21 Multimedia-Architekturen

Ansgar Scherp, Susanne Boll

Mit der steigenden Leistungsfähigkeit moderner Computer und der Unterstützung für eine qualitativ hochwertige Audio-, Video- und Bildwiedergabe Anfang der 90er Jahre hat Multimedia in zahlreichen Anwendungsbereichen Einzug gehalten. Für den Begriff Multimedia gibt es eine Vielzahl an Definitionen und Begriffsklärungen. Eine häufig zitierte Definition ist die von Steinmetz [Ste00], wonach eine Multimedia-Anwendung gekennzeichnet ist durch die rechnergesteuerte, integrierte Erzeugung, Manipulation (Interaktion des Benutzers mit den Medien), Darstellung/Präsentation, Speicherung und Kommunikation von unabhängigen Informationen, die in mindestens einem kontinuierlichen und einem diskreten Medium kodiert sind. Kontinuierliche Medienelemente sind Video, Audio und Animationen, während Bilder und Texte diskrete Medien darstellen. Darauf aufbauend kann nun der Begriff der Multimedia-Architektur definiert werden.

Definition (Multimedia-Architektur) *Eine Multimedia-Architektur ist eine Software-Architektur, die der Erzeugung, der Speicherung, der Transformation, der Präsentation und/oder dem Transport von multimedialen Daten dient.*

Die Definition zeigt bereits auf, dass es nicht *die* Multimedia-Architektur gibt, sondern vielmehr verschiedene Architekturen zur Unterstützung der verschiedenen Aspekte von Multimedia-Anwendungen. Im nachfolgenden Abschnitt 21.1 werden zunächst die typischen Aspekte von Multimedia-Anwendungen und -Systemen identifiziert; zu jedem der genannten Aspekte stellt der daran anschließende Abschnitt 21.2 Beispiele bestehender Multimedia-Architekturen und aktueller Forschungsarbeiten vor. Als zentrales Beispiel einer Multimedia-Architektur wird in diesem Kapitel das komponentenbasierte Framework MM4U (Abkürzung für »Multimedia for you«) zur dynamischen Generierung von personalisierten Multimedia-Inhalten ausführlich erläutert. Dazu diskutiert der Abschnitt 21.3 zunächst den Hintergrund zur Entwicklung des MM4U-Frameworks, bevor in Abschnitt 21.4 die Vorgehensweise zur Entwicklung des Frameworks beschrieben wird.

Diese basiert auf einer Modifikation des Hotspot-getriebenen Entwurfs objektorientierter Frameworks von Wolfang Pree. Abschließend wird in Abschnitt 21.5 die Entwicklung des MM4U-Frameworks im Detail beschrieben, bevor das Kapitel mit dem Fazit endet. Es ist darauf hinzuweisen, dass für die Abschnitte 21.4 und 21.5 der in Abschnitt 20.4 vorgestellte Hotspot-getriebene Entwurf objektorientierter Frameworks vorausgesetzt wird.

21.1 Aspekte von Multimedia

Betrachtet man das Spektrum von Multimedia-Anwendungen, so lassen sich folgende typische Aspekte identifizieren:

- Die Speicherung, Verwaltung und Bereitstellung von Medienelementen und Metadaten durch Medien- und Multimedia-Datenbanken.
- Die Übertragung von multimedialen Datenströmen in Netzwerken (inklusive Manipulation und Transformation dieser Datenströme).
- Die Darstellung und Wiedergabe von diskreten und kontinuierlichen Medienelementen und Multimedia-Präsentationen.
- Die Repräsentation von multimedialen Inhalten.

Eine Multimedia-Anwendung realisiert mindestens einen, oftmals jedoch mehrere dieser Multimedia-Aspekte. Für die Entwicklung von Multimedia-Anwendungen stehen neben Vorgehensmodellen und Entwicklungsmethoden für die Multimedia-Entwicklung, wie zum Beispiel [ESN03, DEM+99, RS99, Saw95, BBK+99], auch vorgefertigte Multimedia-Architekturen, wie zum Beispiel Frameworks, zur Verfügung.

Die Speicherung, Verwaltung und Bereitstellung von Medien und den assoziierten Metadaten kann mit einem Multimedia-Datenbankmanagementsystem erfolgen. Beispiele hierfür sind die aus einem Forschungskontext entstandenen Systeme METIS [RWP04] und QBIC [IBM04] sowie im kommerziellen Umfeld die *inter*Media-Erweiterung [Ora04] von Oracle 10*g*.

Daneben gibt es zahlreiche Multimedia-Architekturen, die die Übertragung und Manipulation von multimedialen Datenströmen in Netzwerken unterstützen. Dazu gehören beispielsweise die Network-Integrated Multimedia Middleware [NMM05], Presentation Processing Engine [PLV97, PVL96], DirectShow [Mic05], Helix DNA [Rea05] und QuickTime [App05]. Diese Ansätze bieten neben der Übertragung und Manipulation teilweise auch Funktionalitäten zur Aufnahme, Bearbeitung und Wiedergabe der Multimedia-Daten an.

Für die Darstellung und synchronisierte Wiedergabe von diskreten und kontinuierlichen Medienelementen existieren Multimedia-Frameworks wie beispielsweise der Forschungsansatz MET++ [Ack96], der ISO/IEC-Standard PREMO [DHM99] sowie das Java Media Framework [Sun05a] von Sun

Microsystems. Für die Darstellung von Multimedia-Präsentationen in den verschiedenen Formaten, wie zum Beispiel SMIL [W3C05], SVG [W3C04] und Macromedia Flash [Mac05], gibt es außerdem eine Vielzahl an entsprechenden Multimedia-Playern für die unterschiedlichsten Betriebssysteme und (mobilen) Endgeräte.

Neben den Multimedia-Aspekten der Speicherung und Übertragung hat die *Repräsentation* von multimedialen Inhalten eine große Bedeutung, da auf dieser Ebene der zeitliche Ablauf, das räumliche Layout und das Interaktionsverhalten von Multimedia-Präsentationen festgelegt wird. Beispiel einer Software-Architektur für die Repräsentation multimedialer Inhalte ist das Komponenten-Framework MM4U, das in den Abschnitten 21.3 bis 21.5 ausführlich vorgestellt wird. Neben der Repräsentation multimedialer Inhalte in einem internen Multimedia-Repräsentationsmodell unterstützt dieses Framework auch die dynamische Generierung von personalisierten Multimedia-Inhalten und adressiert damit einen immer bedeutsamer werdenden Erfolgsfaktor heutiger Multimedia-Anwendungen (vgl. [SN95], S. 764).

21.2 Beispiele für Multimedia-Architekturen

Die im vorangehenden Abschnitt 21.1 genannten Beispiele für Multimedia-Architekturen zur Speicherung, Übertragung und Präsentation von multimedialen Daten werden nun kurz vorgestellt, um damit einen Überblick über das Spektrum der verschiedenen Systeme und Architekturen zu geben. Im Anschluss an Medien- und Multimedia-Datenbanken in Abschnitt 21.2.1 werden in Abschnitt 21.2.2 die Streaming-Architekturen vorgestellt. In Abschnitt 21.2.3 werden schließlich die Beispiele zur Präsentation von Multimedia-Inhalten beschrieben.

21.2.1 Beispiele für (Multi-)Media-Datenbanken

QBIC Die Bilderdatenbank QBIC (Query By Image Content) von IBM war eine der ersten Medien-Datenbanken, die eine Suche nach Bildern auf der Basis der Bilddaten erlaubt [IBM04]. So unterstützt QBIC Anfragen bezüglich inhaltsbasierter Eigenschaften von Bildern, wie etwa nach Farbverteilungen und -layout sowie nach bestimmten Texturen in Bildern.

Oracle 10g interMedia Mit interMedia [Ora04] bietet Oracle eine Erweiterung der relationalen Oracle 10g-Datenbank an, mit der Medienelemente wie Bilder, Audios und Videos gespeichert, verwaltet und wieder abgerufen werden können. Oracle 10g interMedia unterstützt die Extraktion von

Metadaten zu den Medienelementen und erlaubt eine Abfrage des Datenbestandes über diese Metadaten oder über andere spezielle Indizes der Mediendaten [Ora05]. Sowohl für QBIC als auch für Oracle 10g interMedia ist anzumerken, dass es sich hier nicht um Multimedia-Datenbanken im strengen Sinne handelt, da QBIC nur einen Medientyp verwaltet und auch Oracle 10g interMedia die Speicherung von integrierten Multimedia-Dokumenten wie SMIL, SVG und Flash nicht direkt bietet.

METIS Die Multimedia-Datenbank METIS [RWP04, KPW04] stellt ein flexibles Konzept zur Definition und Verwaltung von beliebigen Medienelementen und deren Metadaten zu Verfügung. METIS kann durch Plug-ins an die Anforderungen konkreter Domänen und Anwendungen angepasst und erweitert werden. Dies wird unterstützt durch die Möglichkeit, domänenspezifische Medientypen definieren zu können. Dabei kann auch die semantische Zusammengehörigkeit von bestimmten Medienelementen und deren Metadaten zu neuen, eigenständigen Multimedia-Typen beschrieben werden. Die domänenspezifischen Medientypen können schließlich mittels so genannter *Semantic-Packs* gebündelt und weitergegeben werden. Dieses aus einem Forschungskontext entstandene System stellt einen der umfassendsten heute existierenden Ansätze für integriertes Multimedia-Management dar.

21.2.2 Beispiele für Streaming-Architekturen

Network-Integrated Multimedia Middleware Die komponentenbasierte Network-Integrated Multimedia Middleware (NMM) [NMM05] erlaubt die Übertragung von Audio- und Video-Datenströmen von stationären Systemen auf mobile Endgeräte. Dazu werden die Datenströme auf den stationären Systemen durch geeignete Transformations- und Transkodierungskomponenten an die Fähigkeiten des Endgerätes angepasst. Basierend auf der NMM wurde die so genannte *Multimedia-Box* erstellt, die eine einheitliche Schnittstelle zu verschiedenen Medienangeboten, wie zum Beispiel digitalem Fernsehen und Videorekorder, DVD- und CD-Player sowie Medien-Player, bietet.

DirectShow Die Multimedia-Schnittstelle DirectShow [Mic05] ist Bestandteil der umfangreichen Multimedia-Programmierbibliothek DirectX von Microsoft. Mit DirectShow können Audio- und Video-Datenströme auf der Microsoft-Windows-Plattform verarbeitet werden. DirectShow stellt dabei eine zusammenfassende Schnittstelle dar, die intern auf die Funktionalitäten anderer DirectX-Komponenten zugreift. So verwendet DirectShow zum Beispiel das API DirectSound zur Ausgabe und Aufnahme von Audio und DirectDraw für Videos. Die Schnittstelle von DirectShow umfasst neben der

Wiedergabe von Datenströmen auch deren Aufnahme und Manipulation. Entsprechend diesen drei Funktionalitäten sind in DirectShow drei Typen von so genannten Aufnahme-, Manipulation- und Wiedergabe-*Filter* definiert. Die konkreten Instanzen dieser Filter sind als COM-Komponenten realisiert [BN03].

Presentation Processing Engine Die Presentation Processing Engine (PPE) ist ein Komponenten-Framework zur Verarbeitung und Manipulation von Medienobjekten und -strömen, wie zum Beispiel für die (De-)Komprimierung, Drehung und Skalierung von Bildern und Video. Die PPE arbeitet nach dem Quelle-Komponente-Senke-Modell. Zwischen Quellen, wie Mikrophon und Kamera, und Senken, beispielsweise Lautsprecher und Bildschirm, können Komponenten zur Verarbeitung und Manipulation der medialen Daten eingefügt und zu einer Pipeline aufgereiht werden. Die Komponenten der PPE werden erst zur Laufzeit gebunden. Dadurch kann die Wahl der Komponenten für die Pipeline dynamisch (re-)konfiguriert werden, um sich ändernden QoS-Anforderungen (*Quality of Service*) anpassen zu können. Die QoS-Anforderungen können dabei entweder durch die Restriktionen eines verwendeten Endgerätes oder durch die Einstellungen des Benutzers vorgegeben werden [PLV97, PVL96].

Helix DNA Mit Helix DNA [Rea05] bietet RealNetworks eine Multimedia-Architektur zur Erzeugung, Übertragung und Wiedergabe von medialen Datenströmen im proprietären RealAudio- bzw. RealVideo-Format an. Die Helix-DNA-Plattform besteht aus drei Komponenten: Mit dem Helix DNA Producer werden zunächst die in einem Quellformat vorliegenden Audio- und Videodateien in das RealAudio- bzw. RealVideo-Format umgewandelt. Diese beiden Formate sind darauf ausgerichtet, als mediale Datenströme im Internet in Echtzeit übertragen zu werden. Mit dem Helix DNA Server werden die vorbereiteten Datenströme im Internet zur Verfügung gestellt. Schließlich dient der Helix DNA Client zur Darstellung der medialen Datenströme auf den (mobilen) Endgeräten.

QuickTime Die Multimedia-Architektur QuickTime [App05] von Apple stellt eine plattformübergreifende Multimedia-Architektur für die Aufnahme, Wiedergabe und Bearbeitung von Multimedia-Daten zur Verfügung. Die QuickTime-Technologie umfasst ein API, ein Anwendungsframework für Java und ein Dateiformat. Wesentlicher Bestandteil von QuickTime sind die so genannten *Manager*, die für verschiedene Aufgaben wie die Aufzeichnung, Bearbeitung und synchronisierte Wiedergabe von *QuickTime Movies* zur Verfügung stehen. Anwendungen, die auf die QuickTime-Architektur aufbauen, sind Basisanwendungen wie beispielsweise der QuickTime

Player, der QuickTime Broadcaster oder der QuickTime Streaming Server, aber auch Multimedia-Autorensysteme und Audio-/Videoprogramme.

21.2.3 Beispiele für Präsentationsarchitekturen

Java Media Framework Das Java Media Framework (JMF) [Sun05a] von Sun Microsystems ist wohl das am weitesten verbreitete Multimedia-Präsentationsframework für die Programmiersprache Java. Das JMF unterstützt die Aufnahme, Wiedergabe, Übertragung und Transkodierung von verschiedenen Audio- und Videoformaten. Dazu werden im JMF entsprechende Module definiert, die nach dem Quelle-Verarbeitung-Senke-Modell komponiert werden können (vgl. Presentation Processing Engine in Abschnitt 21.2.2). Das JMF kann sowohl in normalen Standalone-Anwendungen als auch für Java-Applets in einem Web-Browser eingesetzt werden [Sun05b].

MET++ Eines der bekanntesten Frameworks für grafische Benutzungsoberflächen ist das im Rahmen der Dissertation von Erich Gamma entstandene objektorientierte Framework ET++ [Gam92]. Dieses Framework stellt allgemeine Elemente zur Entwicklung grafischer Benutzungsoberflächen zur Verfügung. Das MET++-Framework [Ack96] ist eine Erweiterung des ET++-Frameworks um Funktionalitäten zur Darstellung und Wiedergabe von diskreten und kontinuierlichen Medienelementen sowie zur Darstellung und Animation von 2D- und 3D-Objekten [Ack96]. Zudem bietet MET++ auch Funktionalitäten zur Synchronisierung der Medienelemente.

PREMO Der mehrteilige ISO/IEC-Standard PREMO (PRogramming Environment for Multimedia Objects) [DHM99] definiert eine abstrakte Architektur zur Aufnahme, Übertragung und Wiedergabe von medialen Datenströmen. Durch so genannte *Wrapper* können zum Beispiel bestehende Ansätze und Systeme zur Wiedergabe von medialen Datenströmen gekapselt werden. PREMO definiert damit eine Art Middleware zur transparenten Verknüpfung verschiedener konkreter Ansätze und Systeme für die Aufnahme, Bearbeitung und Wiedergabe von medialen Datenströmen, die miteinander verknüpft und gemeinsam genutzt werden können.

21.3 Hintergrund zur Entwicklung von MM4U

Mit MM4U wird ein komponentenbasiertes objektorientiertes Framework vorgestellt, das neben der Repräsentation multimedialer Inhalte in einem internen Multimedia-Repräsentationsmodell auch eine umfangreiche Unterstützung für die typischen Aufgaben zur dynamischen Generierung

von personalisierten Multimedia-Präsentationen bietet. Betrachtete Anwendungsbereiche sind beispielsweise multimediale Informationsdienste und personalisierte mobile Multimedia-Anwendungen. Das MM4U-Framework erlaubt die Personalisierung von multimedialen Inhalten hinsichtlich unterschiedlichster Benutzerkontexte, die sich aus den Präferenzen und Interessen des Benutzers, seiner aktuellen Position und Umgebung sowie des verwendeten (mobilen) Endgerätes ergeben. Eine *manuelle* Erstellung solcher personalisierter Multimedia-Präsentationen ist aufgrund der Vielzahl an unterschiedlichen Benutzerkontexten nicht durchführbar [AMR96, SB05b]. Daher wird stattdessen eine dynamische Generierung von personalisierten Multimedia-Inhalten benötigt, um dem jeweiligen Kontext und der Situation der Benutzer bestmöglich entsprechen zu können.

Mit dem MM4U-Framework wurde ein Framework entwickelt, das insbesondere die Aspekte der Repräsentation und der Präsentation von Multimedia-Inhalten adressiert und die Aspekte Verwaltung und Übertragung berührt. Die nachfolgenden Abschnitte beschreiben die Erfahrungen bei der Entwicklung des Frameworks, seine grundlegende Organisation, dargestellt durch dessen Komponenten, deren Beziehungen zueinander und zur Umgebung, sowie die Vorgehensweise beim Entwurf und der Evolution des Frameworks. Ziel des MM4U-Frameworks ist, den Software-Entwicklungsprozess personalisierter multimedialer Anwendungen zu vereinfachen und eine effiziente Entwicklung solcher Anwendungen zu ermöglichen. Das MM4U-Framework löst typische, immer wiederkehrende Aufgaben des Multimedia-Personalisierungsprozesses auf abstrakter Ebene und kapselt diese in einzelne Komponenten. Die folgende Abbildung 21.1 illustriert den allgemeinen Multimedia-Personalisierungsprozess und die einzelnen Aufgaben, die zur dynamischen Generierung personalisierter Multimedia-Inhalte durchzuführen sind. In einem ersten Schritt werden die Medienelemente anhand ihrer Metadaten ausgewählt (engl. *select*), die den Interessen und Präferenzen des Benutzers sowie den Eigenschaften des Endgerätes bestmöglich entsprechen. Im nächsten Schritt werden die ausgewählten Medienelemente in Raum und Zeit arrangiert und zu dem personalisierten multimedialen Inhalt zusammengefügt (engl. *assemble*). Dies geschieht mit Hilfe eines internen Multimedia-Repräsentationsmodells, das von der Syntax und den Fähigkeiten existierender (standardisierter) Multimedia-Präsentationsformate, wie zum Beispiel SMIL, SVG und Macromedia Flash, abstrahiert [SB05b]. Dieses Repräsentationsmodell ist so entworfen, dass es zum einen die Multimedia-Komposition umfassend beschreibt und gleichzeitig einfach in die konkrete Syntax der verschiedenen Präsentationsformate transformiert (engl. *transform*) werden kann. Dies geschieht im nächsten Schritt, der Transformationsphase. Hier werden die multimedialen Inhalte im internen Repräsentationsmodell in die Syntax und Fähigkeiten der konkreten Multimedia-Präsentationsformate

umgewandelt [SB05c]. Schließlich werden die personalisierten Multimedia-Präsentationen auf das (mobile) Endgerät des Benutzers übertragen und von geeigneten Multimedia-Playern wiedergegeben (engl. *present*).

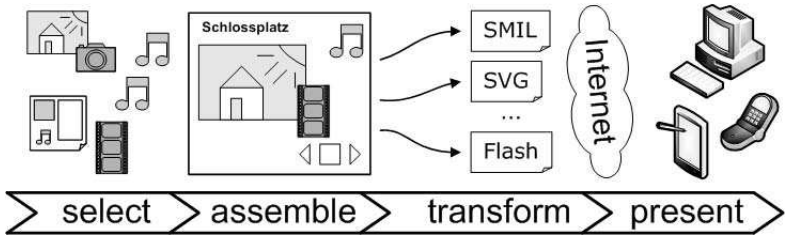

Abbildung 21.1: Der allgemeine Multimedia-Personalisierungsprozess [SB05b, SB05c]

21.4 Vorgehensweise bei der Entwicklung von MM4U

Die Entwicklung des MM4U-Frameworks sollte durch ein Vorgehensmodell und eine geeignete Entwicklungsmethodik für Frameworks unterstützt werden. Wir haben uns zunächst für den Hotspot-getriebenen Ansatz nach Pree entschieden (siehe Abschnitt 20.4.1), da dieser eine ausgereifte und erprobte Vorgehensweise zur Entwicklung objektorientierter Frameworks darstellt. Während der Anwendung dieses Ansatzes wurden jedoch schnell zwei Nachteile deutlich [SB05a]:

- Zum einen wurde eine explizite Unterstützung zur Identifikation von Komponenten und zur Spezifikation von Flexibilitätsanforderungen an diese Komponenten vermisst: Zwar werden zu Beginn des Hotspot-getriebenen Ansatzes aCiRC-Karten erstellt und einzelne Cluster auf Basis von Schlüsselabstraktionen gebildet, die zum Auffinden von Komponenten helfen könnten, jedoch werden diese im weiteren Entwicklungsprozess des Frameworks nicht wieder berücksichtigt.
- Zum anderen ist die Aussage, dass ein Hotspot *in etwa einer Methode* entspricht (siehe [Pre97, Pre95]) für das Komponenten-Framework MM4U zu ungenau: Es ist damit nicht klar, wie viel Funktionalität tatsächlich zu einem Hotspot gehört. Daher wurde die Granularität von Hotspots auf *genau eine Einschubmethode* festgelegt. Diese Entscheidung wird unterstützt durch die Einführung so genannter *Gruppen-Hotspot-Karten*, die die Bildung von logisch zusammengehören-

den Gruppen von Hotspots erlauben. Diese Gruppen-Hotspot-Karten werden zur Identifikation der Komponenten des Frameworks und zur Spezifikation der Flexibilitätsanforderungen an diese Komponenten verwendet.

Konsequenterweise wurde eine Modifikation am hotspot-getriebenen Ansatz vorgenommen, die in Abbildung 21.2 dargestellt ist. Die einzelnen Aktivitäten dieses Entwicklungsprozesses werden im Folgenden beschrieben [SB05a]:

Definition eines speziellen Objektmodells und Identifikation der Hotspots Die komponentenbasierte Software-Entwicklung beginnt typischerweise mit der Definition der Komponenten. Der in Abbildung 21.2 dargestellte Entwicklungsprozess für komponentenbasierte Frameworks beginnt jedoch wie bei Pree mit der Erstellung eines initialen Objektmodells. Wir beginnen mit einem Objektmodell, da die Funktionalität des Komponenten-Frameworks durch schrittweise Abstraktion von diesem Objektmodell entwickelt wird. Nach der Definition des initialen Objektmodells beginnt der Hauptzyklus des Entwicklungsprozesses mit der Identifikation erster Hotspots auf der Granularitätsebene einzelner Methoden und der Erstellung entsprechender Hotspot-Karten.

Gruppierung der Hotspot-Karten und Identifikation der Komponenten Die identifizierten Hotspot-Karten werden in logische Gruppen eingeteilt. Dazu werden die Flexibilitätsanforderungen, die auf den Hotspot-Karten niedergeschrieben sind, miteinander verglichen. Hotspot-Karten, deren Flexibilitätsanforderungen ein gemeinsames Problem beschreiben, werden zu einer logischen Gruppen arrangiert. Für jede logische Gruppe wird schließlich eine entsprechende Gruppen-Hotspot-Karte erstellt. Es erscheint sinnvoll zu sein, jede dieser Gruppen-Hotspot-Karten als eigenständige Framework-Komponente zu realisieren. Die Flexibilitätsanforderungen an die Komponenten werden also durch die Gruppen-Hotspot-Karten definiert. Dadurch wird die Entwicklung von flexiblen Instanzen der Framework-Komponenten unterstützt. Wie in Abschnitt 21.5 zu sehen sein wird, definiert jede der identifizierten Gruppen-Hotspot-Karten eine Komponente im MM4U-Framework und entspricht jeweils einer Aufgabe im allgemeinen Multimedia-Personalisierungsprozess.

Die Gruppierung von Hotspot-Karten zu logisch zusammengehörenden Gruppen stellt ein Mittel zur Abstraktion der Flexibilitätsanforderungen an das Framework in Situationen dar, in denen Details nicht wichtig sind. Dies ist zum Beispiel dann der Fall, wenn nicht jeder einzelne Hotspot mit den Anwendungsexperten diskutiert werden muss. Durch die Bildung logischer Gruppen von Hotspots und deren Festhalten auf einer Gruppen-Hotspot-

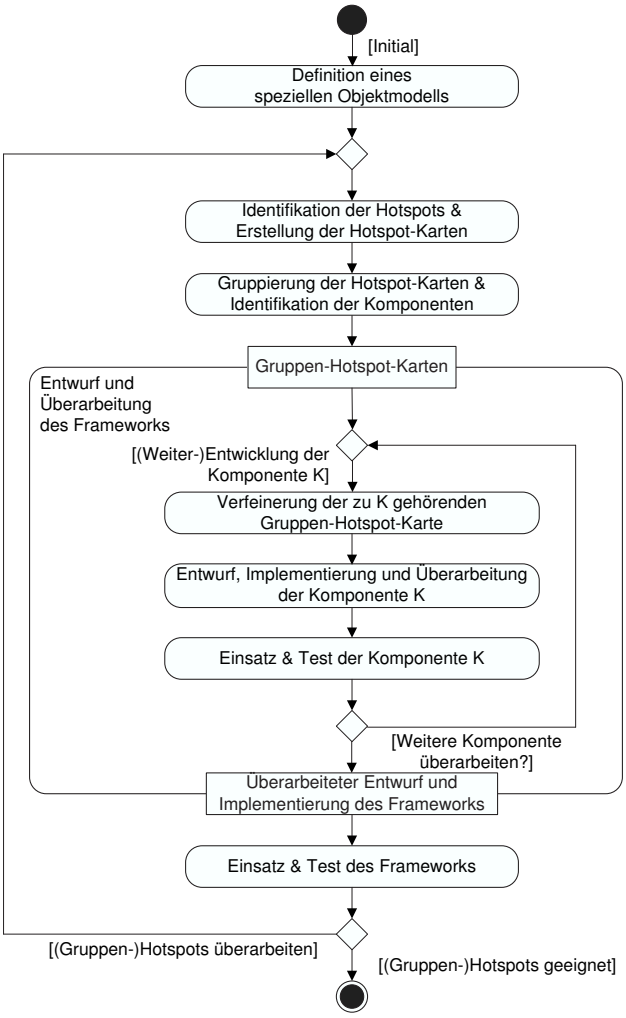

Abbildung 21.2: Der modifizierte Hotspot-getriebene Framework-Entwicklungsprozess

Karte erhält man automatisch einen gröberen und damit übersichtlicheren Blick auf die Flexibilitätsanforderungen an das zu entwickelnde Framework. Die Hotspot-Karten können dabei zum Beispiel mit Hilfe einer Büroklammer hinter der Gruppen-Hotspot-Karte geheftet werden. Schließlich können aus Gruppen-Hotspot-Karten wiederum Gruppen gebildet werden, wodurch sich ein noch abstrakteres Bild auf die Flexibilitätsanforderungen an das zu entwickelnde objektorientierte Framework ergibt. Konsequen-

terweise führt die Verschachtelung von Gruppen-Hotspot-Karten zur einer Hierarchie von Framework-Komponenten.

Die Struktur der Gruppen-Hotspot-Karten ist ähnlich der Struktur der Hotspot-Karten (siehe Abschnitt 20.4.3). Sie haben einen Namen und eine einzeilige Beschreibung der Funktionalität, die flexibel gehalten werden soll, eine Beschreibung der Funktionalität in mindestens zwei verschiedenen konkreten Anwendungssituationen sowie die Angabe, ob eine Anpassung der Funktionalität zur Laufzeit und eine Werkzeugunterstützung für den Endanwender notwendig ist. Optional können auch noch die angehefteten Hotspot-Karten aufgelistet werden.

Entwurf, Implementierung und Überarbeitung des Frameworks Die Aktivität zu Entwurf, Implementierung und Überarbeitung des Frameworks stellt die größte Modifikation des Pree-Ansatzes dar. Es werden für jede Komponente, die in dieser Iteration überarbeitet und verbessert werden soll, folgende Schritte durchgeführt: Zunächst werden weitere Hotspots zu der Gruppen-Hotspot-Karte der jeweiligen Komponente identifiziert, d. h. die Gruppen-Hotspot-Karte wird verfeinert. Anschließend wird die Komponente entworfen und implementiert. Dazu werden die Schnittstellen der Komponente definiert und zumindest eine konkrete Instanz der Komponente implementiert und getestet. Da nicht jede Komponente in jeder Iteration weiterentwickelt werden muss, können die Komponenten unterschiedliche Reife haben. Auch kann die Entwicklung der Komponenten parallel durchgeführt werden (im Diagramm nicht dargestellt). Ergebnis ist ein überarbeiteter Entwurf und eine verbesserte Implementierung der Framework-Komponenten und des Frameworks.

Einsatz und Test des Frameworks Das Framework wird anschließend hinsichtlich der Qualität des Entwurfs getestet. Die einzige Möglichkeit, die Fehler und Schwachstellen zu entdecken, besteht darin konkrete Anwendungen zu entwickeln, die das Framework nutzen [BMM+99]. Das Entwickeln einer konkreten Anwendung bedeutet im Wesentlichen die Komposition von Framework-Komponenten (die die Anforderungen der betrachteten Anwendungsdomäne erfüllen) hinsichtlich der Regeln und Vorgaben des Frameworks. Die Instanzen der Framework-Komponenten können dabei Neuentwicklungen, aber auch beispielsweise Anpassungen bereits existierender Implementierungen oder die Wiederverwendung von Komponenten in unterschiedlichen Konfigurationen sein. Fehler am Framework werden sofort behoben und Schwachstellen im Entwurf werden im nächsten Durchlauf des Hauptzyklus verbessert.

21.5 Entwicklung des MM4U-Frameworks

Der modifizierte Hotspot-getriebene Entwicklungsprozess wurde zur Entwicklung des Komponenten-Frameworks MM4U für personalisierte Multimedia-Anwendungen eingesetzt. Die Entwicklung dieses Frameworks wird in den folgenden Abschnitten 21.5.1 bis 21.5.6 beschrieben.

21.5.1 Analyse verwandter Arbeiten und spezielle Objektmodelle

Hinsichtlich der verschiedenen Aufgaben des allgemeinen Multimedia-Personalisierungsprozesses (siehe Abbildung 21.1) wurde zunächst eine umfangreiche Analyse verwandter Arbeiten in den Bereichen der Benutzermodellierung, Modellierung von Metadaten für Multimedia-Inhalte, Multimedia-Komposition sowie Multimedia-Präsentation durchgeführt (siehe [SB05b]). Des Weiteren wurden existierende Systeme und Ansätze zur Erzeugung von personalisierten Multimedia-Inhalten analysiert, wie zum Beispiel die Cuypers Engine [van04] und das Standardreferenzmodell für intelligente Multimedia-Präsentationssysteme [BFF+97]. Darüber hinaus wurden auch Anforderungen an den Entwurf des Frameworks aus ersten Prototypen von personalisierten Multimedia-Anwendungen extrahiert, die wir für verschiedene Bereiche entwickelt haben. Diese Prototypen sind eine personalisierte Stadtführung durch Wien [Bol03] und Oldenburg [BK03], eine personalisierte GPS-gestützte Schnitzeljagd auf dem Campusgelände der Universität Oldenburg [BKW03] sowie ein personalisierter Musiknewsletter [Ric03]. Die Objektmodelle dieser personalisierten Multimedia-Anwendungen wurden hinsichtlich ihrer Ähnlichkeiten und Gemeinsamkeiten untersucht. Auf Basis der ausführlichen Analyse verwandter Arbeiten und den Erfahrungen mit den Prototypen und deren Objektmodelle wurde ein initiales Objektmodell des MM4U-Frameworks erstellt.

21.5.2 Identifizierung und Erstellung der Hotspot-Karten und Gruppen-Hotspot-Karten

Für die Entwicklung des MM4U-Frameworks wurde gemäß dem Vorgehensmodell in Abbildung 21.2 zunächst eine initiale Menge an Hotspots identifiziert und entsprechende Hotspot-Karten erstellt. Diese Hotspot-Karten wurden in logische Gruppen angeordnet und entsprechende Gruppen-Hotspot-Karten wurden erstellt. Jede Gruppen-Hotspot-Karte behandelt eine Aufgabe des allgemeinen Multimedia-Personalisierungsprozesses (siehe Abbildung 21.1). Zum Auffinden der Hotspots halfen die eigenen Erfahrungen bei der Entwicklung von Prototypen für personalisierte Multimedia-Anwendungen sowie die verwandten Arbeiten zur Generierung

von personalisierten Multimedia-Inhalten. Anschließend wurden in dem iterativen Prozess weitere Hotspot-Karten identifiziert und den Gruppen-Hotspot-Karten zugeordnet. Für das MM4U-Framework wurden insgesamt fünf Gruppen-Hotspot-Karten ermittelt:

1. Unterstützung und Integration unterschiedlicher existierender Systeme und Lösungen zur Speicherung, Verarbeitung, Suche und Zugriff auf Profilinformationen über den Benutzer.
2. Unterstützung und Integration unterschiedlicher existierender Systeme und Lösungen zur Speicherung, Verarbeitung, Suche und Zugriff auf Medien und assoziierten Metadaten.
3. Komposition der personalisierten Multimedia-Inhalte mit Hilfe geeigneter Kompositionselemente in einem internen Multimedia-Repräsentationsmodell. Dieses Repräsentationsmodell muss hinsichtlich anwendungsspezifischer Kompositionsaufgaben erweitert werden können.
4. Transformation der personalisierten Multimedia-Inhalte im internen Repräsentationsmodell in die Syntax und Fähigkeiten unterschiedlicher Multimedia-Präsentationsformate, wie zum Beispiel SMIL, SVG und Flash.
5. Wiedergabe der personalisierten Multimedia-Präsentation im konkreten Zielformat auf dem (mobilen) Endgerät des Benutzers.

21.5.3 Identifikation der Komponenten und Entwurf der Framework-Architektur

Zu Beginn der Entwicklung des MM4U-Frameworks gab es bereits eine gewisse Vorstellung davon, welche Flexibilitätsanforderungen an das Framework zu stellen sind. Es war allerdings nicht klar, wie diese Flexibilitätsanforderungen sinnvoll in Komponenten aufzuteilen sind und wie viele Komponenten notwendig sein würden, um die Anforderungen umzusetzen. Die ermittelten Gruppen-Hotspot-Karten teilen die Flexibilitätsanforderungen an das MM4U-Framework in fünf logische Gruppen auf und adressieren jeweils eine spezielle Aufgabe im allgemeinen Multimedia-Personalisierungsprozess (siehe Abbildung 21.1). Für jede dieser Gruppen-Hotspot-Karten wurde eine (austauschbare) Komponente im MM4U-Framework definiert. Dazu wurden die Schnittstellen der Framework-Komponenten festgelegt und eine konkrete Instanz der Komponenten implementiert. Die konkrete Implementierung der einzelnen Framework-Komponenten wurde jeweils mit Hilfe eines objektorientierten Frameworks realisiert.

Erst mit der Identifizierung der Gruppen-Hotspot-Karten war sicher, dass alle benötigten Komponenten identifiziert wurden und dass die ermit-

telten Komponenten auch die richtigen sind. Das bedeutet, dass die Flexibilitätsanforderungen an das Framework sinnvoll in (die identifizierten) Komponenten aufgeteilt wurden. Die Architektur des MM4U-Frameworks ist in Abbildung 21.3 dargestellt. Entsprechend den Gruppen-Hotspot-Karten besteht das MM4U-Framework aus fünf Komponenten (gekennzeichnet durch die Ziffern 1 bis 5): (1) Die Komponente für den Zugriff auf die Profilinformationen über den Benutzer, (2) die Komponente für den Zugriff auf die Medienelemente und die assoziierten Metadaten, (3) die Komposition der multimedialen Inhalte im internen Repräsentationsmodell, (4) die Transformation in die konkreten Präsentationsformate und (5) die Darstellung der erzeugten Multimedia-Präsentationen auf den (mobilen) Endgeräten. In den folgenden beiden Abschnitten 21.5.4 und 21.5.5 wird beispielhaft der Entwurf der Komponente zur Multimedia-Komposition und der Präsentationsformat-Generatoren vorgestellt.

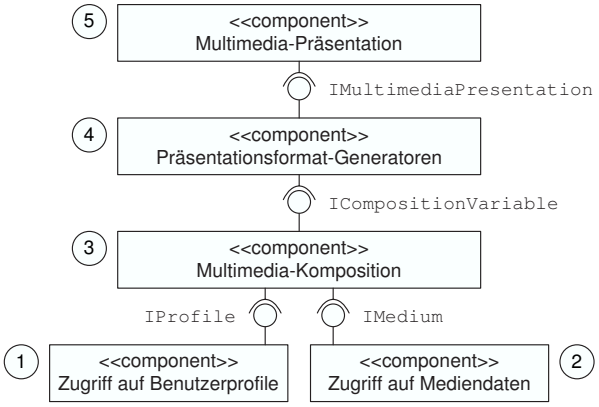

Abbildung 21.3: UML-Komponentendiagramm (Abschnitt 3.2) des MM4U-Frameworks

21.5.4 Entwurf der Multimedia-Kompositionskomponente

Aufgabe der Multimedia-Kompositionskomponente ist die Erstellung beliebiger personalisierter Multimedia-Inhalte mit Hilfe geeigneter Kompositionselemente in einem internen Multimedia-Repräsentationsmodell. Dieses Multimedia-Repräsentationsmodell abstrahiert von der Syntax und den Fähigkeiten aktueller Multimedia-Präsentationsformate. Zur Erstellung der personalisierten Multimedia-Inhalte bietet die Kompositionskomponente eine Menge von anwendungsunabhängigen Basiskompositionselementen an. Darüber hinaus ist die Multimedia-Kompositionskomponente

hinsichtlich komplexer und anwendungsspezifischer Multimedia-Kompositions- und Multimedia-Personalisierungsaufgaben anpassbar und erweiterbar [SB05b]. Die Basiskompositionselemente lassen sich in die folgenden drei Typen aufteilen: Medienelemente, Basisoperatoren und Projektoren. Diese werden im Folgenden vorgestellt.

Medienelemente Das MM4U-Framework unterstützt sowohl diskrete als auch kontinuierliche Medienelemente: Die diskreten Medienelemente, also Texte und Bilder, werden durch die Framework-Klassen `Text` und `Image` realisiert. Hinsichtlich der kontinuierlichen Medienelemente stehen die Framework-Klassen `Audio` und `Video` zur Verfügung. Die Verarbeitung der Medienelemente im MM4U-Framework erfolgt nach dem Proxy-Muster [GHJV04], d. h. die konkreten Medienobjekte verwalten lediglich die Metadaten zu den Mediendaten. Nur bei Bedarf werden die tatsächlichen Mediendaten in das Framework geladen.

Basisoperatoren Zur zeitlichen Anordnung der Medienelemente stehen zeitliche Kompositionsoperatoren, wie zum Beispiel `Parallel` und `Sequential`, zur Verfügung. Der `Parallel`-Operator wird verwendet, um mehrere Medienelemente oder Teilpräsentationen zur gleichen Zeit darzustellen. Mit dem Kompositionsoperator `Sequential` können Medienelemente oder Teilpräsentationen nacheinander präsentiert werden. Abbildung 21.4 zeigt das Beispiel einer einfachen Slideshow, bei der vier Bilder nacheinander dargestellt werden. Parallel zur Slideshow wird ein Audiomedium abgespielt.

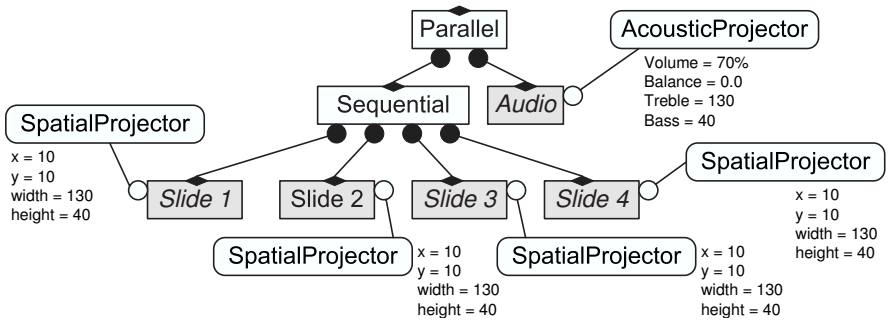

Abbildung 21.4: Beispiel einer Präsentation im internen Multimedia-Repräsentationsmodell

Projektoren Mit Hilfe von so genannten Projektoren wird das Layout der Multimedia-Präsentation realisiert. Wie in Abbildung 21.4 dargestellt, können Projektoren an Operatoren und Medien angeheftet werden. Zur Anordnung visueller Medienelemente im Raum stehen räumliche Projektoren zur Verfügung. Die entsprechende Klasse in der Multimedia-Kompositionskomponente heißt `SpatialProjector`. Ein `SpatialProjector` bestimmt neben der Position im Raum auch die Breite und Höhe eines visuellen Mediums. Neben den Projektoren für das räumliche Layout gibt es auch Projektoren für das akustische Layout einer Multimedia-Präsentation. So können mit Hilfe eines `AcousticProjector` die Lautstärke, Höhen, Bass und Balance eines Audio- oder Videomediums bestimmt werden.

Ein Anwendungsentwickler verwendet die Basiskompositionselemente als *Black Box* und instanziiert beliebig viele Objekte der entsprechenden Klassen. Dabei erstellt der Konstruktor der Basiskompositionselemente alle notwendigen internen Objekte für die Multimedia-Komposition. Die hier genannten Basiskompositionsoperatoren und Projektoren sind nur einige Beispiele aus dem internen Multimedia-Repräsentationsmodell des Frameworks. Eine ausführliche Beschreibung ist in [SB05b] zu finden.

Bezüglich der Verfeinerung der Gruppen-Hotspot-Karte der Multimedia-Kompositionskomponente sind die Basiskompositionselemente nicht weiter von Interesse, da sie atomare Einheiten zur Multimedia-Komposition darstellen. Eine Anforderung an die Multimedia-Kompositionskomponente ist aber auch die Erweiterbarkeit des internen Repräsentationsmodells um komplexere und anwendungsspezifische Funktionalitäten zur Multimedia-Komposition und -Personalisierung. Dazu steht die abstrakte Kompositionsklasse `ComplexOperator` bzw. die Schnittstelle `IComplexOperator` zur Verfügung. Durch Implementierung konkreter Unterklassen der `ComplexOperator`-Klasse kann komplexe und anwendungsspezifische Multimedia-Personalisierungsfunktionalität in das Framework integriert werden. Alternativ kann auch die Schnittstelle `IComplexOperator` verwendet werden. Die abstrakte `ComplexOperator`-Klasse implementiert jedoch bereits einige Standardfunktionen der `IComplexOperator`-Schnittstelle und ist damit für die Anwendungsentwickler einfacher zu verwenden.

Wie bei den Basiskompositionselementen werden bei den komplexen Operatoren die multimedialen Inhalte durch Erzeugen von Objekten der entsprechenden komplexen Operator-Klassen generiert, d. h. durch Aufrufen des Konstruktors der konkreten Implementierung von `(I)ComplexOperator`. Als Hotspot war zunächst die Methode `doCompose(...)` in der Klasse `ComplexOperator` vorgesehen. Allerdings wurde diese in den Konstruktor überführt, da sich der Konstruktor flexibler als die Einschubmethode `doCompose(...)` erwies. Während die Signatur der `doCompose(...)`-Methode in den konkreten Implementierungen nicht mehr verändert oder erweitert werden kann, bleiben bei einer Lösung über den Konstruktor

alle Möglichkeiten zur Nutzung zusätzlicher Ressourcen und Parameter offen, da der Konstruktor in den konkreten Implementierungen jeweils eine andere Signatur haben kann. Im Folgenden wird das Konzept des komplexen Operators sowie dessen Erweiterung zum dynamischen Operator vorgestellt.

Komplexe und dynamische Kompositionsoperatoren Ein *komplexer Kompositionsoperator* kapselt die Kompositionsfunktionalität beliebiger Basiskompositionselemente. Dadurch können Anwendungsentwickler komplexere Bausteine zur Multimedia-Komposition erstellen. Ein komplexer Operator umfasst eine beliebige Anzahl an einfachen Operatoren, Projektoren und Medienelementen. Neben diesen Basiskompositionselementen kann ein komplexer Operator auch weitere komplexe Operatoren beinhalten. Damit stellt das Konzept der komplexen Operatoren zugleich auch ein Mittel zur Wiederverwendung von Teilpräsentationen dar. Kapselt ein komplexer Operator nicht alle Bestandteile einer Teilpräsentation, sondern lässt einige Kompositionselemente offen, so sprechen wir von einem *parametrisierten komplexen Operator*. Der `Slideshow`-Operator in Abbildung 21.5 ist ein Beispiel eines parametrisierten komplexen Operators. Erst durch die Angabe der konkreten Medienelemente für die einzelnen *Slides* kann der komplexe `Slideshow`-Operator initialisiert und verwendet werden.

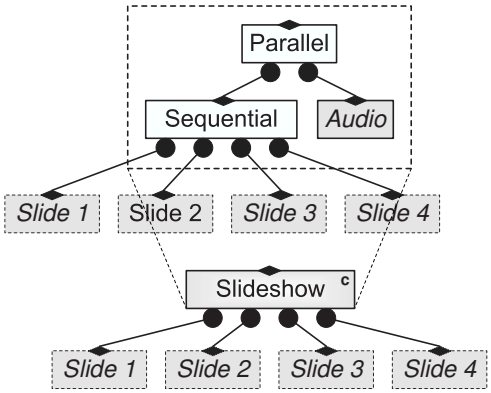

Abbildung 21.5: Beispiel eines parametrisierten komplexen Kompositionsoperators

Auch wenn einige Elemente eines komplexen Operators parametrisiert werden können, so beschreiben komplexe Operatoren immer eine statische Dokumentenstruktur. Die Struktur der erzeugten Multimedia-Präsentation

ist also im komplexen Operator fest verdrahtet. *Dynamische Kompositionsoperatoren* erlauben darüber hinaus die Bestimmung der Struktur und des Layouts der personalisierten Multimedia-Inhalte zur Laufzeit. Dies wird dadurch erreicht, dass komplexe Operatoren durch zusätzlichen anwendungsspezifischen Programmcode erweitert oder dass andere Ansätze zur Multimedia-Personalisierung integriert werden, wie zum Beispiel Stylesheets oder Layoutregeln. Diese zusätzliche Anwendungslogik erlaubt den dynamischen Kompositionsoperatoren zum Beispiel, den konkreten Präsentationsablauf oder die Anzahl der verwendeten Medienelemente für eine Multimedia-Präsentation erst zur Laufzeit zu ermitteln.

Trotz der unterschiedlichen Kompositionseigenschaften komplexer und dynamischer Operatoren werden beide durch konkrete Implementierungen von `(I)ComplexOperator` realisiert. Das liegt daran, dass es für die erzeugten (Teil-)Präsentationen im internen Multimedia-Repräsentationsmodell unerheblich ist, ob die Struktur der personalisierten Multimedia-Präsentation im komplexen Kompositionsoperator vorgegeben und fest verdrahtet ist oder diese Struktur erst zur Laufzeit und in Abhängigkeit von den Benutzerprofilinformationen durch einen dynamischen Operator bestimmt wird. Dabei ist es auch unerheblich, welche Benutzerprofilinformationen oder anderen zusätzlichen Parameter und Datenquellen zur Erfüllung dieser Kompositions- und Personalisierungsaufgabe verwendet werden. Damit das Framework auf die von einem komplexen oder dynamischen Operator erzeugte (Teil-)Präsentation zugreifen kann, stellt die Schnittstelle `IComplexOperator` die Methode `getRootOperator()` zur Verfügung, die auf das Wurzelelement der erzeugten Teilpräsentation verweist. Über dieses Wurzelelement kann das MM4U-Framework rekursiv auf alle Bestandteile des Multimedia-Repräsentationsbaumes zugreifen. Diese Methode ist für die Transformation des internen Repräsentationsmodells in die unterschiedlichen konkreten Präsentationsformate von Bedeutung, die im folgenden Abschnitt beschrieben wird.

21.5.5 Entwurf der Präsentationsformat-Generatorenkomponente

Die personalisierten Multimedia-Inhalte im internen Repräsentationsmodell werden von der Präsentationsformat-Generatorenkomponente mit Hilfe eines anwendungsunabhängigen Transformationsalgorithmus in die konkrete Syntax und Fähigkeiten der Zielformate transformiert. Ausgehend vom Wurzelelement des Multimedia-Repräsentationsbaums traversiert der Algorithmus den Baum und transformiert die personalisierten Multimedia-Inhalte in die konkreten Ausgabenformate [SB05c]. Für den Entwurf der Präsentationsformat-Generatorenkomponente wurde die entsprechende Gruppen-Hotspot-Karte verfeinert und eine Reihe von Hotspots auf der

Methodenebene identifiziert. Jeder dieser Hotspots hat dabei die Umwandlung eines der in Abschnitt 21.5.4 eingeführten Basiskompositionselemente zur Aufgabe: Die Transformation der Repräsentation der Medienelemente in die Syntax des Zielformates erfolgt über die Einschubmethode `medium(...)`. Für die Operatoren stehen die Einschubmethoden `parallel(...)` und `sequential(...)` zur Verfügung. Hinsichtlich der Projektoren sind die Einschubmethoden `spatialProjector(...)` und `acousticProjector(...)` identifiziert und in das Framework integriert worden.

Wie in Abbildung 21.6 zu sehen ist, sind die Einschubmethoden der Präsentationsformat-Generatorenkomponente auf die beiden Einschubklassen `AbstractContentGenerator` und `AbstractLayoutGenerator` verteilt worden. Die `AbstractContentGenerator`-Klasse umfasst die Einschubmethoden zur Transformation der Operatoren und der Medien. Die `AbstractLayoutGenerator`-Klasse dagegen realisiert den Transformationsschritt zur Definition des Layouts der Multimedia-Präsentation und beinhaltet die Einschubmethoden zur Transformation der Projektoren. Dies ist allerdings nur für Multimedia-Präsentationsformate von Bedeutung, die das Layout getrennt vom eigentlichen multimedialen Inhalt, beispielsweise in einem expliziten Kopfbereich (engl. *header*), definieren, wie z. B. SMIL.

Zur Transformation der komplexen und dynamischen Kompositionsoperatoren in die konkreten Multimedia-Präsentationsformate sind keine weiteren Hotspots notwendig. Vielmehr werden diese aus den Basiselementen zusammengesetzten Kompositionsoperatoren vom abstrakten Transformationsalgorithmus automatisch in ihre kleineren Bestandteile, also die Operatoren, Projektoren und Medien, aufgebrochen und mit Hilfe der Einschubmethoden für die Basiskompositionselemente in das Zielformat transformiert. Dazu wird die `getRootOperator()`-Methode des komplexen bzw. dynamischen Kompositionsoperators aufgerufen, um das Wurzelelement der von diesem Operator erzeugten Präsentation zu erhalten. Über das Wurzelelement traversiert der Transformationsalgorithmus schließlich auch den von einem komplexen oder dynamischen Operator generierten Multimedia-Repräsentationsbaum und wandelt ihn in das Zielformat um.

Der Transformationsalgorithmus für multimediale Inhalte ist für beliebige Anwendungsbereiche anwendbar. Es muss lediglich für jedes (neue) Präsentationsformat einmalig ein konkreter Präsentationsformat-Generator entwickelt werden, um eine Unterstützung für beliebige personalisierte Multimedia-Anwendungen zu erhalten. Dazu ist jeweils eine konkrete Unterklasse zu `AbstractContentGenerator` und gegebenenfalls auch zu `AbstractLayoutGenerator` zu erstellen.

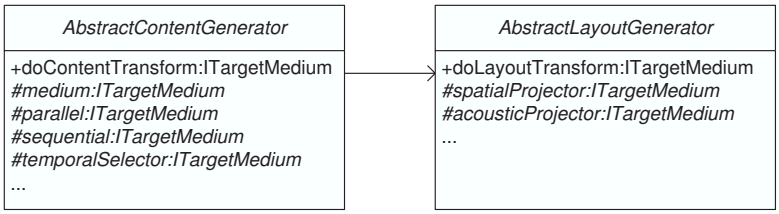

Abbildung 21.6: Klassendiagramm der abstrakten Klassen des Transformationsalgorithmus

21.5.6 Implementierung, Überarbeitung und Nutzung des MM4U-Frameworks

Für den Entwurf und die Implementierung des MM4U-Framworks sind zahlreiche Iterationen des in Abschnitt 21.4 beschriebenen Entwicklungsprozesses durchlaufen worden. Wie die Multimedia-Kompositionskomponente und die Präsentationsformat-Generatorenkomponente, so sind auch die restlichen Komponenten des MM4U-Frameworks so entworfen worden, dass sie für beliebige konkrete personalisierte Multimedia-Anwendungen einsetzbar sind. Der Entwurf und die Implementierung der einzelnen Komponenten war teilweise zeitlich versetzt. So wurde beispielsweise der Entwurf der Komponente für den Zugriff auf die Medienelemente überarbeitet, während die Implementierung an der Komponente zur Multimedia-Komposition bereits beendet war. Und während die Komponente zur Multimedia-Komposition implementiert wurde, befand sich die Komponente zur Präsentationsformat-Generierung noch in der initialen Entwurfsphase.

Die Nutzung des MM4U-Frameworks zur Entwicklung konkreter personalisierter Multimedia-Anwendungen bedeutet im Wesentlichen die Komposition und Wiederverwendung der bereits verfügbaren Implementierungen der Framework-Komponenten. Zum Beispiel können die Präsentationsformat-Generatorenkomponente und die Komponenten zum Zugriff auf die Benutzerprofile und die Medien typischerweise einfach durch unterschiedliche Konfiguration wiederverwendet werden. Die Multimedia-Kompositionskomponente dagegen hängt stark von der jeweiligen Anwendungsdomäne ab. Die konkrete Instanz dieser Komponente ändert sich daher mit der jeweils betrachteten Anwendungsdomäne. Typischerweise implementieren Entwickler von personalisierten Multimedia-Anwendungen ihre eigene Multimedia-Kompositionskomponente. Dazu werden die Basiskompositionselemente der Multimedia-Kompositionselemente verwendet. Außerdem kann beliebige andere Kompositions- und Personalisierungsfunktionalität wiederverwendet werden, die von anderen existierenden

Anwendungen, die auf dem MM4U-Framework basieren, zur Verfügung gestellt wird. Das MM4U-Framework selbst ist ebenfalls als Komponente realisiert. Dazu wurde eine weitere Gruppen-Hotspot-Karte erstellt, die die Flexibilitätsanforderungen an das Framework auf einer höheren Abstraktionsebene kapselt und die Reihenfolge definiert, in der die einzelnen Framework-Komponenten verwendet werden [SB05a].

Auf Basis des MM4U-Frameworks wurden bereits zahlreiche Beispielanwendungen entwickelt, wie etwa eine personalisierte multimediale Stadtführungsanwendung [SB04, BKS04]. Diese Anwendungen haben zusätzliche Rückmeldungen für die Entwicklung des Frameworks geliefert. Sie haben aufgezeigt, welche Stellen des Frameworks noch unflexibel waren, wo also zusätzliche Hotspots eingefügt werden mussten, und wo zu viel Flexibilität vorhanden war, so dass zusätzliche Schablonenmethoden eingebaut wurden.

Mit MPEG-21 [BVH+03] entsteht eine normative Infrastruktur für die Übertragung und Nutzung digitaler (multimedialer) Inhalte. Zentrales Element des Standards ist die Definition eines so genannten *Digital Item*, eines strukturierten digitalen Objektes, das eine Kombination aus (Medien-)Ressourcen, Metadaten und Struktur darstellt. Basierend auf dieser grundlegenden Einheit können Multimedia-Inhalte ausgetauscht, bearbeitet, gehandelt und wiedergegeben werden. Die Repräsentation multimedialer Inhalte im vorgestellten MM4U-Komponenten-Framework lässt sich in diese Entwicklung einbetten.

21.6 Fazit

Eine besondere Herausforderung beim Entwurf des MM4U-Frameworks war es, die Vielzahl der Flexibilitätsanforderungen an das Framework in Hinblick auf die verschiedenen Aspekte zur Personalisierung in eine allgemeine Software-Architektur zu gießen, die auch praktikabel ist. Beim ebenfalls sehr allgemein angelegten Standardreferenzmodell für intelligente Multimedia-Präsentationssysteme [BFF+97] ist das Hauptproblem, dass es in der Praxis nicht anwendbar ist. Das Referenzmodell beschreibt die Struktur von personalisierten Multimedia-Anwendungen auf der Ebene einzelner Aufgaben im Multimedia-Personalisierungsprozess, ohne dabei jedoch konkrete Hinweise zur Software-technischen Umsetzung zu geben. So muss der Entwurf der Anwendungsarchitektur von den Anwendungsentwicklern selbst vorgenommen werden. Das komponentenbasierte Framework MM4U gibt eine allgemeine Architektur für personalisierte Multimedia-Anwendungen vor, die unmittelbar von einer konkreten Anwendung genutzt und angepasst werden kann. Dabei liefern Richtlinien und Checklisten den Anwendungsentwicklern praktische Hinweise, wie bei

der Entwicklung von personalisierten Multimedia-Anwendungen mit dem MM4U-Frameworks vorzugehen ist.

Das MM4U-Framework ist nicht nur ein Beispiel für eine Multimedia-Architektur. Durch die Modifikation der Hotspot-getriebenen Entwicklungsmethode hinsichtlich einer expliziten Unterstützung zur Identifikation und Spezifikation der Flexibilitätsanforderungen an die Framework-Komponenten und die Anwendung dieses Prozesses zur Entwicklung des MM4U-Frameworks gibt es auch wertvolle Hinweise zur Verbesserung des Entwicklungsprozesses komponentenbasierter Frameworks.

Literaturverzeichnis

[Ack96] ACKERMANN, P.: *Developing Object-Oriented Multimedia Software – Based on MET++ Application Framework*. dpunkt.verlag, 1996

[AMR96] ANDRÉ, E.; MÜLLER, J.; RIST, T.: WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring. In: *EUROMEDIA'96*, 1996, S. 95–102

[App05] APPLE, USA: QuickTime. 2005, URL `http://www.apple.com/quicktime/`

[BBK+99] BALZERT, H.; BEHLE, A.; KELTER, U.; NAGL, M.; PAUEN, P.; SCHÄFER, W.; SIX, H.; VOSS, J.; WADSACK, J.; WEIDAUER, C.; WESTFECHTEL, B.: Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme. September 1999

[BFF+97] BORDEGONI, M.; FACONTI, G.; FEINER, S.; MAYBURY, M. T.; RIST, T.; RUGGIERI, S.; TRAHANIAS, P.; WILSON, M.: A standard reference model for intelligent multimedia presentation systems. In: *Computer standards & interfaces* 18 (1997), Nr. 6-7, S. 477–496, doi:10.1016/S0920-5489(97)00013-5

[BK03] BLOCK, M.; KONRAD, S.: *Personalized and multimedia Webservice – Sightseeing4U*. Individuelle Projekte, Carl von Ossietzky Universität Oldenburg, Fakultät II, Department für Informatik, September 2003

[BKS04] BOLL, S.; KRÖSCHE, J.; SCHERP, A.: Personalized Mobile Multimedia meets Location-Based Services. In: DADAM, P.; REICHERT, M. (Hrsg.), *Multimedia-Informationssysteme Workshop im Rahmen der 34. Jahrestagung der Gesellschaft für Informatik (Informatik 2004), Bd. 2*, Ulm, Deutschland: GI, September 2004, Bd. 51 von *LNI*, S. 64–69

[BKW03] BOLL, S.; KRÖSCHE, J.; WEGENER, C.: Paper chase revisited – a real world game meets hypermedia. In: *Proc. der Intl. Conference on Hypertext (HT03)*, ACM, 2003, S. 126–127

[BMM+99] BOSCH, J.; MOLIN, P.; MATTSSON, M.; BENGTSSON, P.; FAYAD, M. E.: Framework Problems and Experiences. In: FAYAD et al. [FSJ99], S. 55–82

[BN03] BRUNS, K.; NEIDHOLD, B.: *Audio-, Video- und Grafikprogrammierung*. München, Wien: Fachbuchverlag Leipzig im Carl Hanser Verlag, 2003

[Bol03] BOLL, S.: Vienna 4 U – What Web Services can do for personalized multimedia applications. In: *Seventh Multi-Conference on Systemics (SCI 2003), Cybernetics and Informatics*, Juli 2003, S. 220–225

[BVH+03] BURNETT, I.; VAN DE WALLE, R.; HILL, K.; BORMANS, J.; PEREIRA, F.: MPEG-21: Goals and Achievements. In: *IEEE MultiMedia* 10 (2003), Nr. 4, S. 60–70

[DEM+99] DEPKE, R.; ENGELS, G.; MEHNER, K.; SAUER, S.; WAGNER, A.: Ein Vorgehensmodell für die Multimedia-Entwicklung mit Autorensystemen. In: *Informatik – Forschung und Entwicklung* 14 (1999), Nr. 2, S. 83–94

[DHM99] DUKE, D. J.; HERMAN, I.; MARSCHALL, M. S.: *PREMO: A Framework for Multimedia Middleware – Specification, Rationale, and Java Binding*. LNCS 1591, Springer-Verlag, 1999

[ESN03] ENGELS, G.; SAUER, S.; NEU, B.: Integrating Software Engineering and User-centred Design for Multimedia Software Developments. In: *In Proc. IEEE Symposia on Human-Centric Computing Languages and Environments – Symposium on Visual/Multimedia Software Engineering*, Auckland, New Zealand: IEEE Computer Society Press, Oktober 2003

[FSJ99] FAYAD, M. E.; SCHMIDT, D. C.; JOHNSON, R. E. (Hrsg.): *Building Application Frameworks – Object-Oriented Foundations of Framework Design*. Wiley computer publishing, John Wiley & Sons, 1999

[Gam92] GAMMA, E.: *Objektorientierte Software-Entwicklung am Beispiel von ET++ – Design-Muster, Klassenbibliothek, Werkzeuge*. Springer-Verlag, 1992

[GHJV04] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software*. Programmer's Choice, Addison-Wesley, Juli 2004

[IBM04] IBM: QBIC Home Page. 2004, URL `http://wwwqbic.almaden.ibm.com/`

[KPW04] KING, R.; POPITSCH, N.; WESTERMANN, U.: METIS: a flexible database foundation for unified media management. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, ACM Press, 2004, S. 744–745, doi:http://doi.acm.org/10.1145/1027527.1027695

[Mac05] MACROMEDIA, INC., USA: Macromedia – Flash MX 2004. 1995–2005, URL http://www.macromedia.com/software/flash/

[Mic05] MICROSOFT CORPORATION, USA: Microsoft DirectShow 9.0. 2005, URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/directshow.asp

[NMM05] *Network-Integrated Multimedia Middleware*. Technischer Bericht, Computer Graphics Lab, Saarland Universität, Saarbrücken, 2005, URL http://www.networkmultimedia.org/

[Ora04] ORACLE CORPORATION: *interMedia*. Technischer Bericht, 2004, URL http://www.oracle.com/technology/products/intermedia/index.html

[Ora05] ORACLE CORPORATION: *interMedia Documentation*. Technischer Bericht, 2005, URL http://www.oracle.com/technology/documentation/intermedia.html

[PLV97] POSNAK, E.; LAVENDER, R.; VIN, H.: An adaptive framework for developing multimedia software components. In: *Communications of the ACM* 40 (1997), Nr. 10, S. 43–47, doi:10.1145/262793.262802

[Pre95] PREE, W.: *Design Patterns for Object-Oriented Software Development*. ACM Press Books, Addison-Wesley, 1995

[Pre97] PREE, W.: *Komponentenbasierte Softwareentwicklung mit Frameworks*. dpunkt.verlag, 1997

[PVL96] POSNAK, E.; VIN, H.; LAVENDER, R.: Presentation Processing Support for Adaptive Multimedia Applications. In: *Proc. of Multimedia Computing and Networking 1996 (MMCN96)*, Januar 1996, S. 234–245

[Rea05] REALNETWORKS: *Helix Community*. Technischer Bericht, 2005, URL https://helixcommunity.org/

[Ric03] RICHTER, C.: *Entwurf und Realisierung eines Web-basierten personalisierten multimedia Musik-Newsletters*. Individuelles Projekt, Carl von Ossietzky Universität Oldenburg, Juli 2003

[RS99] ROUT, T.; SHERWOOD, C.: Software Engineering Standards and the Development of Multimedia-Based Systems. In: *In Fourth IEEE International Symposium and Forum on Software Engineering Standards*, Mai 1999

[RWP04] ROSS, K.; WESTERMANN, G. U.; POPITSCH, N.: METIS – A Flexible Database Solution for the Management of Multimedia Assets. In: *Proc. of the 10th International Workshop on Multimedia Information Systems (MIS 2004)*, August 2004

[Saw95] SAWHNEY, M.: *Entwicklung eines Vorgehensmodells für die Multimedia-Anwendungsentwicklung am Beispiel eines Informations- und Orientierungssystems für eine Universität*. Diplomarbeit, Universität Osnabrück, Fachbereich Wirtschaftswissenschaften, Osnabrück, Juni 1995

[SB04] SCHERP, A.; BOLL, S.: Generic support for personalized mobile multimedia tourist applications. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia – Technische Demonstration*, New York, NY, USA: ACM Press, Oktober 2004, S. 178–179, doi:http://doi.acm.org/10.1145/1027527.1027566

[SB05a] SCHERP, A.; BOLL, S.: A lightweight process model and development methodology for component frameworks. In: *Proceedings of the tenth International Workshop on Component-Oriented Programming*, Juli 2005, URL `http://research.microsoft.com/~cszypers/events/WCOP2005/`

[SB05b] SCHERP, A.; BOLL, S.: MM4U – A framework for creating personalized multimedia content. In: SRINIVASAN, U.; NEPAL, S. (Hrsg.), *Managing Multimedia Semantics*, Hershey, PA, USA: IRM Press, Kap. 11, 2005

[SB05c] SCHERP, A.; BOLL, S.: Paving the Last Mile for Multi-Channel Multimedia Presentation Generation. In: CHEN, Y.-P. P. (Hrsg.), *Procedings of the 11th Multimedia Modeling (MMM) Conference*, Melbourne, Australia: IEEE Computer Society, Januar 2005, S. 190–197

[SN95] STEINMETZ, R.; NAHRSTEDT, K.: *Multimedia: Computing, Communications and Applications*. Prentice Hall, 1995

[Ste00] STEINMETZ, R.: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer-Verlag, 3. Aufl., 2000

[Sun05a] SUN MICROSYSTEMS: *Java Media Framework API (JMF)*. Technischer Bericht, 1994–2005, URL `http://java.sun.com/products/java-media/jmf/`

[Sun05b] SUN MICROSYSTEMS: *JMF 2.0 Documentation Downloads*. Technischer Bericht, 1994–2005, URL `http://java.sun.com/products/java-media/jmf/2.1.1/specdownload.html`

[van04] *The Cuypers Multimedia Transformation Engine*. Technischer Bericht, CWI, 2004, URL `http://media.cwi.nl:8080/demo/`

[W3C04] Scalable Vector Graphics (SVG) 1.2. 2004, URL `http://www.w3.org/TR/2004/WD-SVG12-20040510/`

[W3C05] Synchronized Multimedia Integration Language (SMIL 2.0) - [Second Edition]. 2005, URL http://www.w3.org/TR/2005/REC-SMIL2-20050107/