Ein Ansatz zur formatneutralen Verwaltung von Metadaten in komponentenorientierten Softwareprozessen

Hans-Jörg Happel[*)], Axel Korthaus[†)], Stefan Seedorf[†)] und Peter Tomczyk[*)]

[*)] FZI Forschungszentrum Informatik
Fachbereich Information Process Engineering (IPE)
Haid-und-Neu-Str. 10-14
D-76137 Karlsruhe
{happel|tomczyk}@fzi.de

[†)] Universität Mannheim
Lehrstuhl für Wirtschaftsinformatik III
Schloss, L 5,5
68131 Mannheim
{korthaus|seedorf}@wifo.uni-mannheim.de

Abstract:

Der Wiederverwendung von Lösungsbausteinen, wie z.B. fertiger Softwarekomponenten oder anderer Softwareartefakte, wird im Rahmen der Softwareentwicklung eine hohe ökonomische Bedeutung beigemessen. Eine notwendige Voraussetzung für praktikable Wiederverwendungsprozesse bilden Komponentenspeicher. Die existierenden Ansätze zur Komponentenspeicherung sind jedoch häufig an Vorbedingungen bezüglich bestimmter Beschreibungsformate, Vorgehensmodelle oder eines bestimmten Wiederverwendungsansatzes geknüpft. Der inhärenten Heterogenität von Softwareartefakten wird dabei nur selten Rechnung getragen, was die allgemeine Akzeptanz von Komponentenspeichern beeinträchtigt. Das hier vorgestellte Metaschema für einen generischen Komponentenspeicher adressiert dieses Problem durch die Unterstützung der formatneutralen Beschreibung, strukturierten Speicherung und des Wiederauffindens von Softwareartefakten. Es unterscheidet zwischen den drei Dimensionen Artefakttyp, Format und Informationsaspekt. Im Rahmen der Arbeit wird aufgezeigt, wie das Metaschema zur flexiblen Beschreibung von Softwarekomponenten eingesetzt werden kann. Dazu werden die Architektur des Komponentenspeichers und ein Anfragemechanismus vorgestellt. Abschließend erfolgt eine Bewertung des präsentierten Ansatzes.

1 Problemstellung

Komponentenorientierte Ansätze in der Softwareentwicklung dienen zur Steigerung der Wiederverwendung von Softwarebausteinen sowie zur effizienten Gestaltung von Entwicklungsprozessen (siehe z.B. [CD00, ABB01, Ko01]). Im Allgemeinen wird unter dem Begriff der Komponente eine in sich abgeschlossene Kompositionseinheit mit fest definierten Schnittstellen und ausschließlich expliziten Kontextabhängigkeiten verstanden [Sz02]. Komponenten setzen sich aus einer Vielzahl von Softwareartefakten zusammen, die sowohl zu ihrer Spezifikation als auch zu ihrer Realisierung dienen. Umgekehrt können sie ihrerseits wieder als Softwareartefakte aufgefasst werden.

Obwohl komponentenorientierte Architekturen verbreitet sind, ist die Wiederverwendung von Softwarekomponenten und –artefakten aus inner- und überbetrieblichen Komponentenspeichern bis heute hinter den ursprünglichen Erwartungen zurückgeblieben [Sz02, FK05]. Neben aufbau- und ablauforganisatorischen Fragestellungen [CW94] ist auf technischer Ebene insbesondere die Repräsentation, also die geeignete Abstraktion der Softwareartefakte ein zentrales Problem und stellt somit eine essenzielle Gestaltungsdimension für Wiederverwendung dar [BR89, Kr92]. Sowohl für das Auffinden als auch für die Anpassung und Integration ist eine bedarfsorientierte Beschreibung der Komponente von entscheidender Bedeutung. Zu diesem Zweck müssen die Softwareartefakte mit Metadaten angereichert und in einen logischen Zusammenhang gestellt werden. Eine dabei bisher noch unbeantwortete Frage ist, wie ein Metaschema so gestaltet werden kann, dass es von konkreten Beschreibungen abstrahiert und die Aufgaben eines Komponentenspeichers bestmöglich unterstützt.

Im Rahmen dieser Arbeit wird deshalb ein flexibles Metaschema zur Komponentenbeschreibung für den Aufbau eines heterogenen Komponentenspeichers entworfen. Dazu werden zunächst bestehende Ansätze zur Repräsentation von Softwarekomponenten analysiert. Im Anschluss werden das Metaschema abgeleitet und die technische Realisierung des Komponentenspeichers in Form eines Prototyps erörtert. Schließlich werden die Ergebnisse diskutiert und die sich daraus ergebenden Möglichkeiten für einen umfassenden Wiederverwendungsansatz aufgezeigt.

2 Ansätze zur Repräsentation von Softwarekomponenten

Die Aufgabe eines Komponentenspeichers (*Component Repository*, *Library*) als wichtigem Bestandteil einer Infrastruktur zur Wiederverwendung von Softwarekomponenten besteht darin, die Verfügbarkeit, Auffindbarkeit und Verständlichkeit von Komponenten zu verbessern [FK05]. Komponentenspeicher lassen sich somit eindeutig von Repositorys im Sinne des Konfigurationsmanagements [IEEE04] abgrenzen, deren primäre Funktion in der Versionierung klar definierter Artefakte liegt.

Mili et al. [MMM98] unterscheiden drei zentrale Abstraktionsebenen von Komponentenspeichern, die in Abbildung 1 dargestellt werden: der *Nutzen* für die Anwender, das *Artefakt*, das den gewünschten Nutzen liefern soll, sowie die *Repräsentation* des Artefakts. Die ersten beiden Abstraktionsebenen betreffen die grundlegenden Operationen

eines Komponentenspeichers: der Abruf von Artefakten durch einen Komponentennutzer sowie die Speicherung von Artefakten durch einen Komponentenverwalter.

Weil es sich hierbei um Aktivitäten handelt, die getrennt voneinander durchgeführt werden, fällt dem Komponentenspeicher eine Mittlerrolle zu. Diese Vermittlung geschieht auf Basis der Repräsentation von Artefakten. Sie enthält Informationen über ein Artefakt (d.h., Meta-Informationen), die mit den Anfragen des Nutzers abgeglichen werden. Eine eigenständige Repräsentation wird deshalb benötigt, weil Artefakte sich entweder aufgrund ihrer Natur nicht in die Sprache des Nutzers abbilden lassen, oder weil ein direkter Zugriff aufgrund ihrer Komplexität nicht praktikabel ist [MMM98].

Verbreitete Ansätze zur Abbildung eines Artefakts auf eine Repräsentation sind Indizierung und Klassifikation [VZJ03, FK05]. Bei der Klassifikation definiert der Verwalter des Komponentenspeichers Stichworte zu einer gegebenen Menge so genannter Facetten [Pri91], die ein Artefakt beschreiben. Dagegen erfolgt eine Indizierung automatisch. Mili et al. unterscheiden sechs grobe Klassen von Komponentenspeichern, die sich primär im Modus der Abbildung von Artefakten auf eine assoziierte Repräsentation unterscheiden. Dabei fällt auf, dass die meisten existierenden Komponentenspeicher zwar eine Vielzahl von Artefakttypen vorsehen, aber lediglich ein bestimmtes Repräsentationsformat und eine bestimmte Nutzerrolle voraussetzen.

Viele konventionelle Komponentenspeicher vernachlässigen demzufolge die in der Praxis relevante Vielfalt von Sichten (Informationsaspekten) und Repräsentationen (vgl. [VZJ03]). Dafür sind aus historischer Sicht mehrere Gründe denkbar. So hat in jüngerer Zeit eine Ausdifferenzierung von Softwareprozessen stattgefunden, die zu einer Zunahme spezialisierter Rollen im Entwicklungsprozess geführt hat (vgl. [Kr00]). Diese Rollen haben spezifische, sich unterscheidende Informationsbedarfe und sind an verschiedenen Stufen des Entwicklungsprozesses beteiligt.

Gleichzeitig fällt im Laufe des Entwicklungsprozesses eine größere Menge (semi-) strukturierter Artefakte (z.B. Testfälle, Modelle, Code etc.) an. Insbesondere ist in den vergangenen Jahren die Bedeutung, die Modellen in der Softwareentwicklung beigemessen wird, gestiegen. Dies wird durch die Entwicklung ausgehend von objektorientierten Analyse- und Designmethoden bis hin zu aktuellen Ansätzen der modellgetriebenen Softwareentwicklung belegt (siehe z.B. [HK04]).

Mit einer größeren Anzahl unterschiedlicher Nutzer und semi-strukturierter Daten ergeben sich neue Anforderungen an die Gestaltung von Komponentenspeichern. Die Arbeit

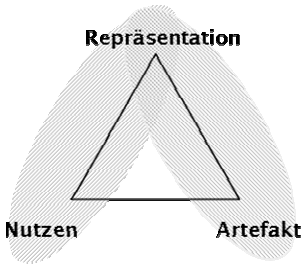

Abbildung 1: Abstraktionsebenen eines Komponentenspeichers (angelehnt an [MMM98])

von Vitharana et al. [VZJ03] trägt diesen Anforderungen teilweise Rechnung, indem sie verschiedene Nutzergruppen und die Repräsentation strukturierter und semi-strukturierter Daten unterscheidet. Weiteres Wissen über die Art der semi-strukturierten Daten sowie mögliche Abfragearten wird jedoch nicht genutzt. Orso et al. [OHR00] schlagen hierzu die Definition von Schemata (DTDs) vor. Allerdings beschränken sie sich dabei auf ein Schema pro Informationsaspekt. Zudem bleiben Speicherung und Abfrage der Metadaten in ihrem Ansatz unklar. Problematisch ist auch die Wiederverwendung von Metadaten, da diese stets an eine bestimmte Komponente gebunden sind.

Mit der zunehmenden Komponentenorientierung in der Softwareentwicklung haben sich Spezifikations-Frameworks zur umfassenden Beschreibung von Softwarekomponenten herausgebildet. So nimmt UnSCom eine eindeutige Trennung der Spezifikationsaspekte in vertragliche Ebenen vor [Ov04]. Dabei werden sowohl maschinenlesbare als auch informelle Beschreibungen verwendet, die in einer Vielzahl von Artefakten resultieren. Der vereinheitlichte Spezifikationsrahmen für Fachkomponenten des GI-Arbeitskreises 5.10.3 schlägt darüber hinaus mehrere Notationen für eine Standardisierung vor [ABC02]. Eine technische Realisierung in Form eines Komponentenspeichers setzt voraus, dass diese vielfältigen Informationsarten parallel verwendet werden können. Daraus resultiert ein Bedarf an flexiblen Komponentenspeichern, in die neben ausführbaren Komponenten auch Entwicklungs- und Spezifikationsartefakte integriert werden.

Um den bestehenden Problemen bei der strukturierten Verwaltung von Spezifikations- und Realisierungsartefakten zu begegnen, erlaubt das nachfolgend präsentierte Metaschema eines generischen Komponentenspeichers die Verwaltung dieser unterschiedlichen Informationen. Dabei liegen dem Entwurf des Metaschemas drei zentrale Gestaltungsdimensionen für die Beschreibung von Softwarekomponenten zugrunde, die sich unmittelbar aus den beschriebenen prinzipiellen drei Abstraktionsebenen von Komponentenspeichern nach Mili et al. ableiten lassen und nachfolgend genauer erläutert werden (siehe Abbildung 2):

Vielfalt der Artefakte. Auf den ersten Blick stellen Softwarekomponenten ein abgeschlossenes Paket von binären Softwareartefakten dar. Tatsächlich werden in einem komponentenorientierten Softwareprozess jedoch sehr unterschiedliche Artefakte erzeugt, die jeweils eine bestimmte Funktion erfüllen. Dies können z.B. Anforderungen, Testfälle, Konfigurationsdateien, Modell- und Schnittstellenbeschreibungen sein. Das Beschreibungsschema eines Komponentenspeichers sollte daher in der Lage sein, beliebige heterogene Softwareartefakte zu integrieren.

Vielfalt der Informationsaspekte. An einem Softwareentwicklungsprojekt sind i.d.R. nicht nur Softwareentwickler, sondern auch Tester, Installateure, Systemadministratoren und Projektmanager beteiligt, die unterschiedliches Wissen für ihre Arbeit benötigen und in den Entwicklungsprozess einfließen lassen. Je nach Rolle benötigen Benutzer andere Lösungsbausteine zur Erfüllung ihrer Aufgaben. Darüber hinaus kann die Anfrageart an einen Komponentenspeicher je nach Informationsaspekt variieren, z.B. Volltextsuche und strukturierte Suche.

Vielfalt der Formate. Für viele Aspekte einer Komponente, wie die Schnittstelle oder die Bindung, gibt es bereits breit akzeptierte Beschreibungsansätze. Vor allem im Bereich der Schnittstellenspezifikation haben sich mit IDL [OMG02], UML [OMG05] und

WSDL [W3C01] gleich mehrere komplementäre Ansätze etabliert. Diese Notationen sind in der Regel partiell und decken nicht alle relevanten zu beschreibenden Aspekte einer Komponente ab, während sie sich andererseits aber zum Teil auch überlappen. Ein mächtiger und somit praxistauglicher Komponenten-Beschreibungsansatz wird diese Notationen aufgreifen und sinnvoll kombinieren, nicht aber ersetzen. Als Beispiel für diese parallele Verwendung mehrerer Beschreibungsformate sei der vereinheitlichte Spezifikationsrahmen für Fachkomponenten genannt [ABC02].

Die Aufgabe des hier vorgestellten Metaschemas ist es, die Informationsaspekte und Formate in Relation zur Repräsentation eines Softwareartefakts zu setzen. Diese flexible Ausgestaltung eines Metaschemas trägt dazu bei, den Komponentenspeicher entlang des gesamten komponentenorientierten Entwicklungslebenszyklus einsetzen zu können. Nachfolgend wird unser Metaschema beschrieben, welches direkt auf den oben beschriebenen Gestaltungsdimensionen aufbaut.

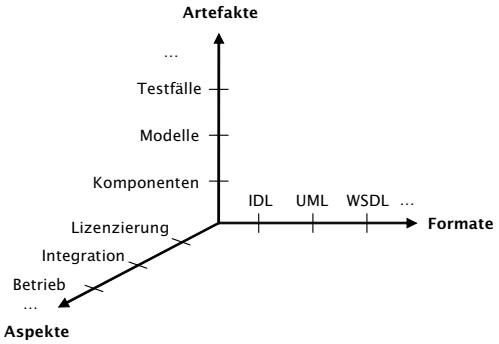

Abbildung 2: Gestaltungsdimensionen eines Komponentenspeichers

3 Ein Metaschema zur Komponentenbeschreibung

3.1. Grundlegende Struktur

Wie die vorangegangenen Ausführungen aufgezeigt haben, deckt die konkrete Beschreibung einer Softwarekomponente eine Reihe von Informationsaspekten ab, die für die am Softwareentwicklungsprozess beteiligten Akteure relevant sind. Abhängig vom Ziel und Entstehungshintergrund einer Beschreibungstechnologie handelt es sich dabei um ein variierendes Spektrum von Aspekten. Eine umfassende Beschreibung einer Komponente ist daher i.A. eine Kombination mehrerer konkreter Beschreibungen, die auf diversen Beschreibungstechnologien basieren. Das hier vorgestellte Metaschema soll die reibungslose Kombination von Beschreibungstechnologien zur Charakterisierung von Softwareartefakten als Bestandteil einer Softwarekomponente abbilden.

Abbildung 3 zeigt den Aufbau des Metaschemas als UML-Klassendiagramm. *Artefakte* und *Metadaten* sind die zentralen Komponenten des Schemas. Ein *Artefakt* besitzt stets genau einen *Artefakttyp*. Auf der Grundlage des Metaschemas werden die Artefakttypen

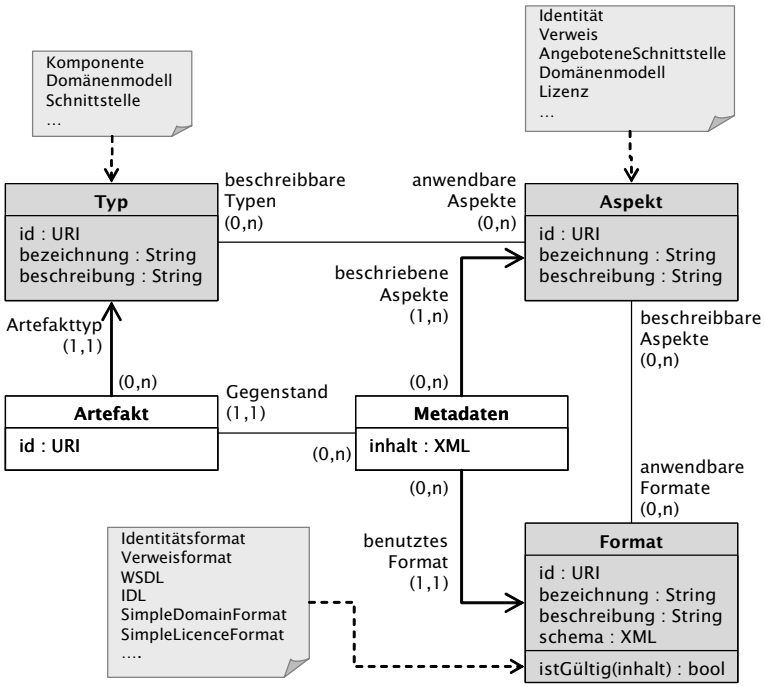

Abbildung 3: Metaschema zur formatneutralen Beschreibung von Softwareartefakten

definiert, z.B. *Komponente* für Softwarekomponenten und *Domänenmodell* für Softwaremodelle. Jedes Artefakt kann mit beliebig vielen Artefaktbeschreibungen (*Metadaten*) versehen werden. Jede Beschreibung hat genau ein Artefakt zum Gegenstand und deckt beliebig viele Informations*aspekte* ab (im Gegensatz zu einer fest vorgegebenen Menge an Facetten [Pri91]). Ähnlich den Artefakttypen werden auf Basis unseres Schemas eine Reihe vordefinierter Informationsaspekte bereitgestellt, wie z.B. *Angebotene-Schnittstelle* für Beschreibungen, die die Schnittstelle eines Artefakts charakterisieren, *Lizenz* für Beschreibungen, die vertragliche Nutzungsbedingungen festlegen usw. Nicht jede Kombination eines Artefakttyps mit einem Informationsaspekt ist sinnvoll. Das Metaschema sieht daher eine *Verträglichkeitsrelation* vor, die für jeden Artefakttyp die Menge der relevanten Informationsaspekte definiert.

Die Kodierung der konkreten Beschreibungen kann auf unterschiedlichen Beschreibungstechnologien basieren, wie das o.a. Beispiel der unterschiedlichen Ansätze zur Schnittstellenspezifikation bereits andeutet. Im Metaschema werden sie durch *Formate* repräsentiert. Eine zweite Verträglichkeitsrelation gibt an, welche Formate zur Beschreibung eines Informationsaspekts eingesetzt werden können.

Die Anwendung des Metaschemas auf die Schnittstellenspezifikation einer Softwarekomponente wird in Abbildung 4 beispielhaft dargestellt. Für einen Informationsgegenstand vom Typ *Komponente* können dabei mehrere Schnittstellenbeschreibungen in unterschiedlichen Formaten existieren, z.B. wenn eine Softwarekomponente sowohl einen CORBA-Dienst als auch eine Web-Service Schnittstelle zur Verfügung stellt.

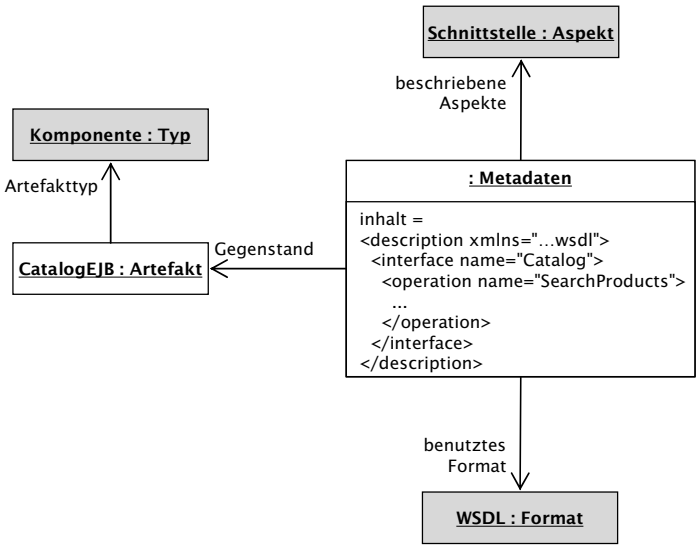

Abbildung 4: Beispielhafte Instanzierung des Metaschemas zur Abbildung der Schnittstellenbeschreibung einer Komponente

Die Abstraktion von konkreten Beschreibungsformaten zu Artefakttypen und anwendbaren Aspekten ermöglicht die Entwicklung eines Komponentenspeichers, der leicht auf individuelle Bedürfnisse angepasst werden kann und gleichzeitig alle Phasen des Komponentenlebenszyklus unterstützt. Dabei können neue Beschreibungsformate einfach anhand ihrer Typen und abgedeckten Aspekte integriert werden, ohne dass Interpretationsfähigkeit verloren geht.

3.2 Modularisierung der Beschreibungen

Um die Wiederverwendung von Artefakten und ihrer Beschreibungen zu fördern, bieten einige Beschreibungstechnologien Modularisierungskonzepte an. So können beispielsweise in der Spezifikation eines Dienstes mittels WSDL die Datentypen aus einer externen XML-Schemadefinition genutzt werden. Das hier präsentierte Metaschema unterstützt diese Form der Modularisierung, indem die Beschreibung eines Artefakts Verweise auf andere Artefakte enthalten kann. Damit ein Artefakt referenziert werden kann, muss es mit einem Identifikator versehen werden. Der Identifikator wird als ein eigenständiger Metadatensatz unter dem Informationsaspekt *Identität* abgelegt. Wir benutzen dazu ein einfaches URI-basiertes Format:

```
<i:Identity xmlns:l="http://www.collabawue.de/sbse/ns/identity">
  <i:Identifier uri="java:de.collabawue.Catalog"/>
</i:Identity>
```

Abbildung 5 zeigt zwei Artefakte: eine Schnittstelle „CatalogAPI“ und eine Komponente „CatalogEJB“, die diese Schnittstelle unterstützt. Der Metadatensatz, der die Komponen-

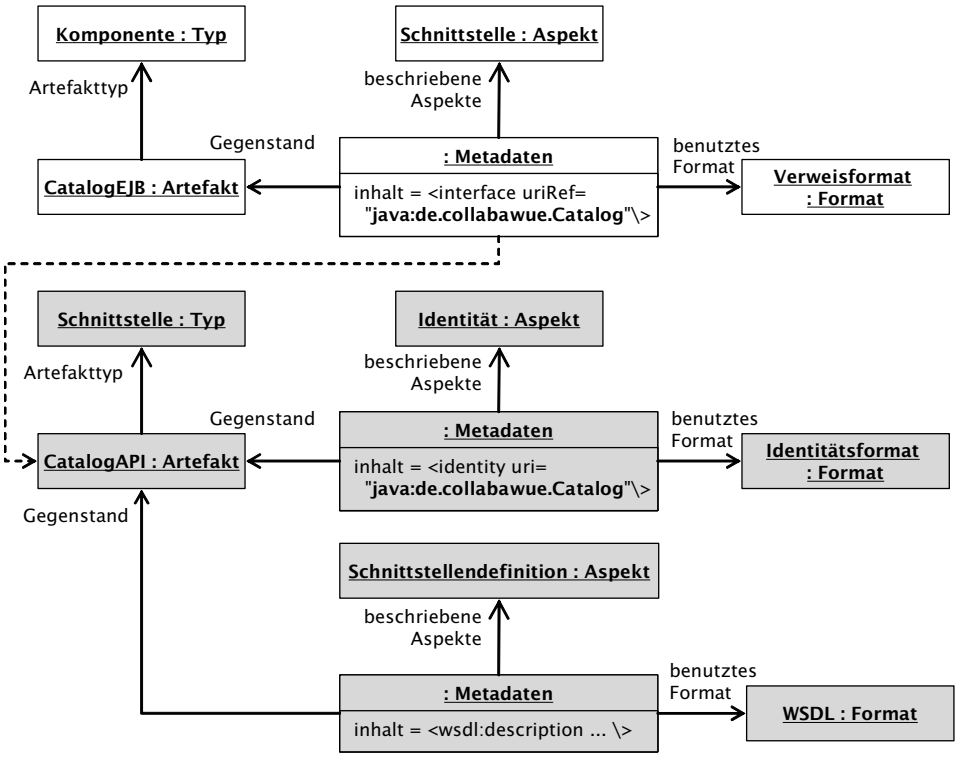

Abbildung 5: Beschreibung modularer Artefakte

te „CatalogEJB“ hinsichtlich ihrer Schnittstelle beschreibt, enthält hier statt einer vollständigen Spezifikation der Schnittstelle lediglich einen Verweis auf das Artefakt „CatalogAPI“.

3.3 Black-Box- und White-Box-Artefakte

Viele Softwarekomponenten im engeren Sinne liegen in binärer Form vor. Es handelt sich dabei um *Black-Box-Artefakte*, deren Suche einzig und alleine anhand ihrer Metadaten erfolgt. Andere Typen von Artefakten, wie etwa MOF-basierte Softwaremodelle, werden aufgrund ihrer offenen Struktur auch als *White-Box-Artefakte* bezeichnet. Ihre innere Struktur kann bei einer Suche im Komponentenspeicher mit ausgewertet werden.

In unserem Ansatz werden Black-Box-Artefakte in einem separaten binären Speicher abgelegt und mittels eines URLs lokalisiert. Um die Verknüpfung zwischen der Beschreibung und dem Artefakt selbst herzustellen, enthält die Beschreibung eines jeden Black-Box-Artefakts den Informationsaspekt *Lokalisierung* mit der Angabe der URL:

```
<l:Location xmlns:l="http://www.collabawue.de/sbse/ns/location">
  <l:Link url="http://www.collabawue.de/binaries/catalog-ejb.jar"/>
</l:Location>
```

White-Box-Artefakte werden hingegen gemeinsam mit ihren Beschreibungen abgelegt. Zu diesem Zweck wird der Informationsaspekt *Identität* verwendet. Wie der Name bereits andeutet, sind die unter diesem Aspekt abgelegten Daten keine Metadaten, sondern das Artefakt selbst, etwa ein in XMI kodiertes Softwaremodell. Der innere Aufbau eines White-Box-Artefakts kann somit leicht zum Gegenstand einer Suchanfrage werden.

4 Technische Realisierung als Komponentenspeicher

Das vorgestellte Metaschema ermöglicht es, heterogene Softwareartefakte als Teil einer Komponente zu beschreiben und strukturiert in einem Komponentenspeicher zu verwalten. Die im Komponentenspeicher abgelegten Softwareartefakte sind nicht auf eine vordefinierte Anzahl von Beschreibungsformaten beschränkt. Demnach kann eine generische Systemarchitektur zur Verfügung gestellt werden, die individuell für bestimmte Einsatzszenarien, wie z.B. die inner- oder überbetriebliche Wiederverwendung, anpassbar ist. Insbesondere ist es mit Hilfe der Aspekte möglich, personalisierte Sichten für Benutzerrollen zu definieren und somit den gesamten Prozess der komponentenbasierten Softwareentwicklung abzudecken.

In Abbildung 6 ist der Aufbau des Komponentenspeichers schematisch dargestellt. Im Mittelpunkt steht die Verwaltung der Komponenten im Komponentenverzeichnis, das sich aus einer XML-fähigen Datenbank und einer Service-Schnittstelle für die Ablage, Suche und Änderung der Beschreibungen zusammensetzt. Die eigentlichen Ressourcen werden in einem separaten Speicher abgelegt. Dabei kann es sich sowohl um ein CVS-Repository als auch eine Web-Ressource handeln. Der Speicherungsort einer Ressource wird vom Informationsaspekt *Lokalisierung* repräsentiert und kann folglich flexibel erweitert werden, ohne die Interpretationsfähigkeit zu verlieren.

Die Benutzerschnittstellen zur Abfrage- und Verwaltung von Artefaktbeschreibungen sind von der restlichen Architektur entkoppelt. Dieses ist vorteilhaft, weil der Abfragemechanismus je nach der Benutzerrolle variieren kann. Beispielsweise ist für die Suche in Anforderungsdokumentationen eher ein schlüsselwortbasiertes Verfahren geeignet, während die Wiederverwendung bereits existierender Analysemodelle in einem ähnlichen Kontext eine strukturierte Suche erfordert. Neben passiven Abfragen wird in be-

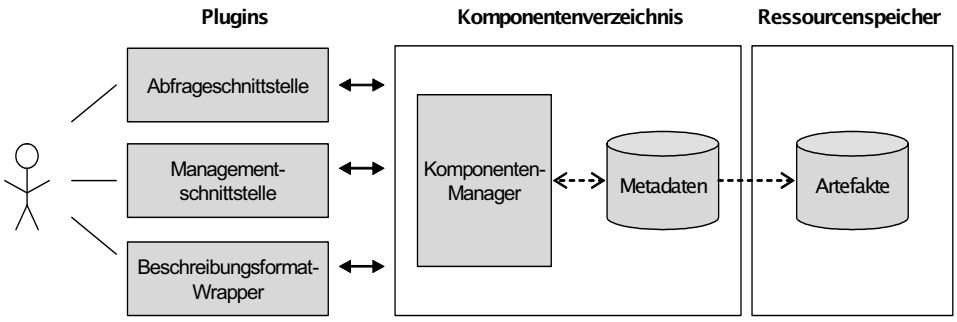

Abbildung 6: Architektur des Komponentenspeichers

stimmten Einsatzszenarien ein aktiver Push-Mechanismus gefordert [Ye02], der in dieser Architektur bei Bedarf über ein separates Modul realisiert werden kann.

Bei der Implementierung einer Abfragesprache sind darüber hinaus die Besonderheiten des Metaschemas zu beachten. Für das Absetzen von Abfragen wurde im Rahmen der prototypischen Implementierung ein XML-basiertes Abfrageschema entwickelt. In seiner Struktur ähnelt das Schema SQL-Anfragen:

```
<q:query
 xmlns:q="http://www.collabawue.de/sbse/repository/namespaces/query">
  <q:prefix name="wsdl" value="http://schemas.xmlsoap.org/wsdl/"/>
  <q:select aspects="Schnittstellenverhalten"/>
  <q:from type="Komponente"/>
  <q:where>
    <q:matches aspect="AngeboteneSchnittstelle" format="WSDL"
     expression="//wsdl:operation[@name='SearchProducts']"/>
    <q:similar terms="Catalog"/>
  </q:where>
</q:query>
```

Im Select-Teil der Anfrage (`q:select`) wird die Art der Metadaten spezifiziert, die im Suchergebnis zurückgeliefert werden sollen. Die Angabe geschieht durch eine Auflistung von Aspekten. Der From-Teil (`q:from`) begrenzt die bei der Suche betrachteten Artefakte auf die Artefakte eines gegebenen Typs. Im Where-Teil (`q:where`) können schließlich weitere, sehr präzise Auswahlbedingungen formuliert werden. Zur Verfügung stehen dabei zwei Arten von Filtern. Der Ausdruck `q:matches` begrenzt die Menge der betrachteten Artefakte auf Artefakte, deren Beschreibung auf einen XPath-Ausdruck passt (`expression`). Die Art der Beschreibung wird dabei durch die Angabe von Aspekt (`aspect`) und Format (`format`) bestimmt. Der Ausdruck `q:similar` begrenzt die Artefakte auf diejenigen, in deren Beschreibungen ein Begriff oder eine Phrase vorkommen (`term`). Die obige Anfrage liefert Informationen über die Schnittstelle und das Verhalten aller Artefakte vom Typ *Komponente*, die eine Schnittstellenbeschreibung in WSDL besitzen, in der die Operation „SearchProducts“ genannt wird, und in deren Metadaten der Begriff „Catalog“ auftritt.

Der hier vorgestellte Ansatz ist geeignet, um strukturiert über die im Komponentenspeicher vorliegenden Metadaten zu suchen. Dieses schließt auch White-Box-Artefakte ein. Allerdings ist eine Berücksichtigung des Inhalts von Black-Box-Artefakten nur dann möglich, wenn ein formatspezifischer Suchindex bereitgestellt wird. Darüber hinaus kann die Architektur in einer späteren Version um weitere Werkzeuge, wie z.B. Wrapper zur Vereinheitlichung von Beschreibungsformaten, ergänzt werden. So ist es bei der Entwicklung eines vereinheitlichten Komponentenverzeichnisses sinnvoll, sich auf ein Beschreibungsformat für das Schnittstellenverhalten zu einigen und die Beschreibungen in einer Sprache (z.B. IDL, WSDL) in Beschreibungen in einer anderen Sprache (z.B. UML) zu übersetzen. Auf diese Weise können die Beschreibungen einer Komponente nachträglich mit automatisch generierten Metadaten und Softwareartefakten angereichert werden.

5 Bewertung und Ausblick

In Rahmen dieser Arbeit wurde ein Metaschema zur flexiblen Beschreibung von Softwarekomponenten vorgestellt. Des Weiteren wurde die prototypische Implementierung des Metaschemas in einem Komponentenspeicher beschrieben. Der vorgestellte Ansatz erlaubt die explizite Berücksichtigung und Nutzbarmachung der Vielfalt von Informationsaspekten, Formaten und Artefakten, die in gegenwärtigen komponentenorientierten Softwareentwicklungsprojekten anzutreffen sind und die eine große Relevanz für den Prozess des Suchens, Auffindens, Verstehens und Integrierens existierender Lösungsbausteine besitzen.

Klassische Ansätze zur Repräsentation von Softwareartefakten in Komponentenspeichern konzentrieren sich auf ausgewählte Gesichtspunkte, wie etwa die Klassifikation oder Indizierung von Softwarekomponenten für eine begrenzte Anwendergruppe. Unser Metaschema deckt diese Funktionalitäten mit ab, da es sich für die Abbildung beliebiger facettierter Klassifikationen [Pri91] eignet, bietet aber darüber hinaus einen weitaus umfassenderen Ansatz, indem es die strukturierte Erfassung von Metadaten zu beliebigen Repräsentationsformaten ermöglicht. Darüber hinaus eignet sich der vorgestellte Ansatz für die parallele Verwaltung von Realisierungs- und Spezifikationsartefakten.

Zukünftige Aufgaben umfassen zum einen die Erweiterung der Instanzierungsebene des Metaschemas um geeignete Artefakttypen und Formate. Hierzu sollte ein einheitlicher Standard angestrebt werden, der die Pflege der Metadaten und ihren Austausch zwischen Komponentenspeichern erleichtert. Die semi-strukurierte Suche auf der Basis eines XML-basierten Abfrageschemas wird durch den vorgestellten Prototyp bereits unterstützt. Um die Skalierbarkeit des Ansatzes zu gewährleisten, soll die Suchfunktionalität um eine Ähnlichkeitssuche auf der Basis von Ontologien erweitert werden. Zum anderen ist es erforderlich, geeignete Benutzerschnittstellen zur Verwaltung und Abfrage der Komponenten zu entwickeln. Hierzu müssen zunächst unterschiedliche Abfragemechanismen auf ihre Eignung für die jeweilige Nutzergruppe evaluiert werden.

Die hier vorgestellte prototypische Implementierung verwendet XML als Repräsentationsformat. Es ist alternativ möglich, Wissensrepräsentationssprachen, wie z.B. die Web Ontology Language (OWL) [W3C04], für die Realisierung heranzuziehen. Mit der Verwendung einer Wissensrepräsentationssprache wie OWL könnte durch deduktives Schließen auf einer Wissensbasis das Spektrum für Suchanfragen deutlich erweitert werden. Dabei müsste jedoch der realisierbare Zusatznutzen gegen die zusätzliche Komplexität abgewogen werden.

Eine Validierung der Vorteilhaftigkeit des hier präsentierten Ansatzes wird im Rahmen der Aktionsforschung des Forschungsprojekts „CollaBaWue“ (siehe die Projekt-Website http://www.collabawue.de) erfolgen.

Danksagung: Diese Arbeit ist im Rahmen des Projekts „CollaBaWue“ (http://www.collabawue.de) des Forschungsverbunds „PRIMIUM“ (http://www.primium.org) entstanden, welches durch das Ministerium für Wissenschaft, Forschung und Kunst des Landes Baden-Württemberg gefördert wird.

Literaturverzeichnis

[ABB01] Atkinson, C., Bayer, J., Bunse C., et al.: Component-based Product Line Engineering with UML, Addison Wesley, 2001.

[ABC02] Ackermann, J., Brinkop, F., Conrad, S. et al.: Vereinheitlichte Spezifikation von Fachkomponenten, Hrsg.: K. Turowski, Memorandum des Arbeitskreises Komponentenorientierte Anwendungssysteme, Februar, 2002.

[BR89] Biggerstaff, T. J. und Richter, C.: Reusability framework, assessment, and directions. In: Software reusability: vol. 1, concepts and models: ACM Press, S. 1-17, 1989.

[CD00] Cheesman, J. und Daniels, J.: UML Components: A Simple Process for Specifying Component-Based Software. Addison-Wesley Professional, 2000.

[CW94] Convent, B. und Wernecke, W.: Bausteinverwaltung und Suchunterstützung - Basis für Software-Wiederverwendung. HMD - Praxis Wirtschaftsinform. 180: 1994.

[FK05] Frakes, W.B. und Kang, K.: Software Reuse Research: Status and Future. In: IEEE Trans. on Softw. Eng., vol 31, no. 7, S. 529-536, 2005.

[HK04] Hildenbrand, T. und Korthaus, A.: A Model-Driven Approach to Business Software Engineering. Proc. of the 8th World Multi-Conf. on Systemics, Cybernetics and Informatics (SCI 2004), Vol. IV, IIIS, Orlando, Florida, USA, July 18-21, S. 74-79, 2004.

[IEEE04] IEEE Computer Society: Guide to the Software Engineering Body of Knowledge. http://www.swebok.org, 2004.

[Ko01] Korthaus, A.: Komponentenbasierte Entwicklung computergestützter betrieblicher Informationssysteme. Peter Lang Verlag, 2001.

[Kr92] Krueger, C. W.: Software Reuse. ACM Comput. Surv., vol. 24, S. 131-183, 1992.

[Kr00] Kruchten, P.: The Rational Unified Process: An Introduction, Second Edition. Addison-Wesley Longman Publishing Co., Inc., 2000.

[MMM98] Mili, A., Milli, R. und Mittermeir, R.T.: A survey of software reuse libraries. In: Annals of Software Engineering, vol. 5, S. 349-414, 1998.

[OHR00] Orso, A., Harrold, M.J. und Rosenblum, D. S.: Component Metadata for Software Engineering Tasks. In: Emmerich, W. und Tai, S. (Hrsg.): Proc. of the 2nd Int. Workshop of Engineering Distributed Objects, LNCS1999, S. 129-144, 2001.

[OMG02] Object Management Group: CORBA 3.0 - IDL Syntax and Semantics. http://www.omg.org/cgi-bin/doc?formal/02-06-07, 2002.

[OMG05] Object Management Group: UML 2.0 Superstructure Specification. http://www.omg.org/cgi-bin/doc?formal/05-07-04, 2005.

[Ov04] Overhage, S.: UnSCom: A Standardized Framework for the Specification of Software Components. In: Weske, M. und Liggesmeyer, P. (Hrsg.): Object-Oriented and Internet-Based Technologies, Proc. of the 5th Annual Int. Conf. on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World, NODe 2004, LNCS 3263, Erfurt 2004, S.169-184.

[Pri91] Prieto-Diaz, R.: Implementing Faceted Classification for Software Reuse. In: Comm. ACM, vol. 34, no. 5, S. 88-97, 1991.

[Sz02] Szyperski, C.: Component Software - Beyond Object-Oriented Programming. Addison-Wesley, 2nd ed., London, 2002.

[VZJ03] Vitharana, P., Zahedi, F. und Jain, H.: Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis. In: IEEE Trans. on Softw. Eng., vol. 29 no. 8, S. 649-664, 2003.

[W3C01] W3 Consortium: Web Services Description Language (WSDL). W3C Language Specification, Version 1.1. http://www.w3.org/TR/wsdl, 2001.

[W3C04] W3 Consortium: OWL Web Ontology Language Reference, W3C Recommendation, http://www.w3.org/TR/owl-ref, 2004.

[Ye01] Ye, Y.: An Active and Adaptive Reuse Repository System. In: Proceedings of 34th Hawaii International Conference on System Sciences (HICSS-34), 2001.