Zur Spezifikation von Komponenten der Informationssystementwicklung mit Softwareverträgen

Sven Overhage

Universität Augsburg
Wirtschaftsinformatik II – Systems Engineering
Universitätsstraße 16
86159 Augsburg
sven.overhage@wiwi.uni-augsburg.de

Abstract: In diesem Beitrag wird ein methodischer Rahmen für die Spezifikation von Komponenten beschrieben, der eine wichtige Voraussetzung für die Schaffung von Werkzeugen zur Unterstützung der betrieblichen Informationssystementwicklung mit Komponenten darstellt. Er basiert auf dem Konzept des Softwarevertrags und erweitert dieses durch die Einführung von Dienst- und Kompositionsverträgen. Mit dem Spezifikationsrahmen lassen sich Kompositionsverträge modellieren, die die Schnittstellen von Komponenten auf mehreren Vertragsebenen beschreiben. Diese Ebenen sind thematisch gruppiert und beinhalten Informationen zu folgenden Aspekten einer Komponente: allgemeine und kommerzielle Informationen, Klassifikationen, benötigte und bereitgestellte fachliche Funktionalität, systemtechnische Architektur der benötigten und bereitgestellten Komponentenschnittstellen sowie benötigte und bereitgestellte Qualität. Darüber hinaus benennt der Beitrag verschiedene Aufgaben einer Entwicklungsmethodologie und zeigt, wie diese mit Informationen über den Kompositionsvertrag einer Komponente von Werkzeugen unterstützt werden können.

1 Einleitung

Die Reduzierung der Entwicklungsarbeit auf das Durchsuchen von Komponentenkatalogen, die Ermittlung einer kompatiblen Komponentenmenge und deren anschließende Verbindung (Komposition) zu einem Informationssystem verheißt zahlreiche Vorteile für betriebliche Entwicklungsvorhaben. So wird in der Literatur vor allem die Verkürzung der Entwicklungszeit, die Verbesserung der Wartbarkeit und Skalierbarkeit und damit eine deutliche Senkung der Entwicklungskosten in Aussicht gestellt [Br00], [Sz03].

Bei genauerer Betrachtung dieses zunächst verlockend wirkenden zentralen Paradigmas der Komponentenorientierung in der Informationssystementwicklung zeigt sich jedoch, dass es eine Vielzahl ungelöster Herausforderungen mit sich bringt: wie ist die *Kompatibilität* von Komponenten zu ermitteln, wie ist mit *Heterogenitäten* zwischen verschiedenen Komponenten umzugehen, wie lassen sich zu erwartende *Eigenschaften des Informationssystems* aus den Eigenschaften der Komponenten vor der Komposition ableiten und wie sind Komponenten schließlich miteinander zu verbinden? Der Fortbestand dieser genannten Herausforderungen lässt vermuten, dass die komponentenbasierte Infor-

mationssystementwicklung vor dem Beginn einer „Kompositionskrise“ stehen könnte und macht die Schaffung einer Entwicklungsmethodologie, die Methoden und Werkzeuge zur effizienten Entwicklung mit Komponenten bereitstellt, zu einem kritischen Erfolgsfaktor.

Bei der Schaffung einer solchen Entwicklungsmethodologie gilt es, vielfältige Aufgaben zu unterstützen und insbesondere Methoden und Werkzeuge für die Zertifizierung, Komponentensuche, Kompatibilitätstests, Adaptergenerierung und Vorhersage von Informationssystemeigenschaften bereitzustellen (siehe Abbildung 1). Dabei sind alle zur Unterstützung dieser Aufgaben zu entwickelnden Methoden und Werkzeuge auf Informationen über einzelne Komponenten angewiesen, die sie zur Erfüllung ihrer Aufgaben jeweils entsprechend auswerten. Zur Informationsbeschaffung sollten sie jedoch möglichst nicht direkt auf physische Komponenten zugreifen, sondern hierfür entsprechende Komponentenspezifikationen verwenden. Dies ist notwendig, um den Einsatz aufwendiger Testmethoden und Verfahren des Reverse Engineering zu vermeiden, die die Entwicklungskosten erhöhen, die verfrühte Beschaffung von Komponenten (schon vor der Eignungsprüfung) voraussetzen und zudem kaum in der Lage sind, die benötigten Informationen aus Black-Box-Komponenten in angemessener, d.h. akkurater Form zu ermitteln [GAO95], [We01].

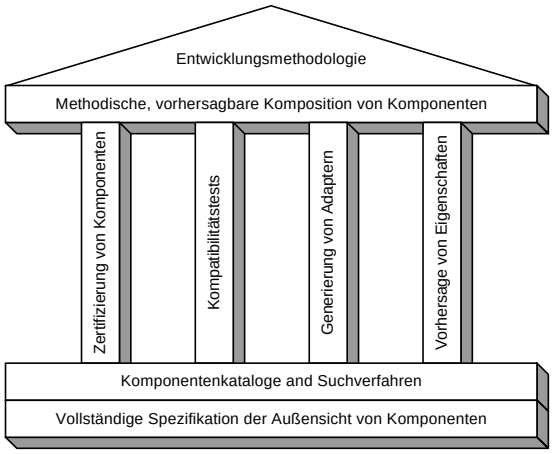

Abbildung 1: Elemente einer Methodologie für die Komposition von Informationssystemen aus Komponenten (basierend auf [Wa03], [ZW97]).

Die Entwicklung einer standardisierten Methode zur Bereitstellung von Komponentenspezifikationen, die Angaben zu den von einer Komponente angebotenen Diensten sowie ihren Kontextabhängigkeiten beinhalten, ist daher eine wichtige Voraussetzung für die Schaffung einer Entwicklungsmethodologie. Dieser Beitrag stellt einen standardisierten Rahmen zur Modellierung und Spezifikation von Komponenten vor, der sowohl den Inhalt als auch die zu verwendenden Notationen vorgibt. Hierdurch wird es möglich, einerseits die Entwicklung von Methoden und Werkzeugen einer Methodologie von verschiedenen Parteien durchführen zu lassen und andererseits, da diese von denselben

Spezifikationen Gebrauch machen, gleichzeitig die Entstehung einer einheitlichen Methodologie sicherzustellen.

Der vorgestellte Modellierungs- und Spezifikationsrahmen trägt den Namen „Vereinheitlichte Spezifikation von Komponenten“ (engl. Unified Specification of Components, UnSCom). Er basiert auf dem Konzept des *Softwarevertrags* (Design by Contract) [Me97] und erweitert dieses für den Einsatz in der komponentenbasierten Informationssystementwicklung durch die Einführung von Dienst- und Kompositionsverträgen. Der Spezifikationsrahmen erfüllt eine Reihe von generellen Anforderungen, die zunächst im Folgenden näher ausgeführt werden, und führt darüber hinaus verschiedene Vertragsebenen ein, die zum Teil von einem Standardisierungsprojekt des Arbeitskreises WI-KobAS der Gesellschaft für Informatik (GI) vorgeschlagen wurden [Tu02]. Nach einer Zusammenfassung relevanter Arbeiten aus dem wissenschaftlichen Umfeld wird abschließend ein kurzer Ausblick auf die Entwicklung einer Methodologie, die auf dem Konzept des Kompositionsvertrags basiert, gegeben.

2 Generelle Anforderungen

Die folgenden generellen Anforderungen sind von einem Spezifikationsrahmen zu erfüllen, damit er als Basis für die Schaffung einer Entwicklungsmethodologie effizient eingesetzt werden kann:

- Er muss alle nach außen sichtbaren und von den Elementen einer Methodologie verwendeten Merkmale einer Komponente *vollständig beschreiben*. Es muss also möglich sein, alle Aufgaben, die zu einer Entwicklungsmethodologie gehören (siehe Abbildung 1), auf der Basis der zur Verfügung gestellten Spezifikationen umzusetzen.

- Der Spezifikationsrahmen sollte ein *Gleichgewicht zwischen der Ausdrucksmächtigkeit* von Komponentenspezifikationen auf der einen *und ihrer Anwendbarkeit* auf der anderen Seite anstreben. Im Allgemeinen sind Notationen, die die Spezifikation komplexerer Sachverhalte unterstützen, auch schwieriger statisch (also zum Zeitpunkt der Komposition) auszuwerten bzw. zu verwenden. Dies gilt beispielsweise für Kompatibilitätstests, die von der Existenz eines effizienten (entscheidbaren) Vergleichsalgorithmus für die spezifizierten Sachverhalte abhängen.

- Seine *Vorgaben sind normativ*, d.h. er schreibt den Inhalt von Spezifikationen (was zu spezifizieren ist) und die zur Spezifikation einzusetzenden Notationen (wie zu spezifizieren ist) exakt vor. Dies ist notwendig, um eine Einheitlichkeit von Komponentenspezifikationen zu gewährleisten, die eine wesentliche Voraussetzung für deren Verwendbarkeit in Entwicklungswerkzeugen ist.

- Der Spezifikationsrahmen sollte über eine *modulare Struktur* verfügen, so dass zusätzliche Spezifikationen abwärtskompatibel als neue Module hinzugefügt werden können. Hierdurch wird die Stabilität von Komponentenspezifikationen gefördert

und sichergestellt, dass auch ältere Werkzeuge mit Komponentenspezifikationen, die nach einer neueren Version des Rahmens angefertigt wurden, umgehen können.

– Er sollte zur Modellierung und Spezifikation von Komponenten möglichst auf *etablierte, industriefähige Notationen* zurückgreifen. Obwohl sich durch diese Maßnahme gegenüber dem Stand der Wissenschaft in der Regel nur ein suboptimales Spezifikationsergebnis erzielen lassen wird, ist jedoch insbesondere die Akzeptanzwahrscheinlichkeit in der *betrieblichen* Informationssystementwicklung sowie die zu erwartende Unterstützung durch Werkzeuge aus der Softwareindustrie deutlich verbessert.

– Die gewählten Notationen sollten sowohl *maschinell auswertbar als auch vom menschlichen Entwickler lesbar* sein. Für die Werkzeugunterstützung ist die Verwendung formaler, präziser Spezifikationen unerlässlich. Dennoch sollten sie ebenfalls von Entwicklern verstanden und benutzt werden können, da eine vollständige Werkzeugunterstützung (noch) nicht für alle Aufgaben einer Entwicklungsmethodologie gegeben ist.

3 UnSCom: Ein vertragsbasierter Spezifikationsrahmen

Auf den zuvor genannten generellen Anforderungen aufbauend wird im Folgenden der Spezifikationsrahmen „Vereinheitlichte Spezifikation von Komponenten“ (engl. Unified Specification of Components, UnSCom) näher vorgestellt. Er basiert auf den Konzepten des Dienstvertrags und des Kompositionsvertrags, die aus dem etablierten Prinzip des *Softwarevertrags* (Design by Contract) [Me97] abgeleitet wurden. Softwareverträge spielen derzeit sowohl in der strukturierten Programmierung als auch der objektorientierten Programmierung (OOP) eine bedeutende Rolle, wo sie zur Spezifikation sog. *Dienstverträge* eingesetzt werden.

Ein *Dienstvertrag* beschreibt dabei die Vorbedingungen, die von einem Klienten erfüllt werden müssen, bevor dieser einen Softwaredienst in Anspruch nehmen kann, sowie die Nachbedingungen, die der Dienst nach seiner Ausführung unter der Voraussetzung garantiert, dass er mit erfüllten Vorbedingungen in Anspruch genommen wurde. Derartige Dienstverträge eignen sich nicht nur für die objektorientierte Programmierung, sondern können (ohne Veränderung des Konzepts) auch zur Beschreibung der Dienste einer Komponente herangezogen werden.

Die Spezifikation von Dienstverträgen ist hilfreich, jedoch für die komponentenbasierte Informationssystementwicklung nicht ausreichend. Da Komponenten im Gegensatz zu Klassen als eigenständige Entwicklungsergebnisse („units of independent deployment“ [Sz03]) anzusehen sind, müssen Entwickler über eine genaue Kenntnis darüber verfügen, welche Dienste eine Komponente anbietet bzw. von ihrer Umgebung (d.h. anderen Komponenten) erwartet, um beispielsweise zu beurteilen, ob sie mit anderen Komponenten erfolgreich zu einem Ganzen (einem Informationssystem) verbunden werden kann [Sz03].

Die von einer Komponente angebotenen bzw. von der Umgebung erwarteten Dienste werden in Form von Schnittstellen spezifiziert [DW99], [Sz03], die jeweils gedachte oder tatsächliche Verbindungen zweier Systeme darstellen [He93]. Hierbei ist zu berücksichtigen, dass eine Komponente in der Regel nur dann Dienste anbietet, wenn die von ihr erwarteten Dienste auch von der Umgebung zur Verfügung gestellt werden (siehe Abbildung 2). Somit besitzt eine Komponente neben Dienstverträgen, die die Bedingungen der Inanspruchnahme ihrer Dienste zur Laufzeit spezifizieren, noch einen *Kompositionsvertrag*, der die Bedingungen beschreibt, unter denen sie ihre Dienste zur Kompositionszeit für einen späteren Aufruf überhaupt zur Verfügung stellt.

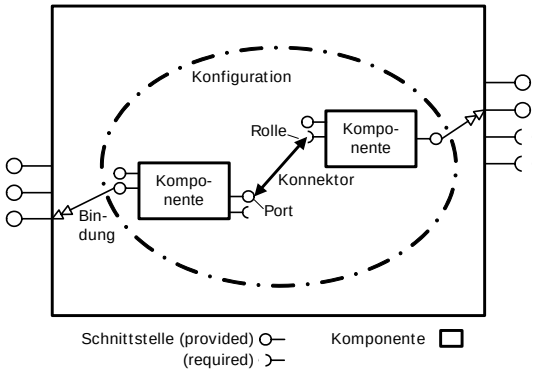

Abbildung 2: Konzeptionelles Komponentenmodell (basierend auf [SG96]).

Ein *Kompositionsvertrag* spezifiziert also die Schnittstellen, die von der Umgebung zur Verfügung zu stellen sind, bevor eine Komponente verwendet werden kann (die sog. benötigten Schnittstellen bzw. „required interfaces“). Darüber hinaus beschreibt er die Schnittstellen, die eine Komponente unter der Voraussetzung anbietet, dass die von ihr erwarteten Schnittstellen zur Verfügung stehen (die sog. bereitgestellten Schnittstellen bzw. „provided interfaces“) [CD00], [DW99]. Um die an den Schnittstellen zu spezifizierenden Dienste näher zu beschreiben, beinhaltet ein Kompositionsvertrag ggf. mehrere Dienstverträge, die somit einen Teil des Kompositionsvertrags einer Komponente darstellen. Darüber hinaus lässt sich ein Kompositionsvertrag auf verschiedenen *Vertragsebenen* spezifizieren, indem die einzelnen Schnittstellen jeweils aus unterschiedlichen Perspektiven beschrieben werden [Be99].

Der in diesem Beitrag vorzustellende UnSCom Spezifikationsrahmen unterstützt die Spezifikation des Kompositionsvertrags einer Komponente und beschreibt die von ihr benötigten bzw. bereitgestellten Schnittstellen detailliert. Dabei wird versucht, eine möglichst vollständige Komponentenbeschreibung zu erreichen, indem verschiedene, teilweise von einem Standardisierungsprojekt eines Arbeitskreises der GI [Tu02] identifizierte Vertragsebenen, zur Verfügung gestellt werden.

Die einzelnen Vertragsebenen lassen sich aus einem Klassifikationsschema (siehe Abbildung 3) ableiten, welches zur möglichst vollständigen Beschreibung einer Komponente in der Horizontalen die verschiedenen Phasen des Komponentenentwicklungsprozes-

ses und in der Vertikalen die zur vollständigen Beschreibung eines Softwareartefakts üblicherweise verwendeten Modellierungs- bzw. Spezifikationsperspektiven unterscheidet [CD94], [Da93], [DW99], [Gr94], [Ol91], [Or97], [Sc97], [Sc98]. Man beachte, dass das Klassifikationsschema auch die klassischen Dienstverträge [Me97] einer speziellen Beschreibungsebene des Kompositionsvertrags zuordnet.

	Funktionalität / Begriffe (fachlich)	Architektur / Schnittstellen (logisch)	Implementierung / Qualität (physisch)
statisch (Struktur)	Informationsobjekte (Datenmodell)	Typdeklarationen, Attribute	Anwendbarkeit, Wartbarkeit, Übertragbarkeit
operational (Wirkungen)	Funktionen (Funktionsmodell)	Ereignisse, Methoden, Ausnahmen, Zusicherungen	Funktionalität, Zuverlässigkeit
dynamisch (Interaktionen)	Prozesse (Prozessmodell)	Interaktions-protokolle	Effizienz

Abbildung 3: Klassifikationsschema zur Ableitung von Vertragsebenen.

Jede der in der Horizontalen unterschiedenen Entwicklungsperspektiven fügt einem Kompositionsvertrag Schnittstellenbeschreibungen mit einem anderen Inhalt hinzu: die *fachliche Perspektive*, die dem Fachentwurf zuzuordnen ist, beschreibt die *fachliche Funktionalität*, die an den Schnittstellen einer Komponente benötigt bzw. bereitgestellt wird. Sie wird in Form domänenspezifischer Begriffe spezifiziert (die entweder als Lexikon oder fachkonzeptuelles Modell dargestellt werden) [CD00], [Si99]. Die *logische Perspektive*, die dem Systementwurf zuzuordnen ist, beschreibt die *systemtechnische Architektur*, die die benötigten bzw. bereitgestellten Schnittstellen aufzuweisen haben [CD00], [DW99]. Sie wird durch Signaturlisten, Dienstverträge und Interaktionsprotokolle beschrieben und kann direkt von Komponentenmodellen wie z.B. dem CORBA Component Model [OM02] verwendet werden. Die *physische Perspektive*, die sich der Implementierung zuordnen lässt, beschreibt die an den Schnittstellen einer Komponente benötigte bzw. bereitgestellte *Qualität*. Diese wird in Form ISO 9216 konformer Qualitätsattribute spezifiziert [IS01], [IS03].

Orthogonal zu den genannten Entwicklungsphasen lassen sich in der Softwareentwicklung verschiedene Modellierungs- bzw. Spezifikationsperspektiven unterscheiden, die den Fokus während der einzelnen Entwicklungsphasen jeweils auf spezielle Aspekte eines Softwareartefakts lenken: die *statische Perspektive* (auch als Datensicht bezeichnet) legt den Schwerpunkt auf die *Struktur* eines Softwareartefakts. Die *operationale Perspektive* (Funktionssicht) beschreibt die *Wirkungen* der Operationen, die auf einem Softwareartefakt ausgeführt werden können. Die *dynamische Perspektive* (Prozesssicht) schließlich spezifiziert den *Ablauf*, d.h. die Interaktionen, an denen ein Softwareartefakt mitwirken kann.

Die mithilfe des Klassifikationsschemas ermittelten neun Ebenen eines Kompositionsvertrags bilden den Kern des UnSCom Spezifikationsrahmens. Er wird durch zwei weitere Spezifikationsebenen vervollständigt, die die Angabe allgemeiner und kommerzieller Informationen sowie die Klassifizierung von Komponenten ermöglichen (siehe Abbildung 4). Die einzelnen Ebenen des UnSCom Spezifikationsrahmens sind thematisch geordnet und verschiedenen Kategorien zugeordnet, die (in Anlehnung an UDDI

[UD00]) als farbige Seiten („Coloured Pages“) unterschieden werden: *White Pages* beinhalten allgemeine und kommerzielle Informationen über Komponenten, *Yellow Pages* bestehen aus Klassifikationen, *Blue Pages* beschreiben die an den Schnittstellen benötigte bzw. bereitgestellte fachliche Funktionalität, *Green Pages* spezifizieren die benötigte bzw. bereitgestellte systemtechnische Architektur von Schnittstellen und *Grey Pages* beschreiben die an den Schnittstellen benötigte bzw. bereitgestellte Qualität.

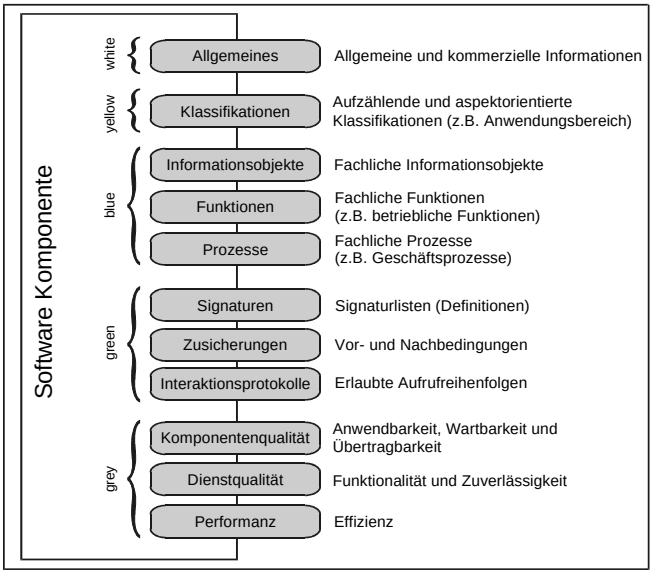

Abbildung 4: Der UnSCom Spezifikationsrahmen.

Der UnSCom Spezifikationsrahmen sieht auf den elf Ebenen verschiedene spezialisierte, (bevorzugt) etablierte, formale und plattformunabhängig verwendbare Notationen für die Erstellung von Komponentenspezifikationen vor. Darüber hinaus unterstützt er nicht nur ein fest definiertes, sondern eine Reihe von Notationsbündeln, die über ein standardisiertes Metaschema integriert werden. Gegenwärtig sind drei Notationsbündel verfügbar: UnSCom/T besteht aus einem Bündel textbasierter Notationen, UnSCom/G besteht aus einem Mix graphischer Notationen der UML 2.0 und WS-Specification [OT03] besteht aus einem Bündel standardisierter Notationen für die Beschreibung von XML Web-Services.

Im Folgenden wird anhand eines kurzen Spezifikationsbeispiels ein vertiefender Überblick über die einzelnen Spezifikationsebenen gegeben. Aufgrund der kompakteren Darstellung werden dabei die textbasierten Notationen aus UnSCom/T verwendet und vorgestellt.

3.1 White Pages: Allgemeine und kommerzielle Informationen

Die „White Pages“ stellen allgemeine und kommerzielle Informationen über Komponenten zur Verfügung, mit denen beispielsweise die Eignung einer Komponente für ein Entwicklungsprojekt von einem wirtschaftlichen Standpunkt aus beurteilt werden kann. Zu den allgemeinen Informationen gehören der Name der Komponente, ihr eindeutiger Schlüssel (sofern vorhanden), die Version, eine Beschreibung sowie der Hersteller und die vorhandenen Möglichkeiten zur Kontaktaufnahme mit diesem (z.B. zur Beschaffung weiterer Informationen).

Name: Oversoft General Ledger
Version: 1.0.3755
Identifier: 206B2F65-E7BC-47a0-8DCF-4E34347322AA
Producer **Identifier**: AEFBE5B6-A13F-4ca4-B322-32ADB89EA14C **Name**: Oversoft Software **Means of Communication** (type: e-mail): info@oversoft.biz **Means of Communication** (type: phone): +49 6003 810260 **Address Line** (type: city): Rosbach **Address Line** (type: country): Germany
Distribution Channel **Identifier**: (type: internet download) **Price** (type: USD): 399.90 **Accepted Payment** (type: credit card) **Scope of Supply:** generalLedger.cab, setup.exe, setup.hlp
Terms of Use **Legal Agreement** (type: license agreement): Oversoft EULA ...

Abbildung 5: Spezifikation allgemeiner und kommerzieller Informationen.

Zu den kommerziellen Informationen gehören sowohl die Kaufbedingungen als auch die Nutzungsbedingungen. Unter den Kaufbedingungen sind Beschreibungen von Distributionskanälen zu verstehen, die jeweils aus einer Bezugsform, einem Preis, möglichen Bezahlverfahren und einem Lieferumfang (d.h. eine Liste der ausgelieferten Artefakte) bestehen. Die Nutzungsbedingungen werden schließlich durch eine Reihe rechtlicher Vereinbarungen spezifiziert, z.B. Lizenzvereinbarungen, Garantievereinbarungen etc.

Die hier genannten Angaben sind vorwiegend beschreibender Natur und werden daher im Allgemeinen umgangssprachlich erfasst. Um jedoch die maschinelle Auswertbarkeit und Verwendbarkeit dieser Informationen zu gewährleisten, schreibt der Spezifikationsrahmen die Verwendung von Taxonomien (die in diesem Fall vorgefertigte Mengen von Alternativen zur Verfügung stellen) vor, mit denen entweder direkt zu spezifizieren ist oder zumindest umgangssprachlich formulierte Informationen zu klassifizieren sind (siehe Abbildung 5).

3.2 Yellow Pages: Klassifikationen

Die „Yellow Pages“ beinhalten Klassifikationen einer Komponente [Pr91], d.h. Einordnungen in eine Reihe von Klassifikationsschemata. Diese Einordnungen stellen vor allem für Werkzeuge, die die Suche nach Komponenten unterstützen sollen, nützliche Informationen dar, da die entsprechenden Klassifikationsschemata beispielsweise zur Strukturierung von Komponentenkatalogen verwendet werden können. Jedes Klassifikationsschema gibt eine Reihe von Kategorien, in die eine Komponente eingeordnet werden kann, in Form einer Taxonomie (also einer hierarchisch geordneten Aufzählung) vor. Dabei ist im Allgemeinen zwischen aufzählenden Klassifikationsschemata („Enumerated Classifications“ [Pr91]), die eine vollständige Menge von Einordnungen vorgeben, und aspektorientierten Klassifikationsschemata („Faceted Classifications“ [Pr91]), die zur Klassifizierung von Spezifikationen eingesetzt werden, zu unterscheiden. In Abbildung 5 finden sich sowohl Beispiele für aufzählende (z.B. Bezahlmethoden) als auch aspektorientierte (z.B. Währung) Klassifikationsschemata.

Domain (type: UN/SPSC - General Ledger and Accounting)
Component Type (type: specialized component) **Reuse Concept** (type: logical) **Component Model** (type: Microsoft .NET 1.0)

Abbildung 6: Spezifikation von Klassifikationen.

Prinzipiell ist es möglich, beliebige Klassifikationsschemata zu definieren, wodurch unter anderem der Aufbau betriebsinterner Komponentenkataloge unterstützt wird. Der UnSCom Spezifikationsrahmen gibt jedoch bereits eine Menge aufzählender Klassifikationsschemata vor, mit denen der Anwendungsbereich von Komponenten klassifiziert werden kann (z.B. UN/SPSC und NAICS). Darüber hinaus existieren weitere standardisierte Klassifikationsschemata, mit denen der konzeptionelle Komponententyp (Fachkomponente, Systemkomponente, Komponenten-Anwendungs-Framework, Komponenten-System-Framework, Anwendung etc. [Tu02]), das während der Implementierung verwendete Komponentenmodell sowie die Wiederverwendungsart (logisch, also als installierbares Artefakt zur Verfügung gestellt bzw. physisch, also als entfernter Dienst zur Verfügung gestellt [OT03]) spezifiziert werden können (siehe Abbildung 6).

3.3 Blue Pages: Spezifikation der fachlichen Funktionalität

Die „Blue Pages“ beschreiben die fachliche Funktionalität, die eine Komponente an ihren Schnittstellen benötigt (d.h. von anderen Komponenten erwartet) bzw. selbst für die Nutzung durch andere Komponenten bereitstellt. Die Spezifikation der fachlichen Funktionalität ist insbesondere für die Auswahl einer Komponente im Rahmen eines Entwicklungsvorhabens von wesentlicher Bedeutung [Fe97], [Si99]. Der UnSCom Spezifikationsrahmen beschreibt die fachliche Funktionalität einer Komponente in Form eines Lexikons [Or97], das auch als fachkonzeptuelles Modell dargestellt werden kann [CD00]. Ein solches Lexikon bzw. Modell beinhaltet fachliche Begriffsdefinitionen und Beziehungen zwischen den definierten (Fach-) Begriffen (siehe Abbildung 7).

Der Spezifikationsrahmen erlaubt die Spezifikation verschiedener Begriffsklassen, die in Anlehnung an die Referenzmodellierung [Sc98] als Informationsobjekte, Funktionen und Prozesse bezeichnet werden. Informationsobjekte beschreiben die von einer Komponente benötigten bzw. bereitgestellten fachlichen Informationen und deren Struktur. Funktionen beschreiben die von einer Komponente benötigten bzw. bereitgestellten fachlichen Aktionen. Prozesse beschreiben schließlich das fachliche Protokoll [Fe97], mit dem fachliche Aktionen zu einem Ablauf aneinandergereiht werden können.

Concept (type: information object): BALANCE **Short Definition**: A BALANCE yields a comparison of ASSETS and LIABILITIES of a company at a special date (CUTOFF DATE) based on a LEGAL REGULATION. **Relationship** (type: specialization): US-GAAP BALANCE is a BALANCE. **Relationship** (type: specialization): IAS BALANCE is a BALANCE.
Concept (type: function): ANNUAL ACCOUNTING **Short Definition**: ... **Relationship** (type: decomposition): ANNUAL ACCOUNTING consists of CLOSING OF ACCOUNTS. **Relationship** (type: decomposition): ANNUAL ACCOUNTING consists of BALANCING. **Relationship** (type: decomposition): ANNUAL ACCOUNTING consists of PROFIT & LOSS ACCOUNTING.
Concept (type: process): EXECUTE ANNUAL ACCOUNT **Short Definition**: ... **Relationship** (type: order): EXECUTE ANNUAL ACCOUNT starts with CLOSING OF ACCOUNTS. **Relationship** (type: order): CLOSING OF ACCOUNTS is followed in sequence by PROFIT & LOSS ACCOUNTING. **Relationship** (type: order): PROFIT & LOSS ACCOUNTING is followed in sequence by BALANCING.

Abbildung 7: Spezifikation fachlicher Funktionalität in Form eines Lexikons.

Darüber hinaus stellt der UnSCom Spezifikationsrahmen vordefinierte Beziehungsarten [Or97] zur Verfügung, mit denen Begriffe untereinander in Beziehung gesetzt werden können: Abstraktionen (Identität, Spezialisierung etc.) drücken ein gewisses Maß an Gleichheit zwischen Begriffen aus und können beispielsweise zur Bestimmung der (fachlichen) Kompatibilität von Komponenten verwendet werden. Kompositionen (Teil-Ganze-Beziehungen, Reihenfolgebeziehungen etc.) werden verwendet, um Begriffe zu zusammengesetzten Strukturen zu verbinden (siehe Abbildung 7).

Die Spezifikation der fachlichen Funktionalität einer Komponente entspricht bewusst der Vorgehensweise zur Definition eines Referenzmodells [Sc98]. Hierdurch ist einerseits die Möglichkeit gegeben, die fachliche Funktionalität einer Komponente durch einen Verweis auf existierende Referenzmodelle zu beschreiben, sofern die Komponente einen solchen fachlichen Standard unterstützt. Andererseits wird hierdurch auch das Entstehen solcher Referenzmodelle unterstützt, z.B. falls sich mehrere Hersteller von Komponenten darauf verständigen, dasselbe fachkonzeptuelle Modell als Bestandteil ihrer jeweiligen Komponentenspezifikation verwenden und es so zum fachlichen Standard werden lassen.

Von einem eher technischen Standpunkt aus betrachtet bilden die spezifizierten Lexika (bzw. fachkonzeptuellen Modelle) ihrer Struktur nach eine einfache Ontologie mit einer vordefinierten Menge von Beziehungsarten zwischen Begriffen. Durch die Interpretation einer fachkonzeptuellen Spezifikation als Ontologie wird es möglich, fachliche Funktionalität beispielsweise unter Rückgriff auf die Technologie des Semantic Web auch semantischen Suchverfahren bzw. automatisierten Kompatibilitätstests zugänglich zu ma-

chen. Sowohl die mit dem UnSCom Spezifikationsrahmen intendierte Förderung der Entstehung fachlicher Standards als auch die automatisierte Verarbeitung von Informationen über die fachliche Funktionalität einer Komponente tragen zur Erkennung und Beseitigung fachlicher Heterogenitäten zwischen Komponenten bei, die bislang meist nur schwer zu entdecken sind und eines der wesentlichen Hindernisse der komponentenbasierten Entwicklung von Informationssystemen darstellen [GAO95].

Die empfohlene textbasierte Notation zur Spezifikation fachlicher Funktionalität auf der Basis von Begriffen ist Normsprache [Or97], [Sc97], mit der sich Ontologien derart definieren lassen, dass sie sowohl maschinenlesbar als auch für Entwickler verständlich sind. Als Normsprache wird eine regulierte Form der natürlichen Sprache bezeichnet, in der beispielsweise Satzbaupläne für die Bildung von Sätzen vorgegeben sind, mit denen Begriffe in bestimmte Beziehungen zueinander gesetzt werden können (z.B. „A ist ein B“ für Spezialisierung). Wegen der angestrebten Internationalisierung des vorgeschlagenen Spezifikationsrahmens werden derzeit englische Satzbaupläne verwendet. Eine Lokalisierung ist jedoch (zumindest im westlichen Sprach- und Kulturkreis) vorstellbar.

Neben der Normsprache wurden auch andere, weiter verbreitete und komplexere Notationen (wie beispielsweise das Resource Description Framework RDF) auf ihre Einsetzbarkeit hin untersucht. Gerade die Normsprache unterstützt jedoch speziell die Lesbarkeit von Spezifikationen durch menschliche Anwender, ohne hierfür beispielsweise auf Werkzeuge zur Visualisierung angewiesen zu sein. Da sich solche Technologien (z.B. im Rahmen des entstehenden Semantic Web) derzeit noch in der Entwicklung befinden, ist dies (aktuell) ein entscheidender Vorteil. Neben dieser textbasierten Notation stellt UnSCom/G jedoch auch grafische Notationen zur Verfügung, die der Spezifikation der fachlichen Funktionalität von Komponenten (bei gleichem Inhalt) eine größere Ähnlichkeit mit der Darstellung von Referenzmodellen verleiht.

Die Spezifikation der fachlichen Funktionalität dient jedoch nicht nur den bereits genannten Zwecken. So können Fachbegriffe insbesondere auch zur näheren Beschreibung der systemtechnischen Architektur der Komponentenschnittstellen verwendet werden und angeben, welches systemtechnische Element der Schnittstelle eine bestimmte Funktionalität repräsentiert: Informationsobjekte können insbesondere zur Charakterisierung von Typdeklarationen, Attributen sowie Parametern und Rückgabewerten von Diensten verwendet werden. Mit Funktionen lassen sich Dienste bzw. Ereignisse charakterisieren, und (fachliche) Prozesse korrespondieren mit systemtechnischen Protokollen (Aufrufreihenfolgen). Damit können spezifizierte Fachbegriffe auch zur Erkennung und Beseitigung syntaktischer Heterogenitäten beitragen, beispielsweise wenn es systemtechnische Architekturelemente zu finden gilt, die die gleiche Funktionalität aufweisen aber unterschiedliche Signaturen (z.B. Namen, Parameterreihenfolgen etc.) besitzen [ZW95].

3.4 Green Pages: Spezifikation der systemtechnischen Architektur

Die „Green Pages“ beschreiben die systemtechnische Architektur der benötigten bzw. bereitgestellten Schnittstellen einer Komponente, die während des Systementwurfs festgelegt wurde. Diese Informationen sind für die korrekte Benutzung einer Komponente notwendig und können beispielsweise von Komponentenmodellen wie dem CORBA

Component Model [OM02] verarbeitet werden. Sowohl die statische als auch der grundlegende Teil der operationalen Architektur werden durch die Spezifikation von Signaturlisten beschrieben, die verschiedene Definitionen enthalten: Typ- und Konstantendeklarationen, Attribute, Dienste (Methoden), Ausnahmen und Ereignisse (siehe Abbildung 8). Die primäre Notation zur Spezifikation von Signaturlisten in UnSCom/T ist OMG Interface Definition Language (IDL) [OM02], die in der Version 3.0 auch die Spezifikation mehrerer benötigter bzw. bereitgestellter Schnittstellen ermöglicht.

```
interface IGeneralLedger {
  typedef string accountNo;
  typedef double quantity;
  struct account {
    accountNo n;
    quantity  safetyQuantity;
    quantity  reorderImgQuantity;
  };
  struct booking {
    accountNo n;
    date      executionDate;
    string    orderNo;
    double    bookingQuantity;
  };

  exception notEnoughMoney {};

  void book(in booking b);
  void reserve(in booking b) raises notEnoughMoney;
  quantity calculateBalance(in accountNo n, in date z);
};


interface IDatabase {
  exception openConncectionFailed {};
  exception openTransactionFailed {};
  exception invalidSql {};

  void openConnection(in string database) raises openConnectionFailed;
  void openTransaction() raises openTransactionFailed;
  void closeTransaction();
  void rollback();
  void executeSql(in string sql) raises invalidSql;
};
```

Abbildung 8: Spezifikation von Signaturlisten mit OMG IDL.

Die systemtechnischen Wirkungen der benötigten bzw. bereitgestellten Dienste können darüber hinaus durch die Spezifikation von Dienstverträgen näher beschrieben werden, die in Form von Vor- und Nachbedingungen (Zusicherungen) [Me97] den Signaturen der jeweiligen Dienste hinzugefügt werden. Obwohl sich Dienstverträge als sehr mächtige Spezifikationsmethode erwiesen haben, mit der man in der Lage ist, unterschiedlichste Sachverhalte auszudrücken, sollte bei der Verwendung dieser Methode stets bedacht werden, dass Dienstverträge während der Kompositionszeit in der Regel nur heuristisch ausgewertet und daher nur eingeschränkt für Aufgaben wie Kompatibilitätstests verwendet werden können [ZW97]. Die Ursache für diese Einschränkung liegt in der Komplexität der für eine exakte Auswertung notwendigen Algorithmen, die auf dem nur wenig effizient berechenbaren Prinzip des automatischen Beweisens („theorem proving“) basieren [Me97].

Die empfohlene Notation zur Spezifikation von Dienstverträgen ist Object Constraint Language (OCL) [OM96], eine formale Notation, die als Teil der Unified Modeling

Language (UML) standardisiert wurde. Eine Spezifikation in OCL benennt zunächst den Kontext, auf den sie sich bezieht, indem sie auf eine Dienst- oder Schnittstellenvereinbarung verweist. Im Anschluss daran werden die Zusicherungen, die für diesen Kontext gelten sollen, spezifiziert. Der in Abbildung 9 enthaltene Dienstvertrag besagt beispielsweise, dass die Methode calculateBalance nur für ein der Komponente bekanntes Konto aufgerufen werden darf. Die Nachbedingung garantiert, dass der Kontensaldo (der von der Methode calculateBalance zurückgeliefert wird) der Summe der Beträge aller bis dato auf diesem Konto ausgeführten Buchungen entspricht (wobei Minderungen als negative Zuwächse aufgefasst und in Folge dessen subtrahiert werden).

```
IGeneralLedger::calculateBalance(n:accountNo,z:date):quantity
  pre:  self.account->exists(k:account | k.accountNo = n)
  post: result = self.booking->iterate(b:booking; r:quantity = 0 |
          if b.accountNo = n and b.date <= z
          then
            r + b.bookingQuantity
          endif
        )
```

Abbildung 9: Spezifikation eines Dienstvertrags mit OCL.

Die Spezifikation der systemtechnischen Architektur von Komponentenschnittstellen wird durch die Angabe von Interaktionsprotokollen, die erlaubte Aufrufreihenfolgen für Dienste festlegen, vervollständigt. In der Regel können die von einer Komponente angebotenen Dienste nicht in einer beliebigen, sondern nur in einer wohlgeordneten Reihenfolge aufgerufen werden, die alleine auf der Basis spezifizierter Signaturlisten für den Entwickler jedoch nur schwer zu bestimmen ist. Darüber hinaus stellen Protokollspezifikationen eine wertvolle Information dar, die von vielen Methoden und Werkzeugen einer Entwicklungsmethodologie verwendet werden können (etwa zur Generierung von Protokolladaptern [YS97] oder zur Vorhersage von Eigenschaften des zu entwickelnden Informationssystems [RS02]).

Aus diesem Grund unterstützt der UnSCom Spezifikationsrahmen die Beschreibung systemtechnischer Interaktionsprotokolle und verwendet hierzu eine Notation, die auf endlichen regulären Automaten basiert [Ni93]. Diese Notation besitzt den Vorteil effizienter Auswertungsmöglichkeiten (z.B. für Kompatibilitätstests), birgt dafür aber auch eine Reihe von Nachteilen, da sowohl die Aussagekraft als auch die mit dieser Notation erzielbare Präzision von Spezifikationen eingeschränkt ist: so ist die Spezifikation von Aufrufreihenfolgen unter Einbeziehung der Parameter der aufgerufenen Dienste grundsätzlich unmöglich. Darüber hinaus beschreiben als endliche reguläre Automaten spezifizierte Interaktionsprotokolle im Allgemeinen nur eine Obermenge der erlaubten Aufrufreihenfolgen, enthalten also ggf. auch unerlaubte Aufrufreihenfolgen (für eine detaillierte Diskussion sei an dieser Stelle auf [Ni93] verwiesen). Dies ist insbesondere für die Spezifikation bereitgestellter Protokolle ein Mangel.

Derzeit wird daher an der Integration anderer Spezifikationstechniken in den Spezifikationsrahmen gearbeitet, die die genannten Mängel beheben (z.B. Petri-Netze [Pe62] oder temporale Logiken [CT00]). Ein noch zu bewältigendes Problem stellt hierbei jedoch die grundsätzliche Anforderung dar, neben der Ausdrucksmächtigkeit auch die effiziente

Auswertbarkeit und Nutzbarkeit von Protokollspezifikationen zur Kompositionszeit zu gewährleisten.

3.5 Grey Pages: Spezifikation der Qualität

Die „Grey Pages“ beinhalten die Spezifikation der an den Schnittstellen einer Komponente benötigten bzw. bereitgestellten Qualität. Unter dem Begriff „Qualität“ werden nicht-funktionale Eigenschaften einer Komponente (z.B. Zuverlässigkeit und Effizienz) subsumiert, die nach der Funktionalität die wichtigsten Auswahlkriterien für eine Komponente darstellen. Die Qualitätsmerkmale werden als ISO 9126 konforme Qualitätsattribute spezifiziert. Der Spezifikationsrahmen stellt hierfür eine auf dem ISO 9126-1 Standard [IS01] basierende Bibliothek standardisierter Qualitätsattribute zur Verfügung. Darüber hinaus legt er, unter Rückgriff auf den ISO 9126-2 Standard [IS03] auch die jeweiligen für die Messung zu verwendenden Metriken fest.

```
Type Library
…
type Efficiency = contract {
  responseTime: decreasing numeric msec;
  throughput:   increasing numeric calls/sec;
}
```

```
ServiceLevel normalQuality
…
normalGLProfile for IGeneralLedger = profile {
  from book require Efficiency contract {
    responseTime < 50 msec;
    throughput > 100 calls/sec;
  }
}
normalDBProfile for IDatabase = profile {
  from executeSql require Efficiency contract {
    responseTime < 20 msec;
    throughput > 200 calls/sec;
  }
}
```

Abbildung 10: Spezifikation von Qualitätsattributen mit QML.

Der Spezifikationsrahmen definiert drei Klassen für die Spezifikation von Qualitätsattributen: statische Qualitätsattribute bzw. Komponentenqualität (Anwendbarkeit, Wartbarkeit und Übertragbarkeit), operationale Qualitätsattribute bzw. Dienstqualität (Funktionalität und Zuverlässigkeit) und dynamische Qualitätsattribute bzw. Performanz (Effizienz). Im Gegensatz zur Funktionalität und zur systemtechnischen Architektur lassen sich die benötigten und bereitgestellten Qualitätsattribute jedoch nicht als konstant auffassen. Zunächst hängen die bereitgestellten Qualitätsattribute nicht nur, wie im Kompositionsvertrag angenommen, von den Qualitätsattributen ab, die von anderen Komponenten zur Verfügung zu stellen sind, sondern auch von der Hardwareumgebung, in der die Komponente installiert wird. Darüber hinaus toleriert eine Komponente gewisse Schwankungen der Hardwaremerkmale und der benötigten Qualitätsattribute, ohne deshalb grundsätzlich in ihrer Funktionalität beeinträchtigt zu sein. Jedoch werden durch diese Schwankungen die von der Komponente bereitgestellten Qualitätsattribute beeinträchtigt, die deshalb ebenfalls veränderlich sind. Um diese Schwankungsmöglichkeiten (zumindest näherungsweise) abzudecken, erlaubt der Spezifikationsrahmen zur Be-

schreibung der Qualität von Komponenten die Spezifikation mehrerer Qualitätsniveaus (Service-Level), die die jeweils bereitgestellte Qualität in Abhängigkeit von der Hardwareumgebung und der von anderen Komponenten gebotenen Qualität beschreiben. Hierdurch bestehen die Qualitätsebenen jedoch nicht mehr nur aus einer einzigen, sondern aus mehreren alternativen vertraglichen Vereinbarungen.

Die empfohlene Notation zur Spezifikation von Qualitätsattributen ist die Quality Modeling Language (QML) [FK98], die sowohl die Spezifikation von Qualitätsattributen in Form von *Vertragstypen*, als auch die Spezifikation konkreter Messungen in Form von *Verträgen* erlaubt (siehe Abbildung 10). Der Spezifikationsrahmen definiert eine Standardbibliothek ISO 9126 konformer QML Vertragstypen, die zur Spezifikation von Messungen verwendet werden können. Darüber hinaus ist die Definition weiterer Vertragstypen möglich. Da QML die Spezifikation der Hardwareumgebung nicht unterstützt, führt der Spezifikationsrahmen zusätzlich eine Reihe von Taxonomien ein, um diese (als Teil eines Qualitätsniveaus) zu dokumentieren.

Qualitätsattribute lassen sich in ähnlicher Form wie Fachbegriffe zur Charakterisierung von Elementen der systemtechnischen Architektur (vorzugsweise der dort definierten Dienste) einsetzen. Auf diese Weise lässt sich beschreiben, welches systemtechnische Element (welcher Dienst) gewisse Qualitätsattribute besitzt. Diese Beziehung zwischen Qualitätsattributen und Architekturelementen wird von QML direkt unterstützt, das hierfür *Qualitätsprofile* zur Verfügung stellt (siehe Abbildung 10).

4 Relevante Arbeiten aus dem wissenschaftlichen Umfeld

Die Wichtigkeit von Komponentenspezifikationen wird in der Literatur allgemein anerkannt und ist bereits von zahlreichen Autoren besprochen worden. Häufig wird in den entsprechenden Arbeiten die Verwendung formaler Notationen betont [Sz03], [Ki99], [Fe97]. Darüber hinaus existieren Arbeiten, die verschiedene Spezifikationsebenen unterscheiden und hierfür Notationen vorschlagen [Sz03], [Be99], [CT00]. Die vorgenannten Arbeiten vernachlässigen dabei die Spezifikation der fachlichen Funktionalität, die für die Auswahl einer Komponente jedoch das wesentliche Kriterium darstellt [Fe97]. Diese wird erst in neuerer Zeit in entsprechenden Arbeiten berücksichtigt, die sich jedoch ausschließlich auf die Spezifikation von Komponenten der betrieblichen Anwendungsbereiche (Fachkomponenten) konzentrieren [Tu02], [CD00].

Dieser Beitrag baut auf den genannten Arbeiten auf und rechtfertigt vorgeschlagene Spezifikationsebenen über ein einheitliches Klassifikationsschema. Hierdurch wird zum einen die Ableitung von Spezifikationsebenen modelltheoretisch begründet, und zum anderen eine stabile Struktur erreicht, die ein ausgewogenes Bündel von Spezifikationsebenen zur Verfügung stellt. Darüber hinaus stellt er einen einheitlichen Spezifikationsrahmen für Modellierung von Komponenten aller Anwendungsbereiche zur Verfügung. Dies ist auch für die betriebliche Informationssystementwicklung von Vorteil, da Fachkomponenten bekanntermaßen immer nur eine Teilmenge der einzusetzenden Komponenten darstellen [Tu02].

Neben den vorgenannten Arbeiten wird in der Literatur zur Komponentenorientierung vor allem in neuerer Zeit die Wichtigkeit der Spezifikation der benötigten und bereitgestellten Schnittstellen betont [CD00], [DW99], die aus dem Bereich der Software-Architekturen schon seit längerem bekannt sind und dort zur vollständigen Beschreibung gekapselter Module verwendet werden [DK76], [SG96]. Die vorliegende Arbeit berücksichtigt diese Entwicklung und setzt darüber hinaus beide Schnittstellenarten durch die Einführung des Kompositionsvertrags in eine vertragliche Beziehung. Hierdurch wird es vor allem möglich, Konzepte des etablierten Softwarevertrags [Me97] auch in der komponentenbasierten Informationssystementwicklung zu verwenden (siehe Kapitel 5).

Gegenwärtig hat trotz der anerkannten Bedeutung von Komponentenspezifikationen nur ein Rahmen den Rang eines Standards erlangt. Dieser Rahmen trägt den Namen „Universal Description, Discovery, and Integration“ (UDDI), ist auf die Spezifikation von XML Web-Services beschränkt und Teil der in diesem Bereich entstehenden Methodologie [UD00]. Der in dieser Arbeit vorgeschlagene Spezifikationsrahmen wurde entwickelt, um diese Lücke zu füllen und die Entstehung einer Entwicklungsmethodologie für die komponentenorientierte Informationssystementwicklung zu unterstützen.

5 Ausblick

Der vorgeschlagene Spezifikationsrahmen wird derzeit in der Praxis der betrieblichen Informationssystementwicklung auf seine Anwendbarkeit hin evaluiert. Darüber hinaus befinden sich Werkzeuge zur Unterstützung des Spezifikationsprozesses sowie Komponentenkataloge (Unternehmensrepositorien und Komponentenmarktplätze) in der Entwicklung. Im Bereich des Spezifikationsrahmens wird derzeit hauptsächlich nach einer geeigneten Methode zur Spezifikation von Interaktionsprotokollen geforscht (siehe Abschnitt 3.4). Darüber hinaus gilt es, Parametrisierte Kompositionsverträge [RS02], die Vorteile bei der Vorhersage von Informationssystemeigenschaften aus Komponenteneigenschaften versprechen, in den Rahmen einzubinden.

Der Schwerpunkt zukünftiger Arbeiten stellt jedoch die Schaffung einer Entwicklungsmethodologie auf Basis des vorgestellten Spezifikationsrahmens dar. Das eingeführte Prinzip des Kompositionsvertrags ermöglicht es hierbei, die zu unterstützenden Aufgaben (siehe Abbildung 1) einheitlich zu formulieren und Antworten auf die eingangs gestellten Fragen zu geben. So lassen sich Kompatibilitätstest zwischen zwei Komponenten auf (weitgehend erforschte) Kompatibilitätstests von Schnittstellen zurückführen und auf den eingeführten Vertragsebenen durchführen: Zwei Komponenten A und B können miteinander verbunden werden (sind kompatibel), falls mindestens eine bereitgestellte Schnittstelle von B kompatibel zu (also eine Spezialisierung von) einer der benötigten Schnittstellen von A ist.

Die Ersetzbarkeit von Komponenten lässt sich durch Vertragsspezialisierung („Subcontracting“ [Me97]) definieren: Die ersetzende Komponente B darf mehr (speziellere Schnittstellen) bereitstellen und weniger (generellere Schnittstellen) benötigen, ohne den ursprünglichen Vertrag zu verletzen. Ähnlich lässt sich Adaptergenerierung als Analyse der Unterschiede zwischen den benötigten Schnittstellen einer Komponente und den

bereitgestellten Schnittstellen einer anderen Komponente verstehen. Schließlich kann auch die Vorhersage von Informationssystemeigenschaften auf der Basis der Verkettung (Komposition) der Kompositionsverträge methodisch unterstützt werden.

Es steht zu erwarten, dass die Schaffung einer Entwicklungsmethodologie auf der Basis von Kompositionsverträgen das Potenzial besitzt, einen Beitrag zur Lösung der bestehenden Herausforderungen zu leisten, einer möglichen Kompositionskrise entgegenzuwirken und so die komponentenbasierte Informationssystementwicklung auf dem Weg zum Standardansatz der betrieblichen Softwareentwicklung ein Stück voranzubringen.

Literaturverzeichnis

[Be99] Beugnard, A. et al.: Making Components Contract Aware. In: IEEE Computer 32 (1999) 7; S. 38-45.

[Br00] Brown, A. W.: Large-Scale Component-Based Development. Prentice Hall, Upper Saddle River, New Jersey, 2000.

[CD00] Cheesman, J.; Daniels, J.: UML Components: A Simple Process for Specifying Component-Based Software. Addison Wesley, Upper Saddle River, New Jersey, 2000.

[CD94] Cook, S.; Daniels, J.: Designing Object Systems. Object-Oriented Modeling with Syntropy. Prentice Hall, Englewood Cliffs, 1994.

[CT00] Conrad, S.; Turowski, K.: Vereinheitlichung der Spezifikation von Fachkomponenten auf der Basis eines Notationsstandards. In (Ebert, J.; Frank, U. Hrsg.): Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik. Fölbach, Koblenz, 2000; S. 179-194.

[Da93] Davis, A. M.: Software Requirements: Objects, Functions, and States. Prentice Hall, Englewood Cliffs, 1993.

[DK76] DeRemer, F.; Kron, H.H.: Programming-in-the-Large versus Programming-in-the-Small. In: IEEE Transactions on Software Engineering 2 (1976) 2; S. 80-86.

[DW99] D'Souza, D.; Wills, A. C.: Objects, Components, and Frameworks with UML: The Catalysis Approach. Addison Wesley, Reading, Massachusetts, 1999.

[Fe97] Ferstl, O. K. et al.: Bausteine für komponentenbasierte Anwendungssysteme. In: HMD 34 (1997) 197; S. 24-46.

[FK98] Frolund, S.; Koistinen, J.: QML: A Language for Quality of Service Specification. Technical Report HPL-98-10, Hewlett-Packard Laboratories, 1998.

[GAO95] Garlan, D.; Allen, R.; Ockerbloom, J.: Architectural Mismatch: Why Reuse Is So Hard. In: IEEE Software 12 (1995) 6; S. 17-26.

[Gr94] Graham, I.: Migrating to Object Technology. Addison Wesley, Wokingham, 1994.

[He93] Heinrich, L. J.: Wirtschaftsinformatik. Einführung und Grundlegung. Oldenburg, München, 1993.

[IS01] International Standardization Organization (Hrsg.): Software Engineering - Product Quality - Quality Model, ISO/IEC Standard 9126-1, 2001.

[IS03] International Standardization Organization (Hrsg.): Software Engineering - Product Quality - External Metrics, ISO/IEC Standard 9126-2, 2003.

[Ki99] Kiniry, J. R.: Leading to a Kind Description Language: Thoughts on Component Specification. Tech. Rep. CS-TR-99-04, California Institute of Technology, 1999.

[Me97] Meyer, B.: Object-Oriented Software Construction. 2. Auflage, Prentice Hall, Englewood Cliffs, New Jersey, 1997.

[Ni93] Nierstrasz, O.: Regular Types for Active Objects. In: Proceedings of the OOPSLA 93. ACM SIGPLAN Notices 28 (1993) 10; S. 1-15.
[Ol91] Olle, T. W. et al.: Information Systems Methodologies: A Framework for Understanding. Addison Wesley, Wokingham, 1991.
[OM96] Object Management Group (Hrsg.): Object constraint language, Specification, Version 1.1, 1997.
[OM02] Object Management Group (Hrsg.): CORBA Components, Specification, Version 3.0, 2002.
[Or97] Ortner, E.: Methodenneutraler Fachentwurf. Teubner, Stuttgart, 1997.
[OT03] Overhage, S.; Thomas, P.: WS-Specification: Specifying Web Services Using UDDI Improvements. In (Chaudri, A. B.; Jeckle, M.; Rahm, E.; Unland, R. Hrsg.): Web, Web Services, and Database Systems. Lecture Notes in Computer Science (LNCS 2593), Springer, Berlin, 2003; S. 100-118.
[Pe62] Petri, C. A.: Fundamentals of a Theory of Asynchronous Information Flow. In: Information Processing 62, IFIP, 1962; S. 386.391.
[Pr91] Prieto-Diáz, R.: Implementing Faceted Classification for Software Reuse. In: Communications of the ACM 34 (1991) 5; S. 89-97.
[RS02] Reussner, R. H.; Schmidt, H.W.: Using Parameterised Contracts to Predict Properties of Component-Based Software Architectures. In (Crnkovic, I., Larsson, S., Stafford, J. Hrsg.): Workshop on Component-Based Software Engineering, 2002.
[Sc98] Scheer, A.-W.: ARIS. Vom Geschäftsprozess zum Anwendungssystem. 3. Auflage, Springer, Berlin, 1998.
[Sc97] Schienmann, B.: Objektorientierter Fachentwurf. Ein terminologiebasierter Ansatz für die Konstruktion von Anwendungssystemen. Teubner, Stuttgart, 1997.
[SG96] Shaw, M.; Garlan, D.: Software Architecture – Perspectives on an Emerging Discipline. Prentice Hall, Upper Saddle River, New Jersey, 1996.
[Si99] Sinz, E. J.: Anwendungssysteme aus fachlichen Komponenten. In: Wirtschaftsinformatik 41 (1999) 1; S. 3.
[Sz03] Szyperski, C.: Component Software – Beyond Object-Oriented Programming. 2. Auflage, Addison-Wesley, Harlow, 2003.
[Tu02] Turowski, K. (Hrsg.): Vereinheitlichte Spezifikation von Fachkomponenten – Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme, 2002.
[UD00] UDDI Organization (Hrsg.): UDDI Technical White Paper. UDDI Standards Organization, 2000.
[Wa03] Wallnau, K. C.: A Technology for Predictable Assembly from Certifiable Components, Tech. Rep. CMU/SEI-2003-TR-009, Software Engineering Institute, 2003.
[We01] Weyuker, E. J.: The Trouble with Testing Components. In (Councill, W. T.; Heineman, G. T. Hrsg.): Component-Based Software Engineering. Putting the Pieces Together. Addison Wesley, Upper Saddle River, New Jersey, 2001; S. 499-512.
[YS97] Yellin, D.; Strom, R.: Protocol Specifications and Component Adaptors. In: ACM Transactions on Programming Languages and Systems 19 (1997) 2; S. 292-333.
[ZW95] Zaremski, A. M.; Wing, J. M.: Signature Matching: A Tool for Using Software Libraries. In: ACM Transactions on Software Engineering and Methodology 4 (1995) 2; S. 146-170.
[ZW97] Zaremski, A. M.; Wing, J. M.: Specification Matching of Software Components. In: ACM Transactions on Software Engineering and Methodology 6 (1997) 4; S. 333-369.