Adaptive Dokumentbereitstellung von Erfahrungsdaten

Wolfgang Mahnke, Ulrich Marder, Norbert Ritter

SFB 501 Bericht 04/2002

Adaptive Dokumentbereitstellung von Erfahrungsdaten

Wolfgang Mahnke, Ulrich Marder, Norbert Ritter*

{mahnke,marder,ritter}@informatik.uni-kl.de

Sonderforschungsbereich 501
Technischer Bericht 04/2002

AG Datenbanken und Informationssysteme

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
Germany

*. Neue Adresse: AB Verteilte Systeme und Informationssysteme, Fachbereich Informatik, Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, E-Mail: ritter@informatik.uni-hamburg.de

Zusammenfassung

Die bisherige prototypische Realisierung des für den SFB 501 entwickelten Reuse-Repository unterstützt noch keine Datenunabhängigkeit für die die eigentlichen Erfahrungsdaten beinhaltenden Dokumente, deren Formate bislang noch voll und ganz von dem Anwender, der die Daten in das Reuse-Repository einstellt, abhängen. Da Software-Entwicklungswerkzeuge jedoch oftmals nicht standardisierte bzw. wenig gebräuchliche Datenformate einsetzen, ist es generell erforderlich, dass das Reuse-Repository alternative Repräsentationen insbesondere für diejenigen Anwender bereit stellt, die nur über Werkzeuge zur Darstellung und Bearbeitung von Standard-Datenformaten wie z. B. PDF verfügen.

Das VirtualMedia-Modell wurde – ebenfalls im Rahmen des SFB 501 – unabhängig vom Reuse-Repository zu dem Zweck entwickelt, Datenunabhängigkeit für große, multimediale Objekte zu realisieren. Es bietet mithilfe eines leicht zu benutzenden Anfragemechanismus' die Möglichkeit, externe Repräsentationen dynamisch zu erzeugen, ohne dass der Anwendung die entsprechende interne Repräsentation bekannt sein muss. Das VirtualMedia-Modell bietet somit prinzipiell die dem Reuse-Repository noch fehlende Funktionalität für eine flexible, adaptive Bereitstellung von Erfahrungsdaten. Wir legen in diesem Bericht dar, wie dieses Modell in die Architektur des Reuse-Repository zu integrieren ist. Anhand eines ausführlichen Beispiels wird zudem verdeutlicht, wie eine Anfrage zur Bereitstellung eines externen Dokuments verarbeitet werden würde.

Schlüsselwörter:

Erfahrungsdatenbank, Datenunabhängigkeit, Multimedia-Metacomputing

Inhaltsverzeichnis

1 Motivation

Das im Rahmen des SFB 501 entwickelte, Datenbank-gestützte Reuse-Repository [3, 4, 8, 13] dient der Unterstützung von Softwareentwicklungsprojekten nach dem Quality-Improvement-Paradigma (QIP) [1, 3]. Das QIP bietet einen ablaufbezogenen Rahmen zur umfassenden Wiederverwendung von Software-Artefakten. Das Reuse-Repository verwaltet dabei im Wesentlichen Daten, die Erfahrungen von Softwareentwicklungsprozessen repräsentieren, sowie Daten, die allgemeine Erfahrungen bezüglich der Softwareentwicklung darstellen. Das Spektrum reicht hier von Beschreibungen/Bewertungen von in früheren Projekten genutzten Methoden und Technologien über Prozessmodelle bis hin zu konkreten Produktdaten, die in einem neuen Prozess wiederverwendet bzw. zwecks neuartiger Verwendung angepasst werden können. Diese sogenannten Erfahrungsdaten bestehen in der Regel aus Dokumenten sowie beschreibenden Metadaten, die in einem Charakterisierungsvektor (CV) zusammengefasst werden. Während die CVs im aktuellen Prototypen des Reuse-Repository feingranular in einer Tabellenhierarchie gespeichert werden, erfolgt die Speicherung der Dokumente in BLOBs (binary large objects), dem Datenbank-Äquivalent einer Datei. Dies ist ausreichend, da eine – zum Teil ähnlichkeitsbasierte – Suche [5] nur auf den CVs stattfindet. Problematisch ist allerdings, dass inhaltsgleiche Dokumente in verschiedenen Repräsentationen vorkommen können, bspw. kann die Beschreibung eines Design-Pattern sowohl als PostScript-, Word- und PDF-Datei vorliegen als auch in einem speziellen Format für ein bestimmtes Entwicklungswerkzeug. Dies ist im aktuellen Prototypen derart gelöst, dass für zusammengehörige Erfahrungsdaten neben dem CV eine Menge von Repräsentationen gespeichert wird, die den gleichen Inhalt in unterschiedlichen Dokumentformaten bereit stellt. Dabei müssen die verschiedenen Repräsentationen vom Benutzer, der die Daten eingibt, bereit gestellt werden. Dieser Ansatz führt jedoch zu folgenden Nachteilen:

- Es kann nicht systemseitig sichergestellt werden, dass verschiedene Repräsentationen den gleichen Inhalt beschreiben. Dies muss vom Benutzer gewährleistet werden, der die Repräsentationen bereitstellt oder wartet. Insbesondere bei Wartungsarbeiten, d. h. Änderungen der Daten, die bereits in vielen verschiedenen Repräsentationen vorliegen, ist dies sehr umständlich.
- Das Reuse-Repository kann ein Dokument nur dann in einem bestimmten Format zur Verfügung stellen, wenn das Dokument vorher auch in diesem Format eingegeben worden ist. Falls ein Benutzer ein anderes Format benötigt, muss sich der Benutzer um eine entsprechende Transformation des Dokuments selber kümmern. Dafür muss jeder Benutzer die entsprechenden Werkzeuge für eine Transformation zur Verfügung haben, andernfalls können einige Erfahrungen nicht von allen Benutzern verwendet werden.
- Der Benutzer kann nicht spezifizieren, in welchem Format er ein Dokument erhält, er kann lediglich aus den vorhandenen Repräsentationen eine ihm geeignet scheinende auswählen. Idealerweise sollte in der Systemumgebung des Benutzers eingestellt sein, welche Formate er verwenden kann, d. h., welche Werkzeuge bei ihm vorhanden sind, und über eine Prioritätenliste automatisch das geeignete Format ausgewählt und bereitgestellt werden.

Neben den genannten Problemen ergibt sich beim bisher verfolgten Ansatz eine weitere Schwierigkeit bei der Verwaltung von projektspezifischen Produktdaten, die nicht in Dokumenten, sondern in

einem Datenbank-gestützten Produktdatenschema verwaltet werden. Obwohl die Daten in diesem Fall feingranular ins Reuse-Repository gebracht bzw. von diesem verwaltet werden können, muss nach wie vor die auf Dokumente und CVs ausgelegte Schnittstelle des Reuse-Repository verwendbar sein [7].

Das in [9, 10] vorgestellte VirtualMedia-Modell adressiert genau die oben beschriebenen Probleme und ermöglicht eine adaptive Dokumentenbereitstellung im gewünschten Format. Im folgenden beschreiben wir, wie VirtualMedia in das Reuse-Repository integriert werden kann, um den Benutzern des Reuse-Repository eine ideale Unterstützung bei ihren Aufgaben zu gewähren.

Dazu stellen wir in Kapitel 2 das Reuse-Repository genauer vor, präsentieren in Kapitel 3 den VirtualMedia Ansatz und beschreiben in Kapitel 4 die Architektur eines um das VirtualMedia-Modell ergänzten Reuse-Repository. Zum besseren Verständnis dieses integrierten Ansatzes erläutern wir diesen in Kapitel 5 anhand eines Beispiels und schließen mit einer Zusammenfassung sowie einem Ausblick in Kapitel 6.

2 Das SFB-501-Reuse-Repository

Das SFB-501-Reuse-Repository (kurz Reuse-Repository) bietet zum einen die Funktionalität der im Zusammenhang mit dem QIP beschriebenen *Experience Base* [1], in der Daten, die Erfahrungen vorhandener Projekte oder allgemeine Erfahrungen zur Softwareentwicklung repräsentieren, gespeichert werden. Zum anderen wird der Ansatz der Experience Base noch erweitert, indem die Daten, die während eines Projektes anfallen, ebenfalls vom Reuse-Repository verwaltet werden. Damit wird eine umfassende Unterstützung von Softwarentwicklungsprojekten ermöglicht, und es können in einfacher Weise die Beziehungen zwischen den Daten der Erfahrungen und deren Verwendung in Projekten gewartet werden.

Unter *Erfahrungsdaten* verstehen wir die Repräsentation von Erfahrungen, d. h. in Projekten erzeugte oder anderweitig bereitgestellte Artefakte[1], die eine potenzielle Wiederverwendbarkeit in nachfolgenden Projekten aufweisen. Die Verwaltung dieser Erfahrungsdaten in der aktuellen prototypischen Implementierung des Reuse-Repository wird in Abschnitt 2.1 beschrieben. *Produktdaten* sind Daten, die sich in der Bearbeitung durch momentan laufende Projekte befinden und daher (aus der allgemeinen Nutzer-Sicht des Reuse-Repository) natürlicherweise von beschränkter Sichtbarkeit und Gültigkeit sind. In Abschnitt 2.2 werden sowohl die Möglichkeiten zur Produktdatenverwaltung im aktuellen Prototypen des Reuse-Repository erläutert als auch ein erweiterter Ansatz zur integrierten Verwaltung von Produktdaten vorgestellt.

1. Unter dem Begriff ‘Artefakt’ verstehen wir etwas Physisches, z. B. eine Datei oder eine Menge von Tupeln in einer DB, unter ‘Erfahrung’ hingegen etwas Abstraktes, nicht zuletzt wegen des eher subjektiven Charakters dieses Begriffs im allgemeinen Sprachgebrauch.

2.1 Erfahrungsdatenverwaltung

Die Erfahrungsdaten werden in einem bestimmten Bereich des Reuse-Repository verwaltet, der sogenannten *OWS* (von engl.: *organization-wide section*). In diesem Bereich werden die im SFB-Bericht 08/99 [4] beschriebenen Strukturen zur Klassifikation der Erfahrungsdaten verwendet. Innerhalb dieser Strukturen wird jede einzelne erfasste Erfahrung als ein *Erfahrungsdateneintrag* (EDE) gespeichert. Ein EDE besteht aus einem *Erfahrungsrepräsentant* (ER), der die wiederverwendbaren Erfahrungsdaten enthält, sowie einem *Charakterisierungsvektor* (CV), der die Erfahrungsdaten beschreibt (siehe Abbildung 1). Da inhaltsgleiche Erfahrungsdaten in verschiedenen Formaten vorliegen können, wie das bereits erwähnte Design-Pattern, das sowohl im PostScript-, Word- und PDF-Format als auch in einem speziellen Format für ein bestimmtes Entwicklungswerkzeug gespeichert werden kann, verwaltet der ER eine Menge von *Repräsentationen* (Rep). Jede Rep besteht aus einem Erfahrungsdatum in einem Format. Hier ist allerdings nicht unbedingt das Format eines Dokuments gemeint. Eine Rep kann auch aus mehreren Dokumenten verschiedener Formate bestehen, wie dies beispielsweise bei einer HTML-Seite, die Bilder enthält, der Fall ist. Die Rep würde im HTML-Format vorliegen, einzelne Dokumente der Rep aber in GIF-, JPG- oder anderem Format. In der aktuellen prototypischen Implementierung wird jede Rep als eine Menge von BLOBs im Datenbanksystem verwaltet.

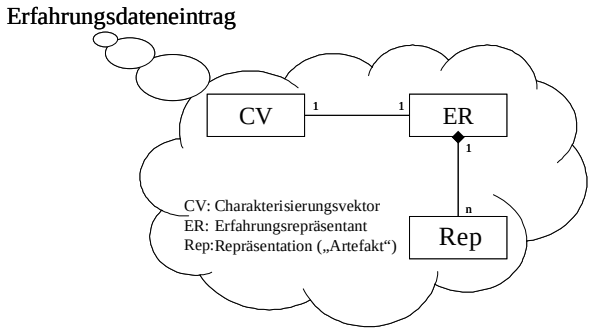

Abbildung 1: Konzeptionelles Schema eines Erfahrungsdateneintrags

Die Äquivalenz zwischen den Reps ist durch den ER modelliert, alle äquivalenten Reps besitzen dadurch zusammen einen CV. Dieser CV wird zum einen benötigt, um die Metadaten der Erfahrungsdaten zu verwalten. Dies sind sowohl Daten, die i. d. R. nicht direkt aus den Erfahrungsdaten extrahiert werden können, wie beispielsweise der Entstehungszeitpunkt, in welchem Kontext die Erfahrung anwendbar ist usw., als auch Daten, die auch implizit in den Reps enthalten sind, beispielsweise die verwendete Programmiersprache, Lines of Code etc. Dies ist erforderlich, um die Suche auf den EDEs zu ermöglichen. Da die Erfahrungsdaten in beliebigen Formaten vorliegen können, ist es nicht möglich, inhaltsbezogen direkt auf den Dokumenten zu suchen. Zum anderen sind in den CVs die Beziehungen zwischen verschiedenen Erfahrungsdaten modelliert, z. B., welche Technologie von welchem Prozessmodell verwendet wird. Eine genaue Beschreibung der Beziehungen kann [4] entnommen werden.

Auf die vom Reuse-Repository angebotenen Benutzerfunktionen zur Manipulation von Erfahrungsdaten bzw. für die ähnlichkeitsbasierte Suche wollen wir an dieser Stelle nicht genauer eingehen. Es sei auf [3, 4] verwiesen.

2.2 Produktdatenverwaltung

Der aktuelle Prototyp des Reuse-Repository verfolgt einen allgemeinen, generischen Ansatz hinsichtlich der Verwaltung von Produktdaten, berücksichtigt aber keine spezifischen Produktdatenmodelle,

wie bereits oben erwähnt. Dennoch werden in diesem Abschnitt sowohl die zugehörigen Verwaltungsmechanismen des gegenwärtigen Prototypen angesprochen als auch ein Konzept zur Unterstützung von Produktdatenmodellen beschrieben.

Produktdatenverwaltung im aktuellen Prototyp

Wie bei der Erfahrungsdatenverwaltung ist im gegenwärtigen Prototypen auch die Produktdatenverwaltung in einem bestimmten Bereich angesiedelt, der ESS (engl.: *experiment-specific section).* Dabei wird jedem laufenden Projekt ein spezieller Bereich innerhalb des ESS zugeordnet, der *ESS-P*. Die Daten sind dort zunächst, d. h. bis auf explizite, selektive Freigabe einzelner Inhalte, der Sichtbarkeit von Nicht-Projektbeteiligten entzogen. Zur Unterstützung der Wiederverwendung können von Projektbeteiligten Daten aus der OWS als Ausgangspunkt für weitere Aktivitäten in den projektspezifischen Teil der ESS kopiert werden. Entsprechend werden auch Produktdaten in der bereits oben angesprochenen Form als BLOBs mit zugehörigen Charakterisierungsvektoren verwaltet. Die Verarbeitung von Produktdaten geschieht in aller Regel durch die Anwendung von Entwicklungswerkzeugen, die in einer Art ‘losen Kopplung’ mit dem Reuse-Repository zusammenarbeiten. Die Eingabedaten werden dafür vor dem Aufruf des Werkzeugs im Dateisystem bereitgestellt und die vom Werkzeug modifizierten Daten werden anschließend wieder in das Reuse-Repository gebracht.

Dieser Ablauf der losen Kopplung eignet sich gut, solange die verwendeten Werkzeuge sowieso auf Dateien des Dateisystems aufbauen. Falls ein Werkzeug seine Daten jedoch feingranularer in einer Datenbank verwaltet, ist dieser Ansatz schwerlich realisierbar und auch nicht sinnvoll. Falls das Wissen um das Produktdatenmodell besteht, sollte es folglich auch im Reuse-Repository verwendet werden, um neben der dokumentenzentrierten Sicht auch einen direkten Zugriff auf die Produktdaten zu ermöglichen.

Produktdatenverwaltung mit integriertem Produktdatenmodell

Falls ein Produktdatenmodell (PDM) direkt integriert werden kann, d. h., die Produktdaten werden feingranular direkt im Reuse-Repository gespeichert, ergibt sich die Möglichkeit, die Produktdaten in die bisher im Reuse-Repository verwendete Sicht auf Produkt- und Erfahrungsdaten zu integrieren. Dies ist notwendig, um nach wie vor die gleiche Schnittstelle auf die Produkt- und Erfahrungsdaten zu gewährleisten. Das zugrundeliegende Konzept ist dabei die Bereitstellung von Reps durch Sichtenbildung auf den Produktdatenmodellen. Ein *Repräsentationsdeskriptor* (RD) spezifiziert dabei deklarativ eine Produktdatenmodell-basierte Repräsentation (i. A. eine Menge von SQL-Ausdrücken).

Wegen der schwerwiegenden Probleme bezüglich der Konsistenzerhaltung bei einem solchen intensional orientierten Ansatz (verursacht durch Änderungsanomalien, siehe [7] für Details) betrachten wir im Folgenden aber nur den *extensional orientierten Lösungsansatz*, bei dem die Konsistenzerhaltung einfacher gewährt werden kann, der allerdings auch nur für bestimmte Produktdatenmodelle, wie unten erläutert, anwendbar ist. Hier wird die *Extension* des Repräsentationsdeskriptors (kurz RD-Extension bzw. RDX) als primäre (PDM-)Repräsentation verwendet. In diesem Fall beziehen sich die CVs (d. h., deren Inhalte) direkt auf die Produktdaten, d. h., es werden entsprechende Referenzen erzeugt und verwaltet (im Gegensatz zum intensional orientierten Lösungsansatz, der die Bezüge durch

inhaltsbezogene Prädikate herstellt [7]).

Um Konsistenzprobleme zu überwinden, die durch das direkte Manipulieren der Produktdaten entstehen können, muss es möglich sein, die aktuell gültige Zuordnung von Produktdaten zu RD-Extensionen, ausgehend von jedem einzelnen Produktdatum, „mit vertretbarem Aufwand“ zu bestimmen[2]. Dies wäre z. B. der Fall, wenn das PDM bereits Zugriffsmöglichkeiten vorsieht, die eine effiziente Berechnung dieser Zuordnung ermöglichen, oder solche Funktionen ohne wesentliche Beeinträchtigung der PDM-basierten Anwendungen hinzugefügt werden können. Als „wesentliche Beeinträchtigung“ könnte man hierbei z. B. einen zu hohen Verwaltungsaufwand für zusätzliche Strukturen, die nötig sein können, um Referenzen bidirektional zu machen, ansehen. Diese Voraussetzungen sind am wahrscheinlichsten bei objektorientierten PDM gegeben, wenn es möglich ist, Erfahrung durch einzelne Objektausprägungen zu repräsentieren.

In Abbildung 2 ist der Lösungsansatz zusammengefasst. Dem ER ist genau ein RDX zugeordnet, der Beziehungen zu den Produktdaten enthält. Falls über die Reuse-Repository-Sicht auf ein solches Produktdatum zugegriffen werden soll, erzeugt ein Formatgenerator die Repräsentation im gewünschten Format. Es ist möglich, aus Gründen der Optimierung sekundäre Repräsentationen intern zu speichern. Falls die Repräsentation geändert wurde, müssen die Daten über einen Check-in-Filter in die Produktdaten eingebracht werden und der RDX so angepasst werden, dass beim erneuten Auslesen genau die eingegebene Repräsentation entsteht. Sekundäre Repräsentationen des RDX verlieren ihre Gültigkeit und müssen gelöscht werden. Falls die Produktdaten direkt geändert werden und dies (überprüfbar durch die Beziehungen zu den RDX) einen (oder mehrere) RDX betrifft, müssen die sekundären Repräsentationen dieser RDX gelöscht und deren CVs markiert werden, so dass einem Anwender des Reuse-Repository bewusst ist, dass sich die Daten der Repräsentation nach der Erstellung des CVs geändert haben.

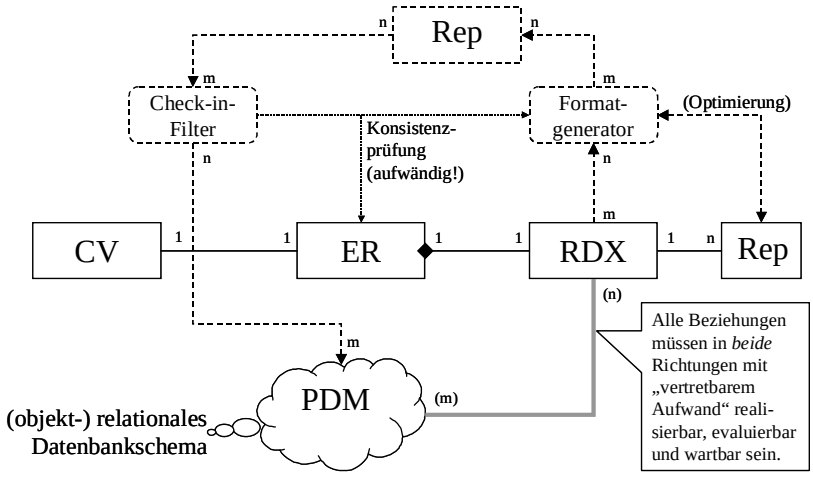

Abbildung 2: Konzeptionelles Schema für den extensional orientierten Lösungsansatz

3 Das VirtualMedia-Modell

VirtualMedia wurde im Rahmen des SFB-501 als ein semantisches Modell für sog. Multimedia-Metacomputing-Umgebungen entwickelt [11]. Dieses Modell definiert

2. Beim intensional orientierten Ansatz wäre es erforderlich, bei jeder dieser Überprüfungen sämtliche Repräsentationsdeskriptoren zu evaluieren, d. h., die aktuelle Extension zu ermitteln. Dies betrachten wir als nicht vertretbaren Aufwand.

- ein abstraktes Kollaborationsmodell,
- die externe Repräsentation von Medienobjekten, Operationen und Benutzeranfragen, und
- wie Benutzeranfragen vorverarbeitet (transformiert) werden, um in der Metacomputing-Umgebung optimal ausgeführt werden zu können.

VirtualMedia gibt den Benutzern bzw. den Anwendungsprogrammierern die Möglichkeit, sowohl ad-hoc Anfragen zur Bereitstellung multimedialer Objekte und Dokumente zu stellen als auch vorgefertigte, parametrisierte Musteranfragen hierfür zu verwenden. Das Modell definiert formal, wie diese Anfragen automatisch in semantisch optimale, ausführbare Pläne für eine gegebene Metacomputing-Umgebung überführt werden. Die Vorteile eines solchen Anfragemechanismus' gegenüber dem heute noch üblichen direkten Erstellen von Programmen für die Mediendatenmanipulation liegen auf der Hand:

- *Benutzerfreundlichkeit*: Benutzeranfragen sind wesentlich einfacher – bei entsprechender Werkzeugunterstützung sogar durch Nicht-Programmierer – zu erstellen, weil man sich nicht selbst darum kümmern muss, dass die Signaturen von Medienoperationen und zu verarbeitenden Medienobjekten zueinander passen.
- *Stabilität*: Benutzeranfragen sind stabiler als Programme. Dies liegt vor allem an der Instabilität von Metacomputing-Umgebungen, in denen Ressourcen vorübergehend oder dauerhaft ausfallen, ersetzt werden sowie, z. B. im Falle von Software-Ressourcen, gelegentlich aktualisiert werden. Ein Programm scheitert gewöhnlich, wenn eine notwendige Ressource nicht mehr verfügbar ist oder aufgrund einer Aktualisierung eine veränderte Schnittstelle oder sonstige veränderte Eigenschaften besitzt. Eine Benutzeranfrage wird hingegen bei jeder Ausführung neu verarbeitet, wobei veränderte Ressourcen jeweils in anderen optimalen (dynamisch erzeugten) Programmen resultieren.
- *Optimierung*: Eine Benutzeranfrage kann, gesteuert durch verschiedene, z. T. sehr flüchtige Optimierungskriterien (z. B. die Systemauslastung), in unterschiedliche „optimale" Programme transformiert werden. Hierbei können verschiedene, teilweise einander entgegengesetzte Aspekte zum Tragen kommen, z. B. zeitliche Restriktionen (Latenzzeit), Kostenschranken, Ressourcenauslastung usw.

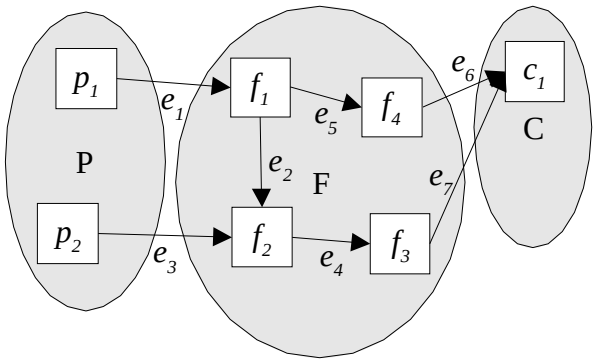

Abbildung 3: Illustration des abstrakten Kollaborationsmodells

Aus Platzgründen skizzieren wir in den folgenden Abschnitten nur einige wesentliche Aspekte und Eigenschaften des VirtualMedia-Modells. Für eine detaillierte Beschreibung wird auf [10] verwiesen.

3.1 Abstraktes Kollaborationsmodell

Das Kollaborationsmodell basiert auf dem Konzept der Filtergraphen (s. Abbildung 3), vergleichbar dem in [2] eingeführten Modell. Die Startknoten eines solchen Graphen sind die Medienproduzenten p_i (von einem Server verwaltete Medienobjekte oder Live-Medienquellen) und die Endknoten sind die Medienkonsumenten c_i (z. B. eine Client-Applikation). Die Zwischenknoten repräsentieren die Medienfilter f_i, welche als Basisoperatoren für beliebige Medienmanipulationen dienen. Die Kanten können demnach als Repräsentanten von Medienobjekten verstanden werden, welche von einer Medienquelle (einem Produzenten oder Filter) zu einer Mediensenke (einem Konsumenten oder Filter) übertragen werden.

Der in Abbildung 3 dargestellte Graph ist wie folgt zu interpretieren: Es gibt einen Medienproduzenten p_1, der ein Medienobjekt erzeugt, welches zum Medienfilter f_1 gesendet wird. Der Filter f_1 erzeugt aus seiner Eingabe wiederum zwei neue Medienobjekte, welche weiter zu den Medienfiltern f_2 bzw. f_4 übertragen werden usw. Keiner der Knoten ist an ein konkretes physisches Objekt oder eine Ressource gebunden. Vielmehr repräsentiert jeder Knoten nur die Semantik einer bestimmten Komponente der Medienverarbeitung, z. B. eines Medienservers oder Filters. Deshalb wird dieses Modell als ein *abstraktes* Kollaborationsmodell bezeichnet.

3.2 Abstrakte Mediensemantik

Um das Kollaborationsmodell für das Multimedia-Metacomputing verwenden zu können, müssen die Filtergraphen mit mehr Medien-spezifischer Semantik angereichert werden. Zu diesem Zweck werden die sog. *Mediensignaturen* eingeführt. Solch eine Signatur besteht aus einer Menge von Eigenschaften, die zu einer der folgenden Kategorien gehören: Typeigenschaften (Format), Qualitätseigenschaften und Inhaltseigenschaften. Für die Filter gibt es zudem sog. *Filtersignaturen*, welche sich aus funktionalen und nicht-funktionalen Eigenschaften zusammen setzen.

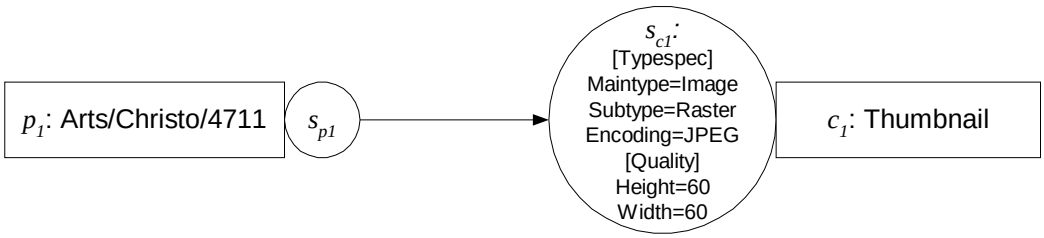

Abbildung 4: Beispiel-Filtergraph mit Signaturen (aus [6])

Abbildung 4 zeigt einen einfachen Beispiel-Filtergraphen mit Signaturen, der nur einen Produzentenknoten p_1 und einen Konsumentenknoten c_1 enthält. Die Kante besitzt zwei Mediensignaturen. Die erste beschreibt das Medienobjekt, welches von dem Knoten erzeugt wird, von dem die Kante ausgeht, und die zweite beschreibt das Medienobjekt, welches der Knoten erwartet, zu dem die Kante hinführt. Im Beispiel ist die erste Signatur leer, d. h., die Eigenschaften des Medienobjekts sind nicht bekannt. Die zweite Signatur enthält hingegen Typ- und Qualitätseigenschaften, die ein kleines Bildobjekt charakterisieren. Folglich lautet die durch den Graphen formal beschriebene Semantik: „Erzeuge aus einem beliebigen Medienobjekt ein Vorschaubild (Thumbnail)“. Dies verdeutlicht, dass das VirtualMedia-Modell geeignet ist, die Transformation und Manipulation von Medienobjekten auf ei-

ner sehr abstrakten, deskriptiven Ebene zu spezifizieren. Aus Sicht einer Anwendung wird hierbei die beabsichtigte Semantik hinreichend formal und präzise ausgedrückt, ohne vorwegzunehmen, wie das System die gegebenenfalls notwendigen Medienoperationen später realisiert.

3.3 Anfrageverarbeitung

Das Hinzufügen zweier Mediensignaturen zu jeder Kante mit der oben beschriebenen Semantik macht aus manchen Kanten etwas, das wir als *magische Kanäle* bezeichnen. Dies trifft beispielsweise auf die Kante in Abbildung 4 zu: die Signatur s_{c1} spezifiziert ein Bildobjekt mit einigen konkret festgelegten Eigenschaften, nämlich, dass es ein JPEG-Bild mit einer bestimmten Größe sein soll. Die leere Signatur s_{p1} bedeutet jedoch, dass am Eingang dieser Kante ein Medienobjekt mit unbestimmten, also beliebigen Eigenschaften akzeptiert werden soll. Das VirtualMedia-Modell führt für derartige Fälle eine Metrik ein, die es erlaubt, die *Signaturdifferenz* entlang einer Kante zu berechnen. Immer, wenn diese Differenz ungleich Null ist, repräsentiert die betroffene Kante eine noch unbestimmte potenzielle Medienoperation, woraus sich die Bezeichnung als „magischer Kanal“ ableitet.

Demzufolge ist eines der Hauptziele der Anfrageverarbeitung im VirtualMedia-Modell das Aufdecken der in den magischen Kanälen des Anfragegraphen steckenden Semantik, so dass ein semantisch äquivalenter Ergebnisgraph entsteht, der keine Signaturdifferenzen mehr enthält und somit durch konkrete Verarbeitungskomponenten, die durch normale Kanäle miteinander verbunden sind, instanzierbar ist. Um diese Umformung des Anfragegraphen zu ermöglichen, definiert das VirtualMedia-Modell eine Anzahl von *semantischen Äquivalenzrelationen*, welche auf verschiedenen nicht-funktionalen Eigenschaften der Medienfilter beruhen. Die drei grundlegenden Äquivalenzrelationen basieren hierbei auf den Eigenschaften Neutralität (bezüglich einer bestimmten Medieneigenschaft wie dem Format, der Qualität oder dem Inhalt), Reversibilität und Vertauschbarkeit (d. h. semantische Unabhängigkeit). Durch eine Generalisierung dieser Konzepte kann dann auch die Semantik der Medienkomposition und -dekomposition mithilfe von Äquivalenrelationen beschrieben werden.

Ein Filtergraph kann Filterknoten enthalten, die sich z. B. wegen der Verwendung leerer Signaturen nicht eindeutig auf einen konkreten, instanzierbaren Medienfilter abbilden lassen. Diese Knoten können mithilfe einer Äquivalenzrelation, die als *semantische Assimilation* bezeichnet wird, konkretisiert werden. Dabei wird in einer Metadatenbank mit den Beschreibungen abstrakter und konkreter Medienfilter nach einer Filtersignatur gesucht, die zu einem gegebenen Filterknoten semantisch äquivalent ist. Die gefundene Filtersignatur wird dann zusammen mit den zugehörigen Mediensignaturen für die Ein- und Ausgänge des Filters in den Graphen eingefügt. Als Nebeneffekt können sich hierbei sowohl vorhandene magische Kanäle auflösen als auch neue entstehen. Etwas vereinfacht ausgedrückt, beschreibt die semantische Assimilation also die Beziehung zwischen abstrakten Medienoperationen (wie sie gewöhnlich in Anfragegraphen spezifiziert werden) und Komponenten, die diese Operationen für konkrete Medienobjekttypen implementieren.

Ein weiterer, insbesondere auch für die Optimierung wichtiger Aspekt ist die Materialisierungsverwaltung, welche die Nutzung verschiedener interner Repräsentationen der Medienobjekte erlaubt. Dies und viele weitere Details sind in [10] beschrieben. Ein ausführlicheres Beispiel zu den Einsatzmöglichkeiten des VirtualMedia-Modells im Reuse-Repository folgt in Abschnitt 5. Zuvor soll je-

doch ein Überblick über die Systemarchitektur des Reuse-Repository mit integriertem VirtualMedia-Modell gegeben werden.

4 Die Architektur eines integrierten Ansatzes

Die Integration des VirtualMedia-Modells in das Reuse-Repository ermöglicht eine höchst flexible, sich an die Benutzerpräferenzen anpassende Verwaltung und Bereitstellung von Dokumenten, die Erfahrungsdaten beinhalten. An die Stelle der bisherigen uneinheitlichen, anwendungsseitigen Konvertierung und Nachbearbeitung der Dokumente tritt hierbei eine serverseitige Generierung externer Dokumente, für deren Steuerung die Anwendungen keinerlei Wissen über die interne Repräsentation der gespeicherten Erfahrung mehr benötigen. Zu beachten ist dabei lediglich, dass das VirtualMedia-Modell nur den Rahmen für die automatisierte Anwendung von Medienfiltern sowie die dazu gehörigen Anfragemechanismen liefert. Die erforderlichen Filter für die Dokumenterzeugung, -konvertierung und -verarbeitung, wie z. B. Filter für die korrekte Interpretation von RDX-Werten oder den Check-in-Prozess, müssen bei einer Implementierung noch zusätzlich bereit gestellt werden.

Die Grundlage für die Integration des VirtualMedia-Modells bildet die bereits im Rahmen der Integration von Produktdatenmodellen [7] weiter entwickelte Architektur. Einen guten Ausgangspunkt für die neue Architektur bietet außerdem die Abbildung 2: Offensichtlich kann VirtualMedia für die Realisierung des universellen Formatgenerators sowie des Check-in-Filters eingesetzt werden. Ferner bietet es erweiterte Konzepte für die Verwaltung der zusätzlichen (redundanten) Repräsentationen.

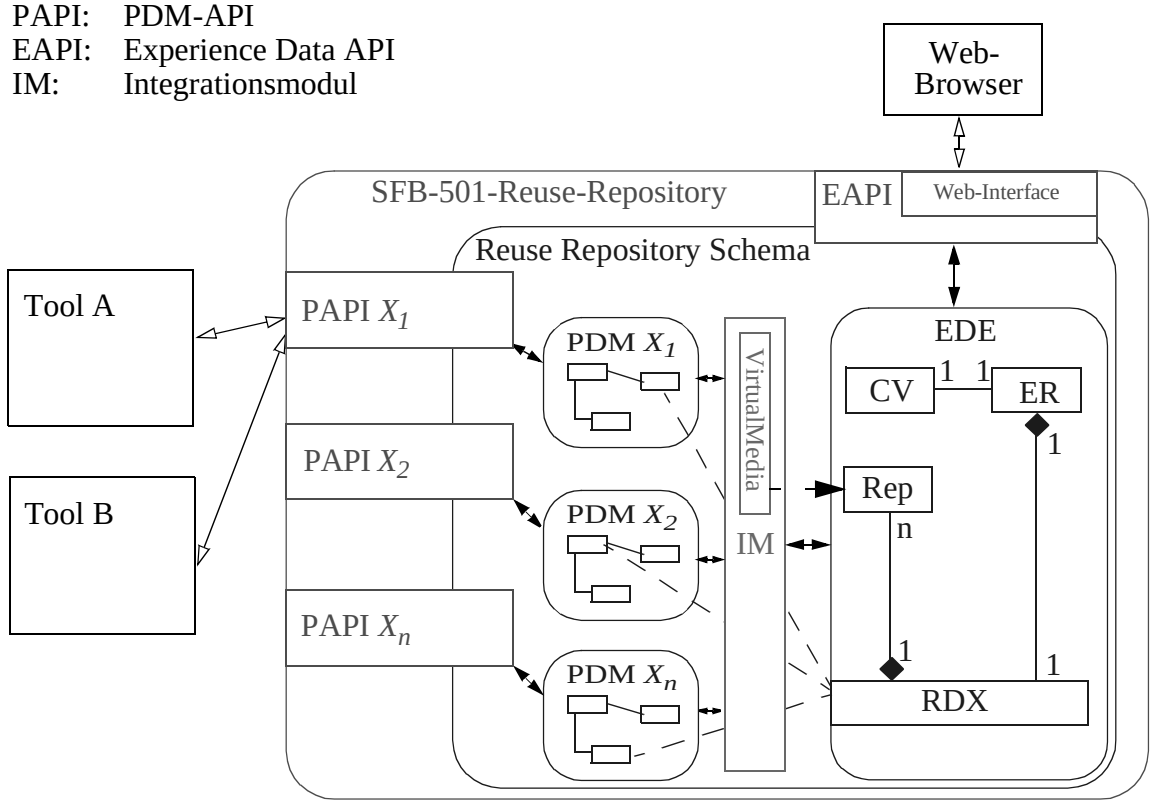

Abbildung 5: Logische Systemarchitektur des integrierten Reuse-Repository

Abbildung 5 zeigt die resultierende logische Systemarchitektur. Es gibt wie bisher zwei prinzipiell unterschiedliche Zugriffsschnittstellen für das Reuse-Repository. Das ist zum einen der Zugriff (eines Web-Browsers) über das Web-Interface. Das Web-Interface ist Teil der *Experience Data API* (EAPI) des Reuse-Repository, über die auf die Erfahrungsdateneinträge (EDE) zugegriffen werden kann. In diesem auch als Erfahrungs-Sicht bezeichneten Teil des Reuse-Repository werden im Wesentlichen EDE verwaltet, die aus einem CV, einem ER und einer Menge von Repräsentationen bestehen. Die PDM-Integration führt dazu, dass daneben noch (beliebig viele) Produktdatenmodelle (PDM) verwaltet werden können. Dementsprechend kann auf die Produktdaten über je eine spezifische *PDM-API* (PAPI) direkt von externen Werkzeugen aus zugegriffen werden (Werkzeug-Sicht).

Anders als in früheren nicht-integrierten Reuse-Repository-Architekturen werden dem ER die Repräsentationen nicht mehr direkt zugeordnet. Stattdessen besitzt der ER genau eine RDX, die maximal eine primäre Repräsentation und beliebig viele sekundäre Repräsentationen enthält. Eine primäre Repräsentation ist eine nicht-leere Menge von Produkten, die gemäß dem extensional orientierten Ansatz durch Beziehungen zwischen RDX und geeigneten PDM-Entities modelliert wird (gestrichelte Linien in Abbildung 5). Ein sog. Integrationsmodul (IM) überwacht alle über diese Beziehungen direkt oder indirekt miteinander verknüpften Objekte. Soll eines dieser Objekte geändert oder gelöscht werden, ist es Aufgabe des IM, Maßnahmen zur Erhaltung der semantischen Konsistenz einzuleiten [7]. Im IM wird nun das VirtualMedia-Modell realisiert, das es ermöglicht, die sekundären Repräsentationen dynamisch entsprechend der Wünsche und Bedürfnisse des jeweiligen Anwenders zu generieren (universeller Formatgenerator). Werden dem IM entsprechende Check-in-Filter zur Verfügung gestellt, ist es auch möglich, Änderungen, die ein Benutzer extern an einer sekundären Repräsentation vorgenommen hat, in die entsprechende primäre Repräsentation im zugehörigen PDM zu propagieren. Im Allgemeinen wird dies erfordern, dass die externe Repräsentation Schemainformation beinhaltet, wie es beispielsweise bei XML-Dokumenten der Fall ist.

Für die technische Realisierung der Integration des VirtualMedia-Modells mit einem ORDBMS gibt es mehrere Ansätze, auf die hier nicht detailliert eingegangen wird:

- *Direkte Integration* unter Verwendung der Erweiterungsmechanismen des ORDBMS [9].
- *Middleware-Ansatz*, d. h., mehrere unterschiedliche Server werden über eine darüber liegende Middleware-Schicht integriert [10].
- *Kompensator-Ansatz*, d. h., der VirtualMedia-Server ist ein externes Subsystem zum ORDBMS, die Anpassung der Schnittstellen erfolgt durch einen sog. Kompensator [6,10].

5 Beispiel einer Anwendung des integrierten Ansatzes

Abschließend soll nun in einem Beispiel sowohl die Anwendung von VirtualMedia als Teil des Reuse-Repository als auch die generelle Arbeitsweise des VirtualMedia-Modells demonstriert werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE VMDESC SYSTEM "vmd.dtd">
<VMDESC>
   <SOURCE>
      <MOID ALIAS="ProjectDocumentation" EXT_REF="EDB/OWS/EDE/4711/">
         <SIGNATURE>
            <PROPERTY NAME="MAINTYPE" CLASS="typespec"><IT>Text</IT></PROPERTY>
         </SIGNATURE>
      </MOID>
   </SOURCE>
   <VIRTUAL NAME="Kurzdoku" MODE="by_value">
      <SIGNATURE>
         <PROPERTY NAME="MAINTYPE" CLASS="typespec"><IT>Text</IT></PROPERTY>
         <PROPERTY NAME="SUBTYPE" CLASS="typespec"><IT>HTML</IT></PROPERTY>
         <PROPERTY NAME="ENCODING" CLASS="typespec"><IT>UTF-8</IT></PROPERTY>
         <PROPERTY NAME="Lang" CLASS="content"><IT>DE</IT></PROPERTY>
      </SIGNATURE>
      <TRANSFORMATION NAME="Summary">
         <OPERATION SEMANTICS="summarize">
            <INPUT ALIAS="in1" REF="ProjectDocumentation"/>
            <PARAM NAME="Squeeze_to" VALUE="10%"/>
         </OPERATION>
      </TRANSFORMATION>
   </VIRTUAL>
</VMDESC>
```

Abbildung 6: Beispielanfrage in VMML

Dazu betrachten wir ein Szenario, in dem ein Anwender des Reuse-Repository über das Web-Interface – beispielsweise unter Verwendung der Suchfunktionalität – zu einem EDE gelangt ist. Von diesem EDE, im Beispiel eine Projektdokumentation, möchte der Anwender die Repräsentation in geeignetem Format bereitgestellt bekommen. Für die Spezifiaktion solcher Benutzeranfragen stellt VirtualMedia die XML-basierte Anfragesprache VMML (VirtualMedia Markup Language) zur Verfügung. Abbildung 6 zeigt eine Beispielanfrage in VMML für den Zugriff auf die im Reuse-Repository verwaltete Projektdokumentation. Im SOURCE-Teil der Anfrage wird der EDE mit dem angenommenen Primärschlüssel „4711“ referenziert, der das gesuchte Erfahrungselement beschreibt. Im VIRTUAL-Teil der Anfrage wird wiederum die gewünschte externe Repräsentation beschrieben. Das zurückzuliefernde Dokument erhält zu diesem Zweck zunächst eine Signatur, die in diesem Beispiel das Format mithilfe der hierfür vorgesehenen Eigenschaften genau festlegt sowie eine inhaltsbezogene Eigenschaft beinhaltet, nämlich, dass die Dokumentsprache Deutsch sein soll. Darüber hinaus wird eine Operation auf dem Dokument spezifiziert, welche durch automatische Zusammenfassung eine Verkürzung des Dokuments auf 10% seiner eigentlichen Länge bewirken soll. Teile dieser Spezifikation können für den Benutzer bereits im Reuse-Repository verwaltet werden (z. B. gewünschte Sprache), andere können über das Web-Interface spezifiziert werden (z. B. Verkürzung des Dokuments auf 10%), sodass das XML-Dokument aus Abbildung 6 nicht vom

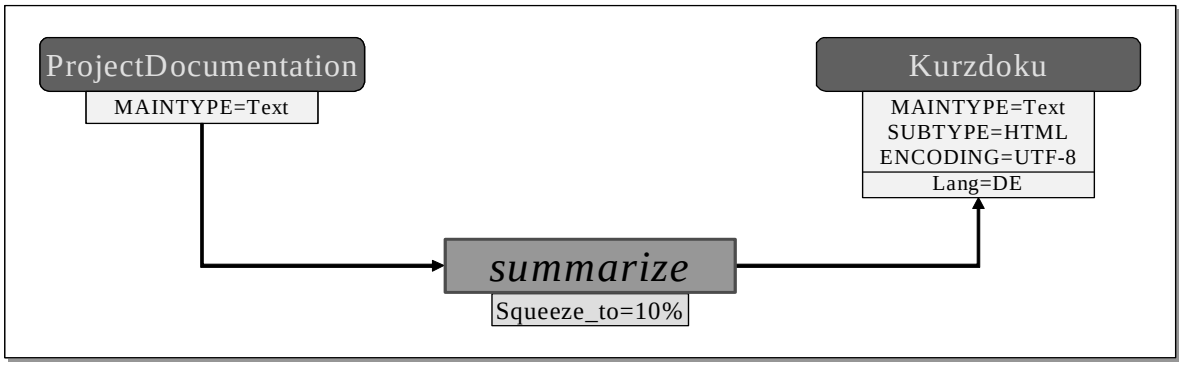

Abbildung 7: Äquivalenter Anfragegraph

Anwender selbst bereitgestellt werden muss, sondern vom Reuse-Repository generiert wird.

Die VMML-Anfrage wird von VirtualMedia zuerst in einen äquivalenten Anfragegraphen übersetzt (s. Abbildung 7). Um diesen Graphen gemäß dem Modell weiter verarbeiten zu können, muss er zunächst mit den Materialisierungsgraphen für die Quellobjekte verknüpft werden. Die Materialisierungsgraphen sind im Aufbau den in Abschnitt 3 eingeführten Filtergraphen sehr ähnlich und beschreiben die Beziehung zwischen einem logischen Medienobjekt (auf das ein Anwender Bezug nehmen kann) und seiner internen Repräsentation [10].

Abbildung 8 zeigt einen möglichen Materialisierungsgraphen für das in der Beispielanfrage referenzierte Quellobjekt. Wir nehmen hierbei an, dass die primäre Materialisierung mithilfe eines PDM verwaltet wird[3] und dass bereits eine sekundäre Materialisierung[4] in Form eines Postscript-Dokuments generiert wurde.

Abbildung 9 zeigt den Anfragegraphen nach der Durchführung der Verknüpfung mit dem Materialisierungsgraphen, wobei aus Gründen der Übersicht nicht der ganze Materialisierungsgraph übernommen wurde, sondern lediglich die beiden alternativen Signaturen für das Quellobjekt gezeigt werden. Die Kante ist dabei mit derjenigen Signatur verbunden, die zu einer geringeren Signaturdifferenz[5] zu der Eingangssignatur der Operation „summarize“ führt. Betrachten wir nun die zweite Kante, an der die Signaturdifferenz wegen der unterschiedlichen SUBTYPE-Eigenschaften ungleich Null ist. Dies kann mithilfe einer Äquivalenzrelation, die das Einfügen eines inhaltsneutralen Formatfilters erlaubt, korrigiert werden (s. Abbildung 10).

3. Sie entspricht somit der primären Repräsentation (RDX) im konzeptionellen Schema, vgl. Abbildung 2.
4. Entspricht einer optionalen Repräsentation (Rep) im konzeptionellen Schema, vgl. Abbildung 2.
5. Die Signaturdifferenz ist als eine diskrete Distanzfunktion zwischen zwei beliebigen Mediensignaturen definiert, basierend auf einer Klassifikation der Eigenschaften (Properties), aus denen sich die Signaturen zusammen setzen, vgl. Abschnitt 3.3.

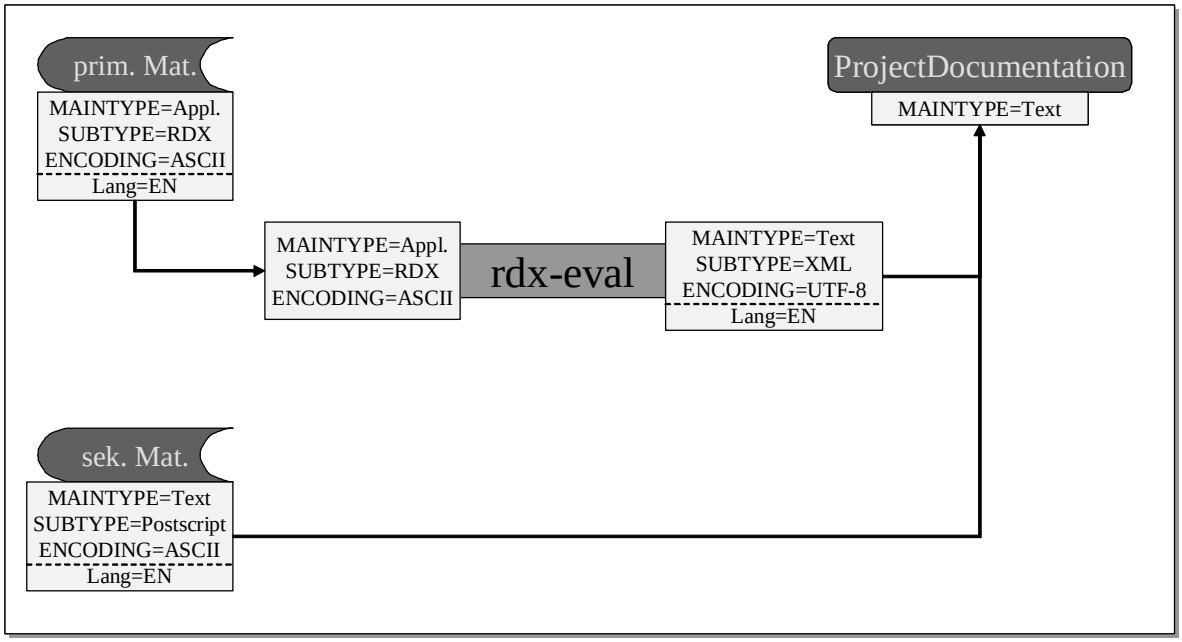

Abbildung 8: Materialisierungsgraph für das Beispiel

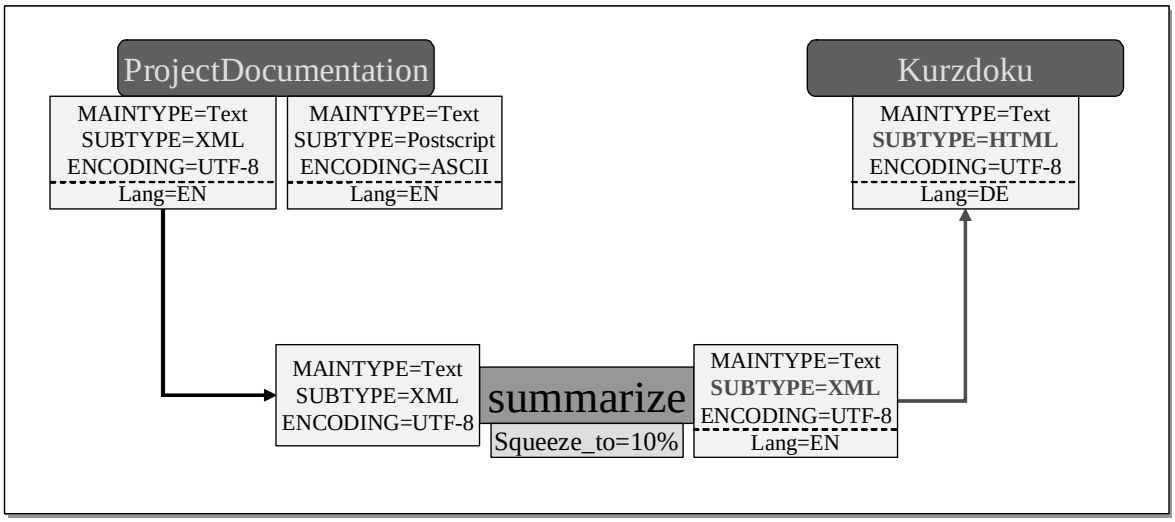

Abbildung 9: Anfragegraph nach der Verknüpfung mit dem Materialisierungsgraphen

Mit dem in Abbildung 10 dargestellten Graphen wenden wir uns nun der Lang-Eigenschaft (Dokumentsprache) zu. Dabei handelt es sich um eine Inhaltseigenschaft, die in der Signatur des Quellobjekts den Wert „EN“ besitzt. Von den beiden Filtern wird diese Eigenschaft jeweils nur weitergereicht, was dadurch angedeutet wird, dass die Eigenschaft nur in den Ausgangssignaturen erscheint. Dies führt dazu, dass es an der letzten Kante zu einem Konflikt kommt, da für das Zielobjekt

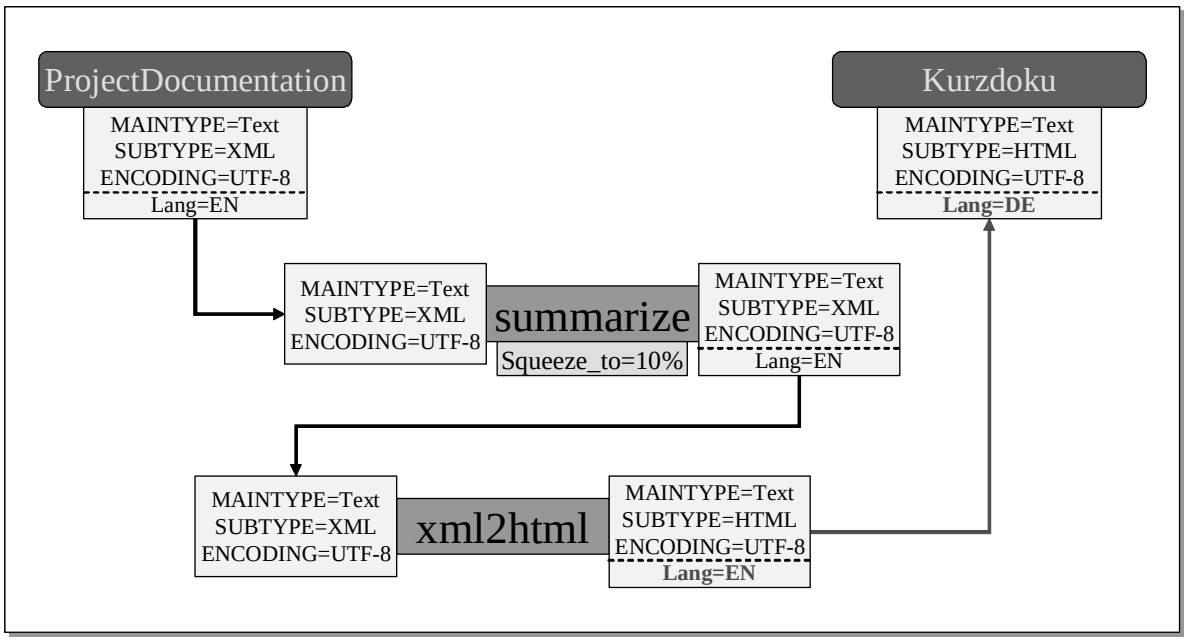

Abbildung 10: Anfragegraph nach dem Einfügen eines Formatfilters

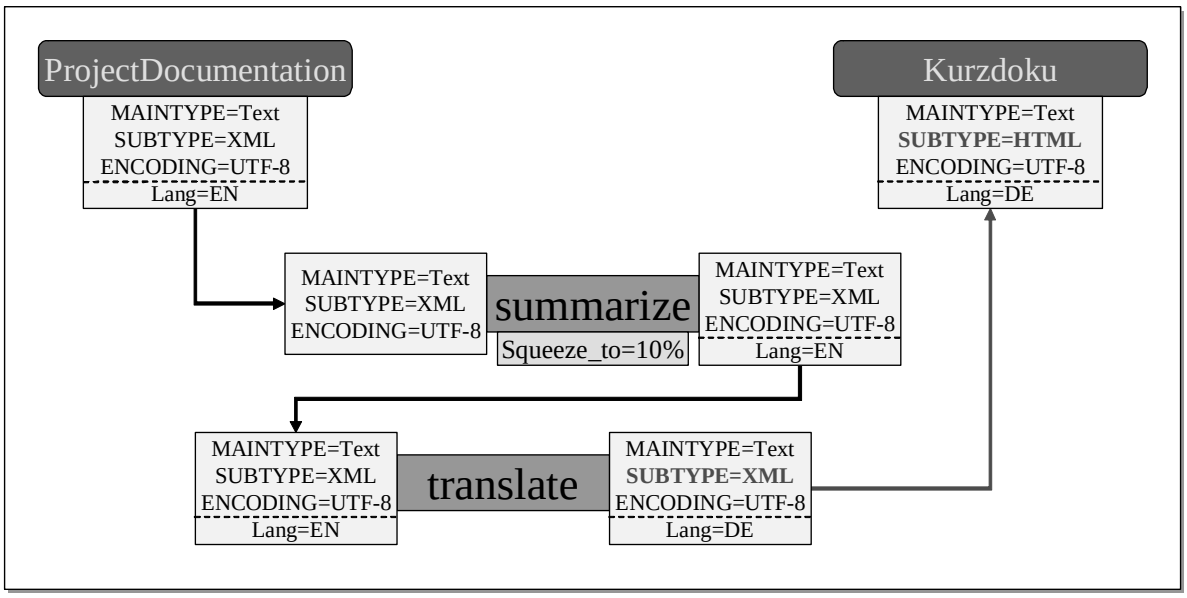

Abbildung 11: Anfragegraph nach dem Einfügen eines Inhaltsfilters

in der Anfrage der Wert „DE“ für diese Eigenschaft gefordert wird. Um das Beispiel nicht zu geradlinig wirken zu lassen, nehmen wir nun an, dass es keinen passenden Filter gibt, so dass sich der Konflikt in Abbildung 10 nicht wie zuvor durch Einfügen eines weiteren Filters auflösen lässt. In dieser Situation hängt es vom Algorithmus ab, was geschieht. Bisher ist im Beispiel eine einfache Greedy-Strategie zum Einsatz gekommen, welche nun nicht mehr weiterführt, da sie keine Rückschritte zulässt, und folglich den Graphen in Abbildung 10 als Endergebnis zurück liefern würde.

In [10] werden verschiedene heuristische Suchverfahren für die Anfrageverarbeitung im Virtual-Media-Modell diskutiert, darunter auch randomisierende Verfahren wie *Simulated Annealing* und

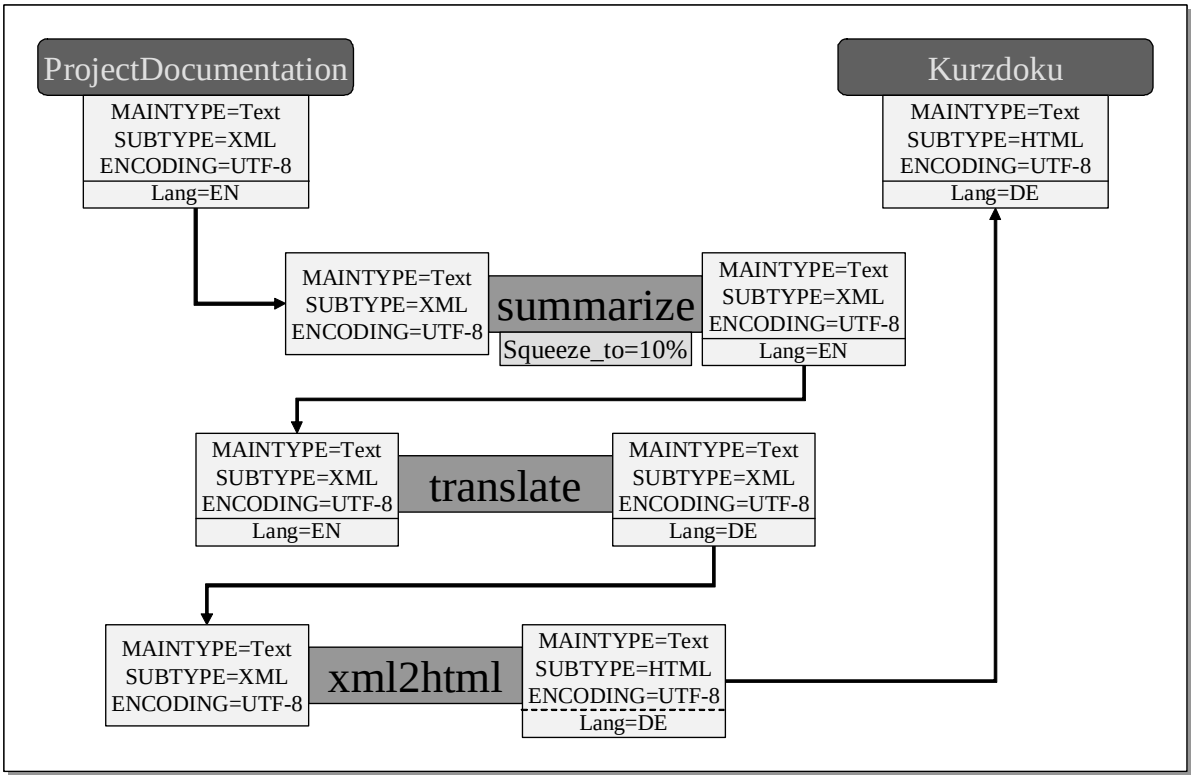

Abbildung 12: Endergebnis der Anfrageverarbeitung

Evolutionäre Algorithmen. Der Vorteil der letzteren beiden ist, dass sie nicht in lokalen Optima stecken bleiben, wenn diese noch keine befriedigende Lösung ergeben. Evolutionäre Algorithmen verfolgen beispielsweise immer mehrere potenzielle Lösungswege parallel. So kann in dem Graphen von Abbildung 9 neben einer Lösung für den Formatkonflikt parallel auch eine Lösung für den Inhaltskonflikt bezüglich der Lang-Eigenschaft gesucht werden. Unter der Annahme, dass es einen Übersetzungsfilter für die gegebene Mediensignatur gibt, führt dies zu dem in Abbildung 11 dargestellten alternativen Lösungsgraphen.

Wie der weiterhin vorhandene Formatkonflikt in Abbildung 11 gelöst wird, ist bereits aus Abbildung 10 bekannt. Dieser letzte Schritt führt schließlich zur endgültigen Lösung der Anfrage, die in Abbildung 12 dargestellt ist.

Dass die optimale Lösung (s. Abbildung 12) in diesem Beispiel nicht durch die Greedy-Strategie erreicht wird, liegt daran, dass bei der Definition der Signaturdifferenz den Formateigenschaften ein höheres Gewicht zukommt als den Qualitäts- und Inhaltseigenschaften. Im Normalfall ist dies vernünftig, da ohne eine Auflösung aller Typ- und Formatkonflikte überhaupt keine ausführbare Lösung zustande käme. Dem gegenüber können Lösungen mit ungelösten Qualitäts- oder Inhaltskonflikten durchaus ausführbar sein und somit auch vom Anwender akzeptiert werden, was auch für die potenzielle, aber nicht optimale Lösung in Abbildung 10 zutrifft. Generell ist zu beobachten, dass besonders die implizite Spezifikation von inhaltsverändernden Operationen, wie im Beispiel die Deklaration der Lang-Eigenschaft, die Greedy-Strategie zum Scheitern bringen kann, randomisierende

Algorithmen hingegen i. d. R. nicht (wenn eine Lösung existiert). Würde die Übersetzungsoperation explizit, d. h. im TRANSFORMATION-Abschnitt der Anfrage spezifiziert, dann fände auch die Greedy-Strategie die optimale Lösung.

Die Diskussion dieses Beispiels zeigt, wie das VirtualMedia-Modell die Generierung externer Dokumente aus dem Reuse-Repository heraus unterstützen kann, ohne dass die anfragende Anwendung sich darum kümmern muss, wie die darzustellenden Erfahrungsdaten intern repräsentiert sind. Dieser Ansatz ist somit ohne Zweifel für die Realisierung eines universellen Formatgenerators geeignet. Darüber hinaus erlaubt er ohne Weiteres auch die Integration und Anwendung inhaltsmanipulierender Medienfilter sowie qualitätsanpassender Filter, wie sie beispielsweise für die Unterstützung kleiner, mobiler Endgeräte benötigt werden. Liegt die Priorität bei der Formatanpassung, genügt ein einfach zu implementierender Greedy-Algorithmus für die VirtualMedia-Anfrageverarbeitung. Sollen auch komplexere Anfragen optimal bearbeitet werden, sind allerdings aufwändiger zu realisierende Algorithmen (z. B. Evolutionäre Algorithmen) oder solche mit nicht-polynomialer Laufzeit (z. B. erschöpfende Suche) erforderlich [10].

6 Zusammenfassung und Ausblick

Die bisherige prototypische Realisierung des für den SFB 501 entwickelten Reuse-Repository unterstützt noch keine Datenunabhängigkeit für die die eigentlichen Erfahrungsdaten beinhaltenden Dokumente. Deren Formate sind jeweils abhängig von dem Anwender, der die Daten in das Reuse-Repository einstellt. Zwar können mehrere solche Repräsentationen gleichen Inhalts, aber unterschiedlichen Formats existieren, jedoch muss deren Erzeugung und Pflege manuell von außen erfolgen, so dass die inhaltliche Äquivalenz durch das Reuse-Repository nicht gewährleistet ist.

Da Software-Entwicklungswerkzeuge oftmals nicht standardisierte bzw. wenig gebräuchliche Datenformate einsetzen, ist es generell erforderlich, dass das Reuse-Repository alternative Repräsentationen insbesondere für diejenigen Anwender bereit stellt, die nur über Werkzeuge zur Darstellung und Bearbeitung von Standard-Datenformaten wie z. B. PDF verfügen.

Das VirtualMedia-Modell wurde unabhängig vom Reuse-Repository zu dem Zweck entwickelt, Datenunabhängigkeit für große, multimediale Objekte zu realisieren. Es bietet mithilfe eines leicht zu benutzenden Anfragemechanismus' die Möglichkeit, externe Repräsentationen dynamisch zu erzeugen, ohne dass der Anwendung die entsprechende interne Repräsentation bekannt sein muss. Zur Generierung von Plänen, die die gewünschte externe Repräsentation mithilfe intern verfügbarer Konvertierungs- und sonstiger Funktionen realisieren, definiert das Modell einen Satz von formalen Regeln für einen heuristischen Anfrageverarbeitungsalgorithmus. Darüber hinaus beinhaltet es Konzepte für die weitgehend automatisierte Erzeugung und Verwaltung alternativer interner Repräsentationen, die für die Optimierung verwendet werden.

Das VirtualMedia-Modell bietet somit prinzipiell die dem Reuse-Repository noch fehlende Funktionalität für eine flexible, adaptive Bereitstellung von Erfahrungsdaten. Wir haben in diesem Bericht untersucht und dargelegt, wie dieses Modell in die Architektur des Reuse-Repository zu integrieren ist. Anhand eines ausführlichen Beispiels wurde anschließend verdeutlicht, wie eine Anfrage zur Be-

reitstellung eines externen Dokuments verarbeitet werden würde und welche Auswirkungen die Heuristiken bzw. die Art des Anfrageverarbeitungsalgorithmus' hierauf haben können.

Ein auf dem VirtualMedia-Modell basierendes Modul lässt sich ohne größere Einschnitte im Zuge der bereits geplanten Erweiterung des Reuse-Repository zur direkten Unterstützung von Produktdatenmodellen in die logische Systemarchitektur integrieren. Die Integration erfordert dabei nicht zwingend eine sehr enge Kopplung zwischen dem Reuse-Repository und dem VirtualMedia-Modul. Denkbar sind z. B. auch Lösungen, bei denen die VirtualMedia-Funktionalität in einer Middleware-Schicht oberhalb des Reuse-Repository untergebracht ist.

Eine zukünftige Integration der VirtualMedia-Funktionalität in das Reuse-Repository erscheint somit sehr vorteilhaft, da einerseits die bisher fehlende Datenunabhängigkeit für die Erfahrungsdaten in einer für die Anwendungen komfortablen Weise hinzugefügt wird, während andererseits nur relativ wenige Auswirkungen auf den bisherigen und zukünftigen Entwurf des Reuse-Repository zu befürchten sind. Dennoch ist derzeit eine Realisierung dieser Vision wegen des bevorstehenden Abschlusses der Projekte des SFB 501 nicht mehr geplant.

7 Literatur

[1] Basili, V. R., Rombach, H. D.: Support for comprehensive reuse. In *IEE Software Engineering Journal*, 6(5):303–316, September 1991.

[2] Candan, K. S., Subrahmanian, V. S., Venkat Rangan, P.: Towards a Theory of Collaborative Multimedia. In: Proc. IEEE Intl. Conf. on Multimedia Computing and Systems (Hiroshima, Japan, Juni 1996), 1996, S. 279-282.

[3] Feldmann, R. L., Geppert, B., Mahnke, W., Ritter, N., Rößler, F.: An ORDBMS-based Reuse Repository Supporting the Quality Improvement Paradigm - Exemplified by the SDL-Pattern Approach, in: Proc. TOOLS USA 2000, 34th International Conference & Exhibition, Santa Barbara, CA, 30. Juli - 3. August 2000, S. 125-136.

[4] Feldmann, R. L., Geppert, B., Mahnke, W., Ritter, N., Rößler, F.: The ORDBMS-based SFB 501 Experience Base - Exemplified by the SDL-Pattern Approach, SFB-Report 08/99, SFB 501, Universität Kaiserslautern, 1999.

[5] Haustein, M.: Ähnlichkeitssuche in objekt-relationalen Datenbanksystemen, Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern, Mai 2002.

[6] Lindner, W., Marder, U.: Advancing Query Capabilities for Digital Libraries. Manuskript in Vorbereitung, 2002.

[7] Mahnke, W., Marder, U., Queins, S., Ritter, N., Schürmann, B.: Integrierte Produkt- und Erfahrungsdatenverwaltung, SFB-Report 08/2000, SFB 501, Universität Kaiserslautern, Aug. 2000.

[8] Mahnke, W. Ritter, N.: The ORDB-based SFB-501-Reuse-Repository, in: Proc. 8th Int. Conf. on Extending Database Technology (EDBT'2002), Software Demonstration Session, Prag, März 2002, S. 745-748.

[9] Marder, U.: Medienspezifische Datentypen für objekt-relationale DBMS: Abstraktionen und Konzepte, in: Tagungsband der GI-Fachtagung 'Datenbanksysteme in Büro, Technik und Wissenschaft' (BTW'99), A. Buchmann (Hrsg.), Informatik aktuell, Freiburg, März 1999, Springer-Verlag, S. 210-231.

[10] Marder, U.: Multimedia-Metacomputing in Web-basierten multimedialen Informationssystemen, Dissertation, Fachbereich Informatik, Universität Kaiserslautern, Dez. 2002.

[11] Marder, U., Kovse, J.: Multimedia Metacomputing. In: Proc. 7th Intl. Workshop on Multimedia Information Systems, MIS 2001, November 2001, S. 173-182.

[12] Ritter, N., Steiert, H.-P., Mahnke, W., Feldmann, R. L.: An Object-Relational SE-Repository with Generated Services, in: Proc. Managing Information Technology Resources in Organizations in the Next Millenium (Computer-Aided Software Engineering Track of IRMA'99), IDEA Group Publ., Mai 1999, S. 986-990.

[13] SFB-501 Reuse Repository, https://edb.sfb501.uni-k.de.