und Attribute und Funktionen müssen nicht neu definiert werden.

Da die Navigationskarte außer der geografischen Orientierung des Nutzers keinerlei eigene Funktion hat, denn diese wird über die CUEPOINT-Objekte bzw. anknüpfende Klassen gesteuert, besitzt sie außer den Methoden zum

- Hinzufügen der CUEPOINT-Objekte (addCuepoints(...))
- und Positionieren der Karte (setMapPosX(...) / setMapPosY(...))

keine weiteren öffentlichen Methoden außer denen, die die MovieClip-Klasse beinhaltet.

Die SEARCHFRAME-Klasse

Um dem Benutzer eine Möglichkeit zu bieten gezielt nach Informationen zu Rundgangshaltepunkten und -objekten zu suchen und diese entsprechend aufbereitet darzustellen wurde eine eigene Klasse für diese Aufgabe entwickelt. Eingebunden wird die SEARCHFRAME-Klasse direkt im TUTOR-Objekt über den Aufruf:

```
var SF:SEARCHFRAME = new SEARCHFRAME(this, paintDC.getChildByName(
"searchWhat") as MovieClip);
```

Anschließend wird das erzeugte Objekt der Bühne hinzugefügt. Der dem SEARCHFRAME-Konstruktor übergebene MovieClip dient, wie bereits auch bei der TUTOR-Klasse, der Positionierung des Suchfeldes auf der Flash-Bühne.

Im Konstruktor wird dann ein TextField mit den Ausmaßen des Ziel-MovieClips erstellt, welches bei Eingabe eines Suchbegriffs und anschließendem Drücken der Enter-Taste eine rekursive Suche beginnend im TUTOR-Objekt startet. Anschließend werden die Ergebnisse unterhalb des Suchfeldes dargestellt, durch Mausklick auf das angezeigte Ergebnis kann dann sofort zum gewünschten Haltepunkt navigiert werden.

Die Klasse SEARCHFRAME enthält keinerlei öffentliche Funktionen bis auf den Konstruktor, da die Suche weder von außerhalb angestoßen noch Einfluss auf die Erscheinungsform sowie das Verhalten des Objektes genommen werden soll. Ebenso findet die Rückgabe der Suchergebnisse über einen der TUTOR-Klasse übergebenen Array statt welcher dann rekursiv durch die untergeordneten Klassen gereicht und mit Werten gefüllt wird. Die Ergebnisse werden in einem dem SEARCHFRAME-Objekt hinzugefügten MovieClip durch Objekte einer erweiterten TextField-Komponente repräsentiert. Da die Suche lediglich dem Auffinden der Ergebnisse und nicht der Steuerung des Videodatenstroms selbst dienen soll, besitzt sie keinerlei derartige Funktionen und delegiert durch Übergabe des aufrufenden TUTOR-Objektes die Auswertung, Steuerung und Ereignisbehandlung an die erzeugten Ergebnis-Textfelder, welche sich selbständig um diese Aufgaben kümmern.

Die TEXTOBJECT-Klasse

Actionscript bietet mit seinem TextField-Objekt eine solide Grundlage für die Darstellung der Suchergebnisse, da dem Ergebnis aber Zieletage, -video sowie -haltepunkt zugeordnet werden soll, musste das TextField durch die TEXTOBJECT-Klasse erweitert werden. Die Erzeugung des Ergebnisses und damit auch der Visualisierungskomponente vom Typ TEXTOBJECT findet im aufrufenden SEARCHFRAME-Objekt durch die Aufrufe

```
txtField=new TEXTOBJECT(where);
txtField.htmlText=found[i].txt;
txtField.target=found[i].targt;
txtField.targetVideo=found[i].vid;
txtField.targetCuepoint=found[i].cue;
txtField.targetFloor=found[i].flr;
```

und die damit verbundene Übergabe des aufrufenden TUTOR-Objektes über den Konstruktor statt, gleichzeitig werden dem Objekt damit die Werte für Etage, Video und Haltepunkt zugeordnet.

Dies ist vor allem nötig damit das TEXTOBJECT die Steuerung des Videos und damit den Sprung zum gewünschten Haltepunkt, die Auswahl der zutreffenden Etage und des zum Haltepunkt gehörenden Videos bei Klick auf das Objekt durchführen kann. Wie auch die SEARCHFRAME-Klasse verfügt das TEXTOBJECT über keinerlei öffentliche Funktionen, da die Steuerung und Ereignisbehandlung innerhalb der Klasse stattfindet und die Interaktion mit dem TUTOR-Objekt nur einseitig von Seiten des TEXTOBJECTs funktioniert. Um auch bei unterschiedlich langen Beschreibungstexten, welche dem Text des jeweiligen Haltepunktes bzw. Objektes entsprechen, ein einheitlichen Design bieten zu können wird das TEXTOBJECT auf die vom SEARCHFRAME gesetzte Breite eingekürzt. Um den Text vollständig darstellen und dem Anwender bei der Auswahl des richtigen Haltepunktes eine Hilfestellung geben zu können, könnte bspw. beim Überfahren des Objektes mit der Maus ein TextField-Objekt erzeugt, welches den vollständigen Text enthält und anschließend beim Verlassen der Maus wieder ausgeblendet wird. Auf der folgenden Seite wird mittels Klassendiagramm, bei welchem nur die Funktionen vermerkt wurden um die Übersichtlichkeit zu gewährleisten, der Anwendungsablauf verdeutlicht.

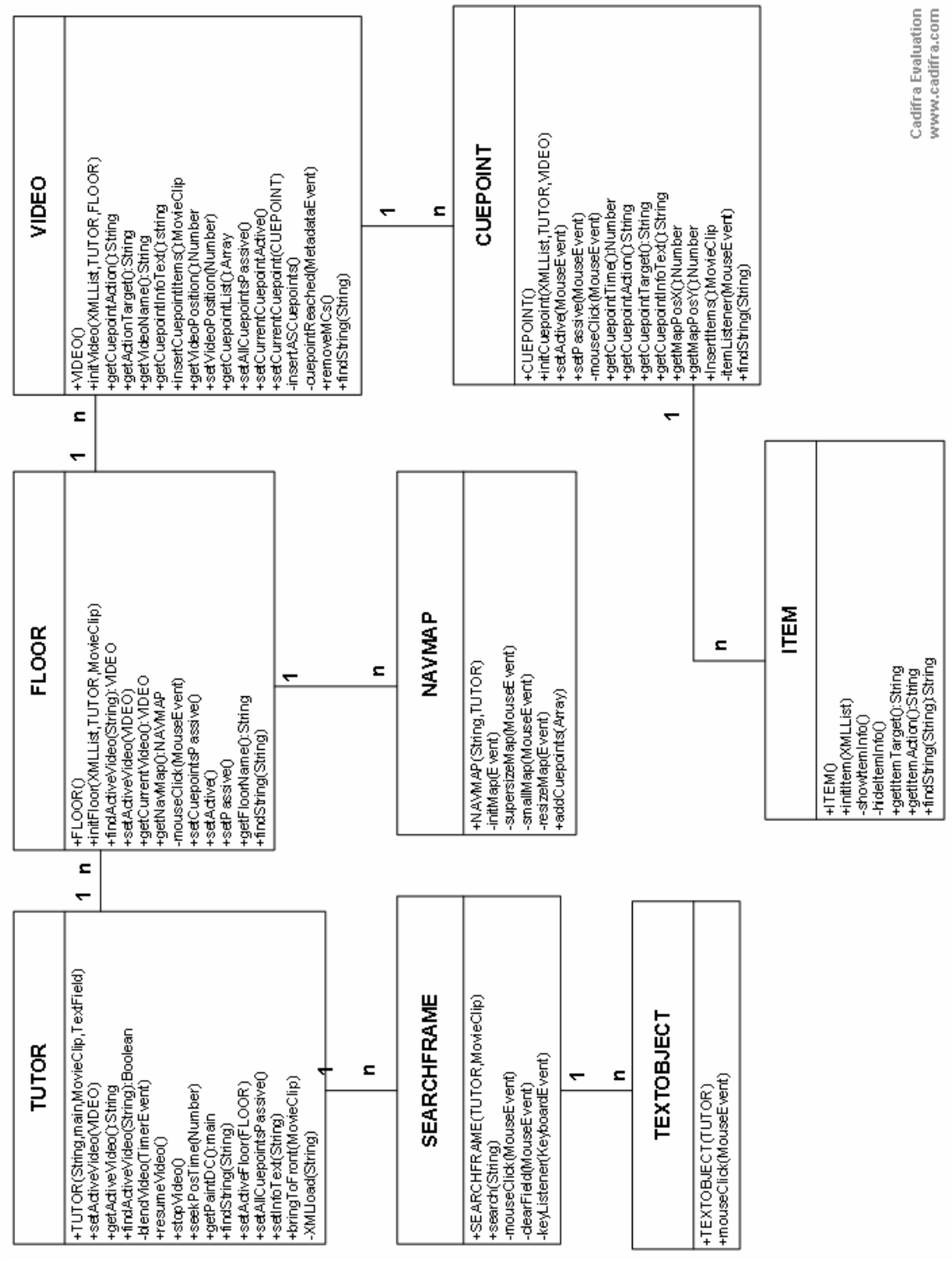

Abbildung 13: Klassendiagramm

5.2 XML-Aufbau

Für die Umsetzung der Szenenbeschreibung für das Projekt "Virtuelle Bibliothek" bietet sich die Arbeit mit XML, der flexiblen Auszeichnungssprache eXtensible Markup Language, an. Damit können im Klartext Hierarchien und Strukturen aufgebaut, mit einem beliebigen Texteditor Änderungen vorgenommen und ohne zusätzliche Software die Szenenbeschreibung entwickelt werden. Dem Aufbau der Anwendung folgend ist auch die XML-Datei in die Kennungen, auch "tags" genannt,

- <TUTOR .../>
- <FLOOR .../>
- <VIDEO .../>
- <CUEPOINT .../>
- und <ITEM .../>

unterteilt, somit bilden die XML-Tags und die jeweiligen Klassen eine logisch konsequente Repräsentation des jeweils anderen. Es existiert nur ein Element vom Typ TUTOR, es bildet das nach XML-Vorgaben notwendige äußerste Element der Hierarchie, geschlossen wird es am Ende der Datei, nachdem die komplette Szenenbeschreibung abgeschlossen ist.

Das FLOOR-Element

Dem TUTOR-Element untergeordnet ist das Element FLOOR, welches jeweils eine Etage des Rundgangs symbolisiert, dabei können beliebig viele, aber normalerweise der Anzahl der Etagen entsprechende, FLOOR-Elemente vorhanden sein.

Das FLOOR-Element muss folgende Attribute besitzen bzw. müssen diese mit Werten belegt sein:

- name (eindeutiger Name der Etage)
- navmap (Pfad zur Bilddatei, welche die jeweilige Ebene darstellt)

Ein Beispiel eines validen FLOOR-Elements würde dann z.B. diese Form haben:

```
<FLOOR name="Erdgeschoss B1" navmap="etage1.jpg">
...
</FLOOR>
```

Diese Etage enthält dann wiederum die ihr zugeordneten Elemente, in der Hierarchie direkt dem FLOOR-Element untergeordnet befinden sich die zur Etage gehörenden Videos.

Das VIDEO-Element

Wie der Name schon sagt, wird durch dieses Element das Videomaterial repräsentiert, welches zur jeweiligen Etage zuzuordnen ist. Innerhalb der FLOOR-Tags können wiederum mehrere VIDEO-Elemente vorhanden sein, da für Abzweige im Etagenvideo eigene Videos vorhanden sein müssen. Zwar ist die Möglichkeit vorhanden, über Spulvorgänge an den Haltepunkten mit nur einem Video das gleiche Ergebnis zu erzielen, dies erfordert aber bspw. an den gewünschten Endpunkten der Abzweige bzw. des Hauptvideos eine entsprechende Haltepunktplatzierung wodurch schnell die Übersicht verloren geht und der logische Aufbau, der vor allem für Wartungs- und Pflegearbeiten von Bedeutung ist, nicht unbedingt ersichtlich ist. Attribute die das VIDEO-Element besitzen muss sind:

- name (entspricht den Pfad des Videos)
- start (optional, falls der Startpunkt des Videos verlegt werden muss, in Sekunden)

Dabei wird durch die Eindeutigkeit des Pfades auch die des Namens gewährleistet, ein Video kann nicht mehrmals in der betreffenden Etage vorkommen. Für den Bibliotheksrundgang ist es allerdings auch nicht von Interesse, das gleiche Video mehrfach vorzufinden, denn jedes Rundgangsvideo erfüllt mit dem von ihm dargestellten Inhalt einen bestimmten Weg, welcher auf der Etage nicht mehrfach existiert. Dem Video untergeordnet sind nun die ihm zuzuordnenden Haltepunkte.

Das CUEPOINT-Element

Diese Haltepunkte werden durch das CUEPOINT-Element vertreten, jedes Video kann mehrere ihm hierarchisch untergeordnete Haltepunkte besitzen. Über diese Zeitmarkierungen wird der Ablauf des Videos gesteuert, denn sie bieten die Möglichkeit, an bestimmten Stellen anzuhalten, Abzweige zu anderen Videos zu erstellen und Textinformationen im laufenden Video zu ändern. Das CUEPOINT-Element muss die Attribute

- time (Zeitpunkt des Auftretens in Sekunden)
- action (auszuführende Aktion am Haltepunkt, gültige Werte sind "stand" um anzuhalten, "goon" um das Video fortzusetzen und "newvideo" um ein neues Video zu laden)
- infotext (der Pfad zur Textdatei, welche die Information enthält)
- posx (der Pixelwert für die horizontale Positionierung auf der Navigationskarte)
- posy (der Pixelwert für die vertikale Positionierung)
- target (optional, Pfad zum Video wenn als Aktion "newvideo" gewählt wurde)

besitzen, um die Funktionalität der Anwendung zu gewährleisten. Ein gültiger Haltepunkt hat dann z.B. diese Schreibweise:

```
<CUEPOINT time="2.0" action="stand" infotext="inf1.txt"  posx="90" posy="400">
...
</CUEPOINT>
```

Beim Eintreten des Haltepunkt Ereignisses wird entsprechend dieser Spezifikation die geforderte Aktion ausgeführt und unabhängig davon der durch den infotext-Parameter genannte Text angezeigt. Die Einordnung auf der Navigationskarte über die posx- und posy-Parameter ist für die geografische Einordnung der Haltepunkte auf der Übersichtskarte von Bedeutung, denn der aktuelle Haltepunkt wird jeweils farblich hervorgehoben, um die Position des Nutzers anzuzeigen, und ist außerdem unerlässlich, damit der Nutzer über das Anklicken des grafischen Platzhalters an die durch den Punkt auf der Karte hervorgehobene Stelle im Rundgang platziert werden kann. Um eine Navigation zu ermöglichen, werden an Haltepunkten mit der Aktion "stand" alle ihm untergeordneten Steuerungssymbole eingeblendet.

Das ITEM-Element

Um die Navigation des Rundgangs an den Haltepunkten zu realisieren, kann jeder Haltepunkt eine beliebige Anzahl von ITEM-Elementen besitzen, welche die jeweilige Steuerungsmöglichkeit repräsentieren. Diese Elemente werden direkt über den Bildinhalt an die Stelle platziert, welche dem Benutzer das bestmögliche Verständnis von dessen Verwendung vermittelt. Dies ist jedoch nur bei Aktionen vom Typ "stand" möglich, Haltepunkte der Typen "newvideo" und "goon" können zwar rein theoretisch Steuerungssymbole vom Typ ITEM besitzen, diese werden aber vollständig ignoriert. Da an diesen Haltepunkten neues Videomaterial geladen bzw. nicht angehalten wird findet auch keine Navigation durch den Nutzer statt.

Das ITEM-Element bildet die unterste Ebene der XML-Hierarchie und muss um den bestimmungsgemäßen Gebrauch zu ermöglichen folgende Attribute besitzen:

- posx (horizontale Platzierung des Symbols über dem Video)
- posy (vertikale Platzierung über dem Video)
- action (die bei Klick auf den Haltepunkt auszuführende Aktion, gültige Werte sind "goon" für das Fortsetzen des Videos, "newvideo" für den Abzweig zu einem anderen Video)
- target (optional, Pfad des Videos bei gewählter Aktion "newvideo")
- pic (Pfad des einzublendenden Steuerungssymbols)
- description (Pfad zur Datei mit der Beschreibung welche beim Überfahren mit der Maus angezeigt werden soll)

Ein Beispiel für ein Symbol zur Fortsetzung des Rundgangs im gleichen Video wäre z.B.:

```
<ITEM posx="300" posy="240" action="goon" pic="pfeil_forw.gif" description="desc.txt"/>
```

Beim Überfahren mit der Maus wird unterhalb der Grafik die angegebene Beschreibung ausgegeben, beim Klick auf die Grafik wird das Video fortgesetzt und das Steuerungssymbol wieder ausgeblendet.

Der XML-Aufbau ist sowohl für Silverlight als auch für Actionscript gleich, deshalb erfolgt keine gesonderte Behandlung dieses Themas im Abschnitt zur Implementierung mit Silverlight in Kapitel 6.

5.3 Besonderheiten der Implementation

Im Allgemeinen gestaltete sich die Umsetzung der geplanten Funktionalität zwar aufwändig aber relativ problemlos, durch die Möglichkeit unter Verwendung von Actionscript 3.0 objektorientiert programmieren zu können lies sich die Anwendung von Grund auf logisch planen und realisieren. Actionscript hat sich in seiner Version 3.0 dank strikter Typisierung, einem durchdachten Namensraum- und Klassensystem, einer fähigen Ereignisbehandlung sowie vieler vorgefertigter Komponenten zu einer ernst zu nehmenden Programmiersprache entwickelt. Die Fähigkeiten reichen vom Erstellen kleiner Filme und Mini-Spiele zu Werbezwecken bis hin zur Möglichkeit, größere Anwendungen im Sinne der Rich-Internet-Applications mit komplexen Funktionen ohne Probleme umsetzen zu können.

Das größte Manko und damit den größten Problemfaktor bei der Implementierung stellte das starre Dogma der FLVPlayback-Klasse hinsichtlich des Haltepunktformates und das Fehlen einer passenden Haltepunkt-Klasse dar. Durch den Umstand, dass die Haltepunkt-Objekte nicht wirklich in Klassenform vorlagen, wurde die Erweiterung dieser um die dringend benötigte und geforderte Funktionalität zu einer zeitraubenden Angelegenheit. Die Videohaltepunkte mussten nicht nur extra neben den eigentlichen CUEPOINTs angelegt werden, sie mussten ebenfalls eindeutig ihren im Funktionsumfang bedeutend umfangreicheren CUEPOINT-Objekten zugeordnet werden. Da der Videohaltepunkt ansich nicht erweiterbar war, bestand lediglich die Möglichkeit, über das Hinzufügen von Attributen zu dem jeweiligen Videohaltepunkt-Objekt eine eindeutige Zuordnung von jenem zu dem eigentlichen CUEPOINT zu gewährleisten. Der abstrakte Object-Typ kann dank der Definition als dynamisch veränderbar Attribute empfangen, welche nicht in seinem Klassenkonstrukt hinterlegt sind.

Beliebige Attribute können mit Werten über den Aufruf

```
meinObjekt.neuerAttributname = "mein neues Attribut";
```

zu der Attributsammlung des Objektes hinzugefügt werden. Leider gibt die FLVPlayback-Komponente beim Erreichen des Haltepunktes nur Werte und keine Referenzen zurück, somit konnte zwar ein CUEPOINT-Objekt mittels

```
theCuePoint.cp=cuePointList[i];
```

vor dem Einfügen des Videohaltepunktes an die Object-Instanz angehängt werden, bei der Rückgabe der Videohaltepunkt-Informationen in der Ereignisbehandlung war dieses dann jedoch verloren gegangen bzw. nur noch als abstraktes Objekt vorhanden. Wahrscheinlich erfolgt innerhalb der FLVPlayback-Klasse eine nicht-offensichtliche Verarbeitung, welche Zahlen und Zeichenfolgen beibehält, komplexere Informationen oder angehängte Objekte aber verwirft bzw. diese in den abstrakten Object-Typ umwandelt. Auch eine implizite Umwandlung in ein CUEPOINT-Objekt mittels

```
var CP:CUEPOINT=eventObject.info.cp as CUEPOINT;
```

führte nicht zum gewünschten Ergebnis, was wiederum bestätigt, dass nicht das bei der Erzeugung des AS-Haltepunktes angelegte Objekt zurückgegeben wird, sondern intern eine Umwandlung stattfindet.

Die einzige, wenn auch unelegante, Lösung bestand darin, über die Indexnummer des CUEPOINT-Objektes durch die Zuweisung

```
theCuePoint.cpt= i;
```

eine Zuordnung zum Videohaltepunkt herzustellen. In der Funktion cuepointReached(eventObject:MetadataEvent) wird anschließend über Abfrage des Parameters eventObject.info.cpt die Indexnummer ausgelesen, der betreffende CUEPOINT ausgewählt, als aktuellen Haltepunkt gesetzt und entsprechende Attributwerte ausgelesen. Somit kann der Haltepunkt zugeordnet und mit dem von ihm gewünschten Verhalten reagiert werden.

Ein weiteres Problem bei der Umsetzung zeigte sich bei der Auswahl des jeweiligen Haltepunktes durch Klick auf das Stellvertretersymbol bzw. direkte Anwahl durch die Anwendungslogik als Abzweig im Video. Da die FLVPlayback-Komponente und damit auch die erweiterte VIDEO-Klasse nur die direkte Ansteuerung von Navigationshaltepunkten, welche direkt ins Videomaterial eingebettet werden müssen, unterstützt, mussten die Actionscript-Haltepunkte über die seekTime-Funktion der Klasse angewählt werden. Dabei wird der Methode seekTime der gewünschte Zeitpunkt in Sekunden übergeben, der angegebene Zeitpunkt wird dann innerhalb des FLVPlayback- oder VIDEO-Objekts gesucht und gesetzt.
Bei der praktischen Umsetzung zeigte sich dann allerdings, dass dies nicht wie gewollt funktionierte. Bei Anwahl des Haltepunktes eines CUEPOINT-Objektes stockte das Bild nur kurz, anschließend fuhr der Player mit dem Abspielvorgang fort, ohne den Haltepunkt zu beachten.

Das Setzen des playheadUpdateInterval, der Zeit nach der der virtuelle Abspielkopf aktualisiert wird, auf kleine Werte wie z.B.

```
this.playheadUpdateInterval=0.01;
```

brachte auch keinerlei Änderung des Verhaltens. Auf den Seiten von Adobe fand sich für dieses Problem keine Lösung, lediglich eine Abspielungenauigkeit von bis zu 0,1 Sekunden wurde erwähnt, resultierend aus der Tatsache, dass die Actionscript-Haltepunkte anders als die Navigationshaltepunkte nicht direkt in den Metadaten des Videomaterials hinterlegt wurden und damit extra verarbeitet werden müssen. Jedoch führte auch die Anwahl von einem Zeitpunkt abzüglich 0,5 Sekunden Verzögerungstoleranz zu dem gleichen Ergebnis, der Haltepunkt wurde übergangen, zum Teil führte diese Methode bei einigen anderen Haltepunkten zum gewünschten Ergebnis, wobei aber eine Toleranz von exakt 2 Sekunden eingerechnet werden musste, um den Effekt bei allen Haltepunkten zu erzielen. Es stellte sich heraus, dass FLV-Videodateien standardmäßig mit Schlüsselbildern im 2-Sekunden-Intervall kodiert werden, und Actionscript bzw. die FLVPlayback-Klasse bieten keine Unterstützung von Haltepunkten, welche außerhalb der Schlüsselbilder liegen. Bei der Einbettung von Navigationshaltepunkten legt Actionscript für den gewünschten Zeitpunkt ein zusätzliches Schlüsselbild an, um das Anhalten und Verarbeiten des Haltepunktes zu ermöglichen. Dieses 2-Sekunden-Intervall führte dazu, dass Haltepunkte, welche an Zeitpunkten mit ungerader Sekundenzahl gelegen waren teilweise verarbeitet wurden und die restlichen Haltepunkte keine Beachtung fanden. In Zusammenhang mit dem auftretenden Intervall von 0,25 Sekunden, welches zur von Adobe angegebenen Verzögerungstoleranz von 0,1 Sekunden addiert werden muss, um das Abspielintervall zu treffen und das Ereignis zu verarbeiten, ist es erforderlich, selbst an geradzahligen Haltepunkten auf eine Zeit von theoretisch mindestens 0,35 Sekunden vor dem eigentlichen Zeitpunkt zu spulen.

Die einzige Möglichkeit, dies zu umgehen ist die Kodierung des Videomaterials mit einem Schlüsselbildintervall von möglichst geringer Größe. Da eine Toleranz von 0,2 Sekunden durchaus zumutbar erscheint, wird das Videomaterial mit einem Schlüsselbildintervall von 5 Bildern kodiert, da bei 25 Bildern pro Sekunde diese 5 Bilder einem Intervall von 0,2 Sekunden entsprechen. Eine Kodierung mit noch kleinerem Schlüsselbildintervall erscheint unsinnig, da rechnerisch alle 2,5 Bilder ein Schlüsselbild gesetzt werden müsste, um ein Intervall von 0,1 Sekunden einhalten zu können, was durch die Notwendigkeit der Ganzzahligkeit ein Schlüsselbildintervall von abgerundet 2 Bildern bedeuten würde. Da die Schlüsselbilder trotz Kompression die größte Datenmenge im Videodatenstrom beanspruchen bedeutet die Herabsetzung des Intervalls und die damit verbundene Zunahme der Schlüsselbildanzahl eine signifikante Erhöhung des Datenaufkommens, welche bei einem 2-Bild-Intervall einen 50 prozentigen Anteil an Schlüsselbildern und damit eine Erhöhung der Schlüsselbildanzahl um 2500 Prozent bedeuten würde. Bei einem Intervall von 5 Schlüsselbildern bilden diese nur einen durchaus vertretbaren Anteil von 20 Prozent, die Anzahl erhöht sich um 1000 Prozent bzw. steigt auf das Zehnfache an. Bei Kodierung in mittlerer Qualität, welche 400KBit/s entspricht und einer Bildgröße von 512x288 Bildpunkten ergeben sich ein Resultat in folgender Form:

Schlüsselbildintervall	Größe in KByte
50	10.861
5	17.955
2	35.933

Tabelle 1: Dateigröße Flash-FLV

Damit verursacht das Verzehnfachen der Schlüsselbildanzahl und das damit geschaffene Intervall von 0,2 Sekunden nicht einmal eine Verdopplung der Dateigröße, wobei ein Intervall von 0,1 Sekunden eine rund 3,5-fache Dateigröße verursachen würde.

Ein Herabsetzen der Datenrate stellt keine Lösung dar, denn dies führt zwangsläufig zu einer Verschlechterung der Qualität welche in keinem Verhältnis zu der um 0,1 Sekunden herabgesetzten Verzögerung steht. Im Anwendungsablauf ist der Unterschied ohnehin nicht feststellbar, eine Toleranz von 0,2 Sekunden bringt für die Anwendung keinen spürbaren Nachteil, die Verzögerung muss lediglich bei der Implementation beachtet werden. Durch die Integration der Verzögerung in die Programmlogik müssen die Zeitpunkte in der XML-Datei nicht angepasst werden, bedürfen somit keiner Rechenarbeit und verringern den Wartungsaufwand.

Im Übrigen gestaltete sich die Programmierung und Umsetzung der Anwendung nach Einarbeitung in die überarbeitete AS3-Sprache lediglich geringe Probleme, welche jedoch aus den funktionellen Unterschieden zwischen dem bereits bekannten Actionscript 2.0 und der neuen Version resultierten. Durch z.B. die komplette Neugestaltung der XML-Verarbeitung musste das streng hierarchische Modell, ohne Möglichkeit einer Geschwisteranwahl, welche durch eine Indizierung ersetzt wurde, adaptiert bzw. umgesetzt werden. Ebenfalls wurde für das Laden der XML-Datei eine Loader-Instanz benötigt, was in AS2 noch nicht der Fall war und sich wie ein roter Faden durch die neue Version von Actionscript zieht. Leider wird deshalb vom Benutzer auch ein, wenn auch eher geringer, Mehraufwand gefordert, um die bisher gewohnte Funktionalität zu erreichen. Bisher wurde ein Bild über den Aufruf

```
var myMC:MovieClip=new MovieClip();
myMC.loadMovie("meinBild.jpg");
```

in den neu erzeugten MovieClip geladen.

Nun muss mittels

```
var imgLoader:Loader=new Loader();
var myMC=new MovieClip();
imgLoader.load(new URLRequest("meinBild.jpg"));
myMC.addChild(imgLoader);
```

ein zusätzliches Loader-Objekt erzeugt, mittels URLRequest das Bild geladen und anschließend dem MovieClip hinzugefügt werden. Dies wirkt zuerst sehr umständlich, ist aber durchaus im Sinne des neuen Actionscript-Konzepts und der damit verbundenen Trennung von Objekten und Laderoutinen.

Eine weitere Besonderheit bei der Implementation stellt die Verarbeitung der Hyperlinks beim Klick-Ereignis auf eben diese dar. Zwar werden Hyperlinks als solche erkannt, jedoch findet standardmäßig keine Ereignisbehandlung für diese Elemente statt. Über den Aufruf

```
tField.addEventListener(TextEvent.LINK,openURL);
```

muss dem TextField-Objekt, welches den Text enthält, eine Behandlungsroutine für dieses Ereignis zugewiesen werden. Damit wurde zwar die Verbindung zwischen Link und Verarbeitung hergestellt, aber eine Auswertung kann noch nicht stattfinden, denn vor der eigentlichen URL in der HREF-Angabe des Link-Elementes erwartet die Komponente eine Kennzeichnung als Event. Somit muss der Text beim Laden durchsucht und die Kennzeichnung als Event über die Zeile

```
itemText=itemText.replace("href=\"","href=\"event:");
```

dem Hyperlink hinzugefügt werden.

Anschließend wird über die Zeile

```
navigateToURL(new URLRequest(events.text),"_blank");
```

ein neues URLRequest-Objekt mit dem Text des Events und damit der eigentlichen URL angelegt und in einem neuen Fenster des Browsers geöffnet, um den Anwendungsablauf nicht zu stören.

Bis auf diese wenigen Hindernisse gestaltete sich die Implementation der Anwendung problemlos, das OOP-Konzept und die zahlreichen vorgefertigten und anpass- bzw. erweiterbaren Komponenten begünstigten ein rasches Voranschreiten des Entwicklungsprozesses.

6. Implementation mit Silverlight 1.0

In diesem Abschnitt wird der grundlegende Aufbau der Silverlight-Anwendung betrachtet und auf auftretende Besonderheiten bzw. Probleme näher eingegangen.

6.1 Aufbau der Anwendung

Ebenso wie die Implementation mit Actionscript ist auch die Umsetzung mit Silverlight objektorientiert geplant und vorgenommen worden. JavaScript, die Programmiersprache der Silverlight-Anwendungslogik, unterstützt in ihren Grundzügen die objektorientierte Programmierung, wobei jedoch nicht von einer vollständigen OOP-Sprache ausgegangen werden kann. Es gibt kein Klassenkonzept an sich mit Vererbungsmöglichkeiten, wie es bei C++, Java, Delphi oder vielen anderen Programmiersprachen der Fall ist. Dennoch bietet auch diese Sprache die Möglichkeit, in einem gewissen Maß nach dem OOP-Konzept arbeiten zu können.

Auf die in der Actionscript-Umsetzung entworfene Klasse TUTOR wurde komplett verzichtet, die Funktionalität der Klasse konnte in der der XAML zugeordneten Datei "Page.xaml.js" nachgebildet werden, in ihr wird der Klassentyp *Page* definiert. Diese Datei bzw. diese Klasse wird standardmäßig schon zur Abbildung der hinter der Designbeschreibung liegenden Programmlogik verwendet, für welche die TUTOR-Klasse in der AS3-Implementierung verantwortlich war. Somit hätte die Auslagerung der Logik in eine zusätzliche Klasse für diesen prototypischen Vergleich keinen offensichtlichen Vorteil sondern lediglich eine zusätzliche JavaScript-Datei zur Folge und wurde aus diesem Grund weggelassen.

Dennoch wurde die gleiche Funktionalität erreicht, denn es wird in dieser Datei

- das Laden und Verarbeiten der XML-Datei
- das Anlegen der einzelnen Etagen und deren Verwaltung
- die Verarbeitung des Lade-Ereignisses der Anwendung
- die Etagensteuerung
- sowie die Bereitstellung einer Referenz auf die Bühne

vorgenommen und damit die Funktionalität der TUTOR-Klasse nachgebildet. Dabei befinden sich die Funktionen, welche sich auch innerhalb der XAML-Definition zur Event-Behandlung referenzieren lassen müssen außerhalb der Klasse, um für diese aufrufbar zu sein. Eine Verlagerung in den Funktionsblock der Klasse führt zur Unsichtbarkeit der Funktion gegenüber den XAML-Elementen. Durch die strikte Trennung von Design und Funktionalität entstehen eben solche Effekte, da die XAML-Datei den Klassenaufbau der JS-Datei nicht kennt und die Verbindung dieser zwei Dateien erst in der JS-Datei stattfindet. Im Übrigen finden die gleichen Klassen Verwendung wie auch schon bei der Umsetzung mit Actionscript 3.0, das wären die Klassen

- iFloor
- iVideo
- iCuepoint
- iItem

sowie die Klassen für die Übersichtskarten und Suchfunktionen. Diese Klassen besitzen zwar namentlich abweichende aber in ihrer Funktion übereinstimmende Funktionen, die Funktionalität der Actionscript-Anwendung muss für den prototypischen Vergleich mit der der Silverlight-Anwendung im größtmöglichen Umfang übereinstimmen. Es wird die gleiche XML-Datei wie bei der Actionscript-Variante verwendet.

6.2 Besonderheiten bei der Implementation

Bei der Umsetzung mit Microsoft Silverlight gibt es zahlreiche Unterschiede zu der Herangehensweise mit Flash und Actionscript. Diese resultieren vor allem aus dem konzeptionellen Unterschied der Sprachen, wobei Actionscript und Flash eine Einheit bilden und Silverlight im Gegensatz dazu eine Mischform zweier bereits existierender Konzepte darstellt.

Das erste Problem zeigte sich bereits beim Versuch, die XML-Datei zu laden, welche die Szenenbeschreibung enthält, denn eine spezielle Funktion dafür existiert nicht. Deshalb musste ein Downloader-Objekt über die Codezeilen

```
LoadData: function(sender, args)
{
        this.plugin=sender.getHost();
        var dler=this.plugin.createObject("downloader");
        var dlerevent= dler.addEventListener("completed",this.createObjects);
        dler.open("GET","Test2.xml");
        dler.send();
}
```

angelegt, die Zieldatei spezifiziert und der Downloadvorgang angestoßen werden. Nachdem der Inhalt der Datei erfolgreich gelesen wurde, musste danach über

```
var xmlfile=string2XML(sender.ResponseText);
```

eine XML-Datei angelegt werden, denn der Downloader liefert kein XML-Objekt, sondern den Inhalt der XML als String. Diese Funktion string2XML musste zuvor angelegt werden, denn JavaScript und somit Silverlight bieten keine entsprechende Lösung dafür.

Dabei muss bei der Erzeugung des XML-Objekts der Browser beachtet werden, in dem die XML erstellt werden soll, denn zwischen Internet Explorer und den restlichen Browsern gibt es gravierende Unterschiede. Für den Internet Explorer muss ein neues ActiveX-Objekt angelegt werden, für die anderen Browser wird hingegen ein DOMParser-Objekt angelegt, welches die Erstellung des XML-Objekts übernimmt.

```
function string2XML(xmlstring)
{
 var doc;
 if (window.ActiveXObject)
 { //IE5
        doc = new ActiveXObject("Microsoft.XMLDOM");
        if (!doc.loadXML(xmlstring))
        {
                alert(doc.parseError.reason);
        }
 }else
 { //Standard-Browser
        var myDOMparser=new DOMParser();
        doc=myDOMparser.parseFromString(xmlstring);
 }
 return doc;
}
```

Das zweite Problem stellt die fehlende Möglichkeit zur Erstellung eigener Elemente bzw. Attribute innerhalb der XAML-Definition dar, zwar liefert Microsoft gut durchdachte Elemente, jedoch sind diese nur in dem von Microsoft vorgesehenen Maße an die eigenen Bedürfnisse anpassbar. Jedes Element besitzt eine Reihe von Attributen, mit denen ihr Aussehen angepasst werden kann, Eventhandler können für die jeweils unterstützten Ereignisse eingebunden werden und verknüpfen damit das Design mit der Programmlogik.

Diese eng abgesteckten Grenzen sorgen dafür, dass z.B. die Erweiterung eines Elements um selbst gewählte Attribute mittels

```
<Image ... meinAttribut="attributWert" .../>
```

zur Generierung eines Fehlers und Abbruch der Verarbeitung führt. Hier müssen vorhandene Attribute genutzt werden, und dann eventuell durch die Programmlogik des JavaScript-Teils der Anwendung zusammengesetzte Attribute interpretiert werden. So ist es bspw. nötig, für die Videohaltepunkt-Elemente vom Typ TimelineMarker über die Zusammensetzung des Textes eine Referenz zum jeweiligen Haltepunkt mittels

```
<TimelineMarker Type="stand" Text="0_0_1" Time="0:0:15"/>
```

nach der Namenskonvention "<Etage>_<Video>_<Haltepunkt>" herzustellen, wobei in diesem Beispiel der Haltepunkt für die nullte Etage, nulltes Video und dessen ersten Haltepunkt steht. Somit geht ein eventuell nützlicher Platzhalter für Text verloren.

Ungeschickt gelöst zeigt sich auch die Erzeugung neuer Elemente, egal ob Bild-, Haltepunkt- oder Video-Objekte, denn diese können unter JavaScript nicht direkt angelegt werden, da der grafische Aufbau wie bereits beschrieben von der Programmlogik komplett unabhängig ist. Der Umweg führt hier über die Erzeugung eines XAML-Objektes welches dann in die bestehende XAML-Beschreibung eingefügt wird. Der durch die Zeilen

```
var tlmString = '<TimelineMarker Type="none" Text="x" Time="0:0:15"/>';
var myMarker = this.paintDC.getHost().content.createFromXaml(tlmString);
```

angelegte Haltepunkt muss anschließend zum bestehenden MediaElement-Objekt hinzugefügt werden.

```
var mE=document.getElementById("SilverlightControl").content.findName("myVid");
mE.markers.add(myMarker);
```

Zwar ist eine Bearbeitung der angegebenen Parameter des TimelineMarkers über

```
myMarker.Type = "stand";
myMarker.Text = "meinText";
myMarker.Time = "0:0:15";
```

möglich, jedoch erscheint diese Herangehensweise der Erzeugung neuer Objekte über die Verarbeitung eines Strings etwas unbeholfen und stellt dennoch, aufgrund der fehlenden Möglichkeit für diese Elemente repräsentative Objekte in JS anzulegen, die einzige Lösung dar.

Eine weitere Schwierigkeit liegt in der XAML-Verarbeitung, welche Silverlight intern durchführt, es können Probleme beim nachträglichen Hinzufügen von XAML-Elemente über JavaScript-Programmcode auftreten, welche weder offensichtlich noch nachvollziehbar sind. Soll in der XAML-Datei einen Element ein Name zugewiesen werden, was bspw. zur eindeutigen Zuordnung eines Elementes nötig ist, erfolgt dies über das x:Name-Attribut, z.B. wird mittels

```
<Image Source="test.png" x:Name="Bild1"/>
```

der Grafik der eindeutige Name "Bild1" zugeordnet, um auf dieses Bild dann mittels

```
document.getElementById("SilverlightControl").content.findName("Bild1");
```

zugreifen zu können. Diese Eigenschaft ist für alle XAML-Elemente verfügbar und verlangt die Eindeutigkeit des Bezeichners.

Leider ist diese Möglichkeit der Referenzierung bei nachträglich eingefügten Bildelementen nicht möglich, dieses Attribut darf nicht gesetzt werden und führt zum Abbruch der Verarbeitung. Dieses Attribut wäre für den virtuellen Bibliotheksrundgang insofern von Interesse, dass sich die am Haltepunkt eingeblendeten Steuerungssymbole wiedererkennen lassen und die Verarbeitung vorgenommen werden kann. Da JavaScript und damit Silverlight auch den Vergleich zweier Objekte auf Gleichheit hinsichtlich ihrer Referenz nicht beherrscht, ist eine Abfrage über

```
var imgString = '<Image Source="test.png"/>';
var myImg = this.paintDC.getHost().content.createFromXaml(imgString);
....
if (sender==myImg) {....}
```

erfolglos, auch wenn es sich um das gleiche Objekt handelt. Ein Vergleich der den Elementen zugrunde liegenden XAML-Strings ist ebenfalls nicht möglich, da der XAML-Text nicht auslesbar ist und es außerdem durchaus vorkommen kann, dass mehrere Elemente den gleichen XAML-Text verwenden.

Abhilfe schafft hier lediglich die Bindung des erzeugten XAML-Elementes an das jeweilige Item-Objekt über die Ereignisbehandlung des XAML-Elementes. Hierfür wird vor dem Anlegen des JavaScript Objektes das XAML Element z.B. mittels

```
var picStr='<Image Opacity="0.5" Source="forward.png" Canvas.ZIndex="11"/>';
var image = this.paintDC.getHost().content.createFromXaml(picStr);
```

erzeugt und anschließend über

```
var item = new iItem(control, image, tempArray[j], j, this);
```

dem Stellvertreterobjekt übergeben.

Dieses führt dann über die Aktionen

```
iItem = function(control, target, vals, id, cp)
{
   this.target = target;
   this.id = id;
   this.cp = cp;
   this.name = vals.getAttribute("name");
   this.pic = vals.getAttribute("pic");
   this.posx = vals.getAttribute("posx");
   this.posy = vals.getAttribute("posy");
   this.action = vals.getAttribute("action");
   this.actiontarget = vals.getAttribute("target");
   this.description = vals.getAttribute("description");

   this.target.cursor = "Hand";
   this.target.addEventListener("MouseEnter", Silverlight.createDelegate(this,
this.handleMouseEnter));
   this.target.addEventListener("MouseLeave", Silverlight.createDelegate(this,
this.handleMouseLeave));
   this.target.addEventListener("MouseLeftButtonDown", Silverlight.createDelegate(this,
this.handleMouseDown));
   this.target.addEventListener("MouseLeftButtonUp", Silverlight.createDelegate(this,
this.handleMouseUp));

   this.target["Canvas.Top"] = this.posy;
   this.target["Canvas.Left"] =this.posx;
   this.target.Source = this.pic;
}
```

die Zuordnung des Elementes zur eigenen Variable target und die Verknüpfung des XAML-Ereignisses mit der zum Objekt gehörenden Funktion handleMouseDown sowie die Positionierung durch, damit kann beim Klick-Ereignis direkt in der Eventhandler-Funktion des Item-Objektes über this.AttributName auf die gesetzten Attribute wie z.B. auszuführende Aktion oder Ziel der auszuführenden Aktion zugegriffen werden. Dies stellt mangels Erweiterbarkeit der XAML-Elemente über JavaScript die eleganteste und auch einzige Möglichkeit dar, XAML-Elemente zu erweitern und mit entsprechender Logik zu versehen.

In gleicher Weise wurden nicht nur die Steuerungssymbole, sondern auch die grafischen Platzhalter für die Etagen und Haltepunkte sowie die einzelnen Videoobjekte um die Funktionalität bereichert, welche bei der AS3-Implementierung über die Erweiterung vorhandener MovieClip-, FLVPLayback- und TextField-Objekte realisiert werden konnte.

Ein weiteres Problem stellte dann, wie auch bei der Umsetzung mit Actionscript, der relativ hoch gewählte Schlüsselbildabstand im Videomaterial dar. Im Gegensatz zu AS3 führt hier das Setzen eines Haltepunkts an eine Position welche kein Schlüsselbild enthält nicht zum Ignorieren des Haltepunkts, wie es bei Actionscript der Fall war. Unter Silverlight wird bei der Positionierung außerhalb eines Keyframes der Haltepunkt angesteuert und die entsprechende Ereignisbehandlungsprozedur aufgerufen. Da jedoch kein Schlüsselbild vorhanden ist und der Silverlight MediaElement-Komponente die Fähigkeit fehlt, bei der Positionierung den Bildinhalt wie beim Abspielvorgang aus den vorangegangenen Schlüsselbildern und den Zwischenbildern zu berechnen, werden die Bildinformationen der betreffenden Stelle nicht korrekt wiedergegeben. So wird z.B. beim Anhalten an dieser Stelle und Einblenden der Steuerungssymbole der Bildinhalt der Videoposition angezeigt, der auch vor dem Spulvorgang auf dem MediaElement zu sehen war, erst nachdem der Abspielvorgang fortgesetzt wird zeigt die Videokomponente den korrekten Inhalt an. Wurde der Haltepunkt aber an einen Zeitpunkt im Video gesetzt, welcher ein Schlüsselbild enthält, funktioniert die Positionierung des virtuellen Abspielkopfes ohne Probleme, die Steuerungssymbole werden an der richtigen Stelle angezeigt. Abhilfe schafft auch hier das Setzen des Schlüsselbildintervalls bei der Kodierung der WMV-Datei auf einen möglichst niedrigen Wert, wie es auch beim FLV-Pendant der Fall war. Standardmäßig gibt der Microsoft Windows Media Encoder ein Schlüsselbildintervall von 6 Sekunden vor.

Im Vergleich zu dem mit 2 Sekunden schon recht groß gewählten Intervall der FLV-Datei bedeutet dies die dreifache Intervalldauer. Der niedrigste vom Windows Media Encoder unterstützte Wert für die Eingabe der Intervalldauer ist eine Sekunde, kleinere Werte sowie Zahlen mit Kommastellen werden nicht akzeptiert. Ein Schlüsselbildabstand von einer Sekunde, entsprechend 25 Bildern, ist in diesem Fall vollkommen ausreichend, denn die MediaElement-Komponente kann sowieso nur auf ganzzahlige Sekundenwerte gesetzt werden, resultierend aus dem TimeSpan-Format, in welchem die Eingabe erfolgt. Damit muss um am gewählten Zeitpunkt anhalten zu können ein Vorlauf eingerechnet und somit die Position auf eine Sekunde vor eintreffen des Haltepunktereignisses gesetzt werden. Diese ungenaue Positionierung fällt bei der Ausführung zwar auf, ist aber nicht zu umgehen und ist mit einer Sekunde für kleine Standardanwendungen akzeptierbar, auch wenn hier seitens Microsoft mit etwas mehr Genauigkeit hätte gearbeitet werden können. Für den virtuellen Rundgang ist diese Zeitdifferenz sowie die Ungenauigkeit bei der Positionierung nicht akzeptabel.

Anders als bei Actionscript hat die Änderung des Schlüsselbildintervalls nur einen sehr geringen Einfluss auf die Dateigröße, da der Windows Media Encoder bei der Kodierung diese Schlüsselbilder bereits einrechnet und somit nur sehr geringe Abweichungen auftreten, wie aus folgender Tabelle ersichtlich:

Schlüsselbildintervall in Sekunden	Größe in KByte
6	10.325
1	10.349

Tabelle 2: Dateigröße Silverlight WMV

Auch die Qualität des erzeugten Videofilms verschlechtert sich nicht in dem Verhältnis, wie es bei gleich bleibender Datenrate mit einer erhöhten Anzahl an Schlüsselbildern zu erwarten wäre.

Zwar sind vor allem bei Vollbildansichten kleinere Qualitätsunterschiede erkennbar, diese sind jedoch beim Einsatz in Originalgröße von 512x288 Bildpunkten so gut wie nicht zu erkennen und somit für die Praxis nicht von Bedeutung.

Die Nachteile der Mischtechnologie zeigt sich auch bei anderen Ungereimtheiten zwischen den Komponenten und der Programmiersprache JavaScript Ein Beispiel ist das Standardformat für die zeitliche Einordnung vom Format TimeSpan, welches sich innerhalb JavaScript nicht erzeugen lässt. So hat das TimeSpan-Format, welches über

```
var mySpan=mediaElement.Position;
```

ausgelesen werden kann, eine Eigenschaft Seconds welche die Sekundenanzahl der Videoposition zurück gibt. Mit dieser Codezeile wird nun mittels

```
alert(mySpan.Seconds);       //Alte Sekundenanzahl wird ausgegeben z.B. "15"
mySpan.Seconds=10;           // setzen der Haltepunktzeit
alert(mySpan.Seconds);      // Anzeige der neuen Sekundenanzahl "10"
```

eine als "mySpan" bezeichnete Variable vom Typ TimeSpan angelegt, deren Position ausgegeben und verändert werden kann. Die direkte Manipulation des Seconds-Parameter ist zwar möglich, jedoch führt

```
mediaElement.Position.Seconds=10;
```

nicht zum gewünschten Erfolg, da das MediaElement nicht selbstständig auf diese Änderung der Position reagiert, der neue Wert wird zwar zugewiesen aber sofort wieder verworfen, es steht auch kein entsprechende Ereignisbehandlungsfunktion zur Verfügung, über welche ein Eingriff eventuell möglich wäre. Wenn nun aber die komplette Position des MediaElements neu zugewiesen wird, führt die Komponente eigenständig den Spulvorgang auf die übergebene Position durch.

Über die Zeilen

```
var mySpan=mediaElement.Position;     //Erzeugung eines TimeSpan-Objektes
mySpan.Seconds=30;                    //Setzen der Sekundenanzahl
mediaElement.Position=mySpan;         //Zuweisung der kompletten Position
```

wird ein manipulierbares Objekt vom Typ TimeSpan erzeugt, indem die Position des MediaElements ausgelesen wird, anschließend wird das Seconds-Attribut der neuen Variable verändert und anschließend das TimeSpan-Objekt der Position des MediaElements zugewiesen, woraufhin das MediaElement die Aktualisierung durchführt.

Eine andere Möglichkeit der Erzeugung eines solchen Objektes ist per JavaScript leider nicht möglich, jedoch findet sich bei Kenntnis des Formates, welches die zeitabhängigen Komponenten für ihre Platzierung benötigen, auch dafür eine Lösung. Beim Anlegen der XAML-Elemente geschieht dies über einen String, welcher die Angaben für Stunden, Minuten und Sekunden enthält. Ein Haltepunkt wird z.B. an die Stelle 0 Stunden, 1 Minute und 10 Sekunden gesetzt, indem man dem Objekt über

```
<TimelineMarker ... Time="0:1:10" />
```

die Werte getrennt durch Doppelpunkt ähnlich der Anzeige einer Digitaluhr zuweist, dabei können führende Nullen weggelassen werden. Das TimeSpan-Objekt ist lediglich eine intern durch Silverlight verarbeitete Repräsentation dieser Zeitangabe. In JavaScript bietet sich nun die Möglichkeit an, dem Position-Attribut einen nach dieser Syntax zusammengesetzten String zuzuweisen.

```
myTimelineMarker.Time="0:2:30";     //Setzen eines Haltepunkts
//bzw.
myMediaElement.Position="0:2:29";   //Setzen der Videoposition
```

Die jeweilige XAML-Komponente verarbeitet diese Wertzuweisung ebenso wie die Zuweisung eines TimeSpan-Objektes, auch die Positionierung des Videoinhaltes beim MediaElement funktioniert und die Aktualisierung des Bildinhaltes wird durchgeführt. Andere Möglichkeiten der Abspielkopfpositionierung bietet das MediaElement ohnehin nicht, sodass diese beiden Möglichkeiten verwendet werden müssen.

7. Zusammenfassung

Im Rahmen dieser Diplomarbeit sollte anhand der konkreten Anwendung "Virtuelle Bibliothek", welche für das eCampus-Projekt entwickelt werden sollte, mittels prototypischer Implementierung die Eignung sowie Vorzüge bzw. Nachteile der einzelnen Technologien dafür untersucht werden.

Schon aus der theoretischen Betrachtung des Funktionsumfangs der beiden Technologien ergibt sich, dass Flash eindeutig das vorteilhaftere und damit zur Umsetzung zu verwendende Werkzeug ist. Trotz einiger Schwachstellen bietet Adobe Flash in Verbindung mit Actionscript 3.0 neben der umfangreichen Bibliothek und der konsequenten Durchsetzung des Klassenkonzepts mitsamt Vererbung das umfassendere Gesamtpaket. Streaming ist im Gegensatz zu Silverlight 1.0 auch ohne Installation einer zusätzlichen Server-Software möglich, was wiederum für die Anwendung "Virtuelle Bibliothek" zwingende Voraussetzung ist, da sie unter der aktuellen Konfiguration des Webservers der HTW-Dresden lauffähig sein soll. Auch sind die Steuerungsmöglichkeiten des Datenstroms weitaus umfangreicher als bei Silverlight und zeigen damit deutlich, dass Flash in Hinsicht auf den Abschnitt Streaming zur Zeit mehr als einen Schritt voraus ist.

Auch bei der prototypischen Implementierung zeigte sich ein deutlicher Vorteil von Flash mit AS3 gegenüber Silverlight, allein das Nichtvorhandensein von Vererbungsmöglichkeiten, der nur umständlich realisierbare Klassenaufbau aufgrund der Verwendung von JavaScript als Programmiersprache, die hohe Ungenauigkeit beim Setzen der Haltepunkte sowie die umständliche Fehlersuche lassen die Entscheidung für Flash als das geeignetere Werkzeug für diese konkrete Aufgabe eindeutig ausfallen.

Eine Umsetzung mit Silverlight wäre zwar grundsätzlich möglich, jedoch mit erhöhtem Aufwand verbunden, ebenfalls müsste das Videomaterial auf einen eigenen Server ausgelagert werden, da sonst keine Möglichkeit besteht, Videostreaming umzusetzen.

Abschließend lässt sich sagen, dass Microsoft mit Silverlight eine solide Technologie geschaffen hat, welche das XAML-Konzept weiterführt und sich bei entsprechender Weiterentwicklung unter Umständen bald zu einem gleichwertigen Konkurrenten zu Flash entwickeln könnte.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AIFF	Audio Interchange File Format
AJAX	Asynchronous JavaScript and XML
AS3	ActionScript Version 3.0
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
CGI	Common Gateway Interface
DSL	Digital Subscriber Line
ECMA	European Computer Manufacturers Association
EDV	Elektronische Datenverarbeitung
FLV	Flash Video
HTML	Hypertext Markup Language
HTW	Hochschule für Technik und Wirtschaft
JSP	JavaServer Pages
MP3	MPEG-1 Audio Layer 3
MPEG	Moving Picture Experts Group
PHP	Personal Home Page Tools / Hypertext preprocessor
SGML	Standard Generalizied Markup Language
URI	Unified Resource Identifier
URL	Unified Resource Locator
VLC	Variable Length Coding
VP6	TrueMotion VP6
XML	eXtensible Markup Language
WAV	Wave-Dateiformat
WLAN	Wireless Local Area Network
WWW	World Wide Web
WYSIWYG	What you see is what you get

Literaturverzeichnis

[Fre06] Freeman, Eric; Freeman, Elisabeth; Sierra, Kathy; Bates, Bert: Entwurfsmuster von Kopf bis Fuß. Beijing u.a.: O'Reilly, 2006

[Hau08] Hauser, Tobias; Kappler, Armin; Wenz, Christian: Actionscript 3 - das Praxisbuch. Bonn: Galileo Press, 2008

[Mac07] MacDonald, Matthew: Silverlight and ASP.NET Revealed. Berkley,CA: Apress, 2007

[Moo07] Moock, Collin: Essential ActionScript 3.0. Beijing u.a.: O'Reilly, 2007

[Mor08] Moroney, Laurence: Introducing Microsoft Silverlight 1.0. Redmond: Microsoft Press, 2008

[Rot03] Rothfuss, Günther[Hrsg.]; Eisenbiegler, Jörn: Contentmanagement mit XML. Berlin: Springer Verlag, 2003

[Str08] Stropek, Rainer; Huber, Karin: WPF und XAML: Programmierhandbuch. Unterhaching: entwickler.press, 2008

Internetquellen

[www01] Wikipedia
http://de.wikipedia.org/

[www02] HTW Dresden
http://www.htw-dresden.de

[www03] Building a better User Experience
http://www.mindfusioncorp.com/weblog/

[www04] Microsoft
http://www.microsoft.com

[www05] Flash&Video
http://www.video-flash.de/

[www06] Silverlight
http://www.silverlight.net

[www07] Flashhilfe
http://www.flashhilfe.de

[www08] Adobe
http://www.adobe.com

[www09] Silverlight Blog
http://www.silverlightblog.de

[www10] Silverlight Forum
http://www.silverlight-forum.de

[www11] Microsoft Dev
http://dev.live.com/silverlight

[www12] UML Tutorial
http://ivs.cs.uni-magdeburg.de/~dumke/UML/2.htm

[www13] Web Design Library
http://www.webdesign.org

Glossar

ActiveX	Softwarekomponenten-Modell von Microsoft für aktive Inhalte
Blu-Ray Disc	digitales optisches Speichermedium
Browser	Software zur Anzeige von Internetseiten
Client	Computerprogramm, das Nachrichten mit einem Server austauscht
CSS	Formatierungssprache für Webinhalte
DirectX	Programmierschnittstelle von Microsoft für Multimedia
Document Object Model	Programmierschnittstelle für den Zugriff auf HTML / XML
Drag-and-Drop	Elemente können gezogen und über einem Ziel fallengelassen werden
E4X	ECMAScript for XML, XML-Unterstützung für ECMAScript basierende Skriptsprachen
ECMAScript	Skriptsprache, Sprachkern von JavaScript
Eventhandler	Funktion zur Ereignisbehandlung
H.264-Codec	Standard für hocheffiziente Videokompression
Hosting	Bereitstellung von Inhalten
Keyframe	Schlüsselbild
Metadaten	allgemeine Daten, welche Informationen über andere Daten beinhalten
MPEG-4	Videoformat / Verfahren zur Audio- und Videokompression
On-Demand	zeitnahe Erfüllung von Anforderungen
Plugin	Computerprogramm, das eine Software ergänzt
Rich Internet Application	Anwendung, die Internettechniken nutzt und intuitive Benutzeroberfläche bietet

Richmedia	Inhalte, die optisch / akustisch (durch Video / Audio / Animation) angereichert wurden
Server	Hardware / Software, welche dem Client Dienste bereitstellt
SWF	abspielbare Adobe Flash Datei
Streaming	kontinuierliche Übertragung von Daten (Datenstrom)
Template	Vorlagen, die mit Inhalt gefüllt werden können
World Wide Web	über Internet abrufbares Hypertextsystem

Inhalt der CD

CD:\Source\Flash\	-	Quelltext Flash
CD:\Source\Silverlight\	-	Quelltext Silverlight
CD:\VBIBO\	-	Anwendung Virtuelle Bibliothek
CD:\Diplom.pdf	-	Diplomarbeit (PDF)

Selbständigkeitserklärung

Ich versichere hiermit, dass ich die Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel genutzt habe.

Peter Rauschenbach