

# Vereinheitlichte Spezifikation von Komponenten: Grundlagen, UNSCOM Spezifikationsrahmen und Anwendung

Sven Overhage

Arbeitsgruppe Component und Service Engineering  
Lehrstuhl für Wirtschaftsinformatik und Systems Engineering  
Universität Augsburg, Universitätsstraße 16, 86159 Augsburg  
sven.overhage@wiwi.uni-augsburg.de

**Abstract:** In diesem Beitrag wird ein Spezifikationsrahmen vorgestellt, mit dem sich die Außensicht von Software-Komponenten in normierter Weise beschreiben lässt. Der vorgestellte Spezifikationsrahmen schafft eine wichtige Grundlage zur Unterstützung des komponentenorientierten Entwicklungsprozesses. Er basiert auf dem Konzept des Software-Vertrags und beschreibt relevante Eigenschaften von Komponenten auf verschiedenen, systematisch hergeleiteten Vertragsebenen. Diese Ebenen sind thematisch gruppiert und stellen Informationen über folgende Eigenschaften einer Komponente bereit: Allgemeine und kommerzielle Merkmale (White Pages), Klassifikationen (Yellow Pages), fachliche Funktionalität (Blue Pages), logische Architektur (Green Pages) und physische Qualität (Grey Pages). Dabei werden Spezifikationsmethoden der Wirtschaftsinformatik und des Software Engineering zu einem Ansatz zusammengeführt.

## 1 Motivation

Das im Vergleich zur traditionellen Anwendungsentwicklung effizient anmutende komponentenorientierte Entwicklungsparadigma, demzufolge Anwendungen durch *Auswählen* und *Koppeln* geeigneter Komponenten zu entwickeln sind, wird mit einer Vielzahl von Vorteilen verbunden, die maßgeblich zur Bewältigung der fortdauernden Software-Krise beitragen sollen [McI68]. So wird u.a. eine verkürzte Entwicklungszeit, eine erhöhte Flexibilität der Anwendungen und eine deutliche Senkung der Entwicklungskosten durch Wiederverwendung existierender Lösungen vorhergesagt [Sam97, Bro00, CD01, SGM02].

Die Umsetzung dieser Vision hängt davon ab, dass es gelingt, den mit dem neuen Entwicklungsparadigma eingeführten *Kompositionsprozess* methodisch zu unterstützen. Hierfür sind zahlreiche Herausforderungen zu lösen und insbesondere geeignete Methoden bzw. Werkzeuge für die *Suche* nach Komponenten, die Durchführung von *Kompatibilitätstests*, die *Adaptergenerierung* sowie die *Vorhersage* der Qualität von Anwendungen, die durch Kopplung der ausgewählten Komponenten entstehen, bereitzustellen [ZW97, Crn02].

Zum zentralen Erfolgsfaktor für die Entwicklung solcher Methoden bzw. Werkzeuge wird dabei die Verfügbarkeit von *Spezifikationen*, die die von einer Komponente angebotenen

Dienste und die bei der Inanspruchnahme zu erfüllenden Bedingungen in angemessener Weise beschreiben [DW99, Bro00, SGM02, Wal03]. Ohne die Verfügbarkeit solcher Informationen könnte die Komposition von Komponenten erst nach einer Analyse ihrer jeweiligen Eigenschaften erfolgen, die jedoch durch das zugrunde liegende Black-Box Prinzip erschwert wird. Die zur Analyse einzusetzenden Techniken des Testens bzw. Reverse Engineering verursachen einen hohen Aufwand und sind kaum in der Lage, akkurate Informationen zu liefern. Damit würden die in Aussicht gestellten Effizienzgewinne der Komponentenorientierung beeinträchtigt oder gar zunichte gemacht [GAO95, Wey01].

Diesem speziellen Umstand wurde durch verschiedene Ansätze Rechnung getragen, die Konzepte und Methoden zur Spezifikation von Komponenten vorschlagen [DW99, CD01, OMG03]. Jedoch sind solche Ansätze meist einseitig auf die Unterstützung der Komponentenentwicklung ausgelegt und vernachlässigen folglich die Anforderungen, die sich aus dem beabsichtigten Einsatz von Spezifikationen im Rahmen des Kompositionsprozesses ergeben. So bleibt bspw. die in [VZJ03] geforderte Spezifikation der fachlichen Funktionalität von Komponenten ebenso unberücksichtigt wie die Beschreibung nicht-funktionaler Eigenschaften. Ferner existieren Ansätze, deren Ziel eine möglichst vollständige Beschreibung von Komponenten ist [BJPW99, SGM02]. Sie beschränken sich jedoch darauf, verschiedene Aspekte von Spezifikationen zusammenzutragen und verzichten auf konkrete Vorgaben hinsichtlich des Inhalts oder der zur Darstellung einzusetzenden Formate.

Im Rahmen dieses Beitrags wird deshalb ein Spezifikationsrahmen zur Beschreibung von Komponenten vorgestellt, der sowohl die zu berücksichtigenden Eigenschaften als auch die zur Spezifikation einzusetzenden Notationen normiert. Er trägt den Namen *Vereinheitlichte Spezifikation von Komponenten* (engl. Unified Specification of Components, UNSCOM) und basiert auf dem Konzept des *Software-Vertrags* (Design by Contract), das er für den Einsatz in der komponentenorientierten Anwendungsentwicklung erweitert. Mit seinen systematisch hergeleiteten Vertragsebenen umfasst der vorgestellte Spezifikationsrahmen bereits existierende Ansätze und ist in der Lage, diese abzulösen. Er erfüllt eine Reihe grundlegender Anforderungen, die in Kap. 2 skizziert werden. Der UNSCOM Spezifikationsrahmen wird anschließend in Kap. 3 beschrieben. Der Beitrag schließt mit einem Ausblick auf weitere Arbeiten, die auf dem Spezifikationsrahmen aufsetzen.

## 2 Grundlegende Anforderungen

Bei der komponentenorientierten Anwendungsentwicklung kommt der Spezifikation von Komponenten eine zentrale Bedeutung zu. Für die Komponentenentwicklung ist sie *präskriptiv*, d.h. sie gibt die zu realisierenden Eigenschaften als Anforderungen vor. Im Kompositionsprozess ist sie *deskriptiv*, d.h. sie stellt Informationen über die Eigenschaften von Komponenten bereit. Obwohl Spezifikationen damit die methodische Basis zur Unterstützung der Auswahl und Kopplung von Komponenten bilden, gibt es bislang kaum wissenschaftliche Kriterien, mit denen die Eignung der Vorgaben eines Spezifikationsrahmens für den Kompositionsprozess beurteilt werden kann (zur Diskussion vgl. [Ove06]). Dennoch lassen sich anhand der Aufgaben, die mit den bereitgestellten Informationen zu unterstützen sind (vgl. Kap. 1), einige grundlegende Anforderungen ableiten:

- **Angemessenheit.** Der Spezifikationsrahmen strebt ein Gleichgewicht zwischen Ausdrucksmächtigkeit und statischer Auswertbarkeit an. Es entsteht ein Kompromiss zwischen der Präzision einer Spezifikation und ihrer Verwendbarkeit bei Tests bzw. Auswertungen, die zur Entwicklungszeit vorgenommen werden.
- **Vollständigkeit.** Der Spezifikationsrahmen muss alle für die Auswahl und Koppung relevanten Eigenschaften von Komponenten beschreiben, so dass es möglich ist, die zu bewältigenden Aufgaben auf Basis der Spezifikationen auszuführen.
- **Abgestimmtheit.** Der Spezifikationsrahmen muss eine explizite Einordnung aller zu spezifizierenden Sachverhalte vornehmen und dabei auch die Zusammenhänge zwischen einzelnen Spezifikationsteilen klären.
- **Verbindlichkeit.** Der Spezifikationsrahmen schreibt sowohl den Inhalt als auch die einzusetzenden Notationen verbindlich vor. Damit wird die Einheitlichkeit und Vergleichbarkeit von Spezifikationen gewährleistet.
- **Verständlichkeit.** Erstellte Spezifikationen sind von Entwicklern und Werkzeugen auswertbar. Für die Werkzeugunterstützung ist die Bereitstellung formaler Beschreibungen unerlässlich. Diese müssen jedoch auch für Entwickler verständlich bleiben.
- **Universalität.** Der Spezifikationsrahmen muss unabhängig von Komponententechnologien und Komponentenmodellen verwendbar sein.
- **Veränderbarkeit.** Der Spezifikationsrahmen besitzt eine modulare Struktur, so dass neue Inhalte abwärtskompatibel hinzugefügt werden können.
- **Praktikabilität.** Zur Spezifikation werden etablierte und industriefähige Notationen verwendet, auch wenn ggf. ein gegenüber dem Stand der Wissenschaft suboptimales Ergebnis erzielt wird. Diese Anforderung sichert die Akzeptanz in der Praxis.

### 3 Der UNSCOM Spezifikationsrahmen

Im Folgenden wird mit dem UNSCOM Spezifikationsrahmen ein Ansatz für die Beschreibung von Komponenten dargestellt, der den in Kap. 2 genannten grundlegenden Anforderungen genügt [Ove06]. Er basiert auf dem etablierten Prinzip des *Software-Vertrags* (Design by Contract) [Mey92], das derzeit vor allem in der objektorientierten Programmierung Anwendung findet und dort zur Spezifikation sog. *Dienstverträge* verwendet wird. Ein Dienstvertrag beschreibt die Vorbedingungen, die von einem Dienstanbieter zur *Laufzeit* erfüllt werden müssen, damit dieser einen Software-Dienst in Anspruch nehmen kann, sowie die Nachbedingungen, die vom Dienstnehmer unter der Voraussetzung garantiert werden, dass der Dienst mit erfüllten Vorbedingungen in Anspruch genommen wurde [Mey92]. Damit lassen sich prinzipiell auch die Dienste von Komponenten beschreiben.

Für die komponentenorientierte Entwicklung ist die Spezifikation von Dienstverträgen jedoch nicht ausreichend. Da Komponenten im Gegensatz zu Klassen als eigenständige Entwicklungsergebnisse [SGM02] anzusehen sind, müssen Entwickler eine genaue Kenntnis

darüber besitzen, welche Dienste eine Komponente anbietet bzw. von ihrer Umgebung erwartet. In der Regel bietet eine Komponente nämlich nur dann Dienste an, wenn die von ihr im Gegenzug nachgefragten Dienste auch zur Verfügung stehen. Somit besitzt eine Komponente neben Dienstverträgen noch einen *Komponentenvertrag*, der die zur *Kompositionszeit* zu erfüllenden Bedingungen beschreibt, unter denen eine Komponente ihre Dienste überhaupt zur Verfügung stellt [CD01, Ove06]. Ein Komponentenvvertrag spezifiziert Schnittstellen mit Diensten, die von der Umgebung zur Verfügung zu stellen sind, und beschreibt Schnittstellen mit Diensten, die von der Komponente unter der Voraussetzung angeboten werden, dass die von ihr nachgefragten Dienste nutzbar sind [DW99, CD01].

|                                  | Funktionalität / Fachbegriffe (fachlich) | Architektur / Beschaffenheit (logisch)            | Implementierung / Qualität (physisch)     |
|----------------------------------|--|---|---|
| Statische Sicht (Struktur)       | Informationsobjekte (Dinge)              | Signaturen (Komponente, Schnittstellen, Methoden) | Verwendbarkeit, Wartbarkeit, Portabilität |
| Operationale Sicht (Wirkungen)   | Funktionen (Handlungen)                  | Zusicherungen                                     | Funktionalität                            |
| Dynamische Sicht (Interaktionen) | Prozesse (Abläufe)                       | Interaktionsprotokolle                            | Zuverlässigkeit, Effizienz                |

Abbildung 1: Klassifikation von Eigenschaften der Komponentenaußensicht.

Ein solcher Komponentenvvertrag lässt sich prinzipiell auf verschiedenen Vertragsebenen spezifizieren, wobei die Schnittstellen (bzw. die dort aufgeführten Dienste) aus jeweils unterschiedlichen Perspektiven beschrieben werden [BJPW99, SGM02, Ove06]. Mit dem UNSCOM Spezifikationsrahmen wird eine möglichst vollständige Beschreibung angestrebt, weshalb die in der Spezifikation zu berücksichtigenden Vertragsebenen unter Verwendung eines *Klassifikationsschemas* (vgl. Abb. 1) systematisch hergeleitet wurden. Das Klassifikationsschema unterscheidet in der Horizontalen die während des Entwicklungsprozesses auftretenden *Abstraktionsebenen* und in der Vertikalen die zur vollständigen Beschreibung von zahlreichen Methodiken [OHMR91, CD94, Sch98, DW99] verwendeten Modellierungs- bzw. *Spezifikationssichten*. Letztere lassen sich aus der Allgemeinen Systemtheorie ableiten und damit modelltheoretisch begründen (vgl. [Rop99, Ove06]).

Jede der in der Horizontalen beschriebenen Abstraktionsebenen beschreibt einen anderen Spezifikationsaspekt: die fachliche Abstraktionsebene beschreibt die *Funktionalität* einer Komponente und nutzt hierzu domänenspezifische Begriffe, die Sachverhalte des Anwendungsbereichs beschreiben und als Lexikon bzw. konzeptuelles Modell spezifiziert werden [Sch98, DW99]. Die logische Abstraktionsebene beschreibt die systemtechnische *Architektur* der Komponente und nutzt dafür Signaturlisten, Dienstverträge und Interaktionsprotokolle [BJPW99]. Die physische Abstraktionsebene beschreibt die *nicht-funktionalen Eigenschaften* einer Komponente und greift hierfür auf Qualitätsattribute zurück, die aus dem ISO 9126 Standard [ISO01, ISO03] abgeleitet wurden. Durch die in der Vertikalen unterschiedenen Spezifikationssichten wird jede Abstraktionsebene weiter differenziert. Dabei fokussiert die statische Sicht auf die *Struktur*, die operationale Sicht beschreibt die *Wirkungen* und die dynamische Sicht spezifiziert die *Interaktionen*.

Die mithilfe des Klassifikationsschemas ermittelten neun Vertragsebenen bilden den Kern des UNSCOM Spezifikationsrahmens. Er wird durch zwei weitere Vertragsebenen ver-

vollständig, die die Angabe allgemeiner und kommerzieller Informationen sowie die Klassifikation von Komponenten ermöglichen (vgl. Abb. 2 links). Die Ebenen sind thematisch gruppiert und verschiedenen Kategorien zugeordnet, die in Anlehnung an UDDI [UDD00] als farbige Seiten unterschieden werden: *White Pages* beinhalten allgemeine Informationen, *Yellow Pages* bestehen aus Klassifikationen, *Blue Pages* beschreiben die Funktionalität aus fachlicher Sicht, *Green Pages* spezifizieren architekturenspezifische Merkmale und *Grey Pages* beschreiben die Qualitätseigenschaften einer Komponente.

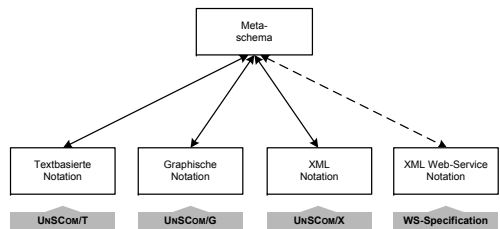
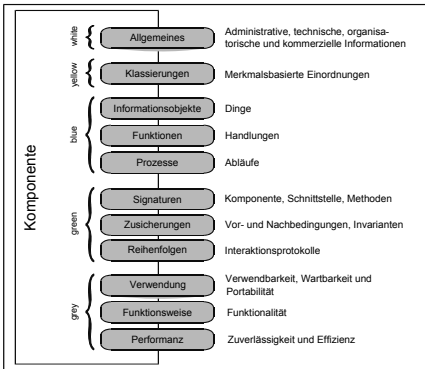


Abbildung 2: Thematische Gruppierung von Vertragsebenen und UNSCOM Spezifikationsformate.

Der UNSCOM Spezifikationsrahmen sieht auf den elf Ebenen verschiedene formale, plattformunabhängig einsetzbare Notationen für die Darstellung von Komponentenspezifikationen vor. Dabei werden verschiedene Darstellungsformen unterstützt und durch ein gemeinsames Metaschema mit inhaltlichen Vorgaben integriert (vgl. Abb. 2 rechts): UNSCOM/T besteht aus textbasierten Notationen, UNSCOM/G bildet ein graphisches Format auf Basis des UML 2 Standards, UNSCOM/X stellt XML Datenformate für den Austausch zwischen Werkzeugen zur Verfügung und WS-Specification [OT05] unterstützt die Beschreibung von XML Web-Services mit den dort standardisierten Notationen.

### 3.1 White und Yellow Pages: Allgemeine Informationen und Klassifikationen

Auf den White Pages sind allgemeine und kommerzielle Informationen über eine Komponente zusammenzutragen. Mit diesen Informationen kann die Eignung einer Komponente für ein Entwicklungsvorhaben aus wirtschaftlicher Sicht beurteilt werden. Zu den *allgemeinen Informationen* gehören der Name der Komponente, die Version, eine Beschreibung sowie Angaben zum Hersteller. Zu den *kommerziellen Informationen* gehören die Kauf- und Nutzungsbedingungen, die sich aus den verschiedenen Bezugsformen (Preis, Vertriebsform, Lieferumfang und akzeptierte Bezahlverfahren) und rechtlichen Vereinbarungen (Lizenzvereinbarungen, Garantievereinbarungen etc.) zusammensetzen.

Die Yellow Pages ermöglichen die Angabe von *Klassifikationen*, mit denen sich Komponenten gemäß einem standardisierten, facettenorientierten Klassifikationsschema [PD91]

in Katalogen ablegen lassen (vgl. UDDI [UDD00]). Das Klassifikationsschema des UNSCOM Spezifikationsrahmens sieht die Angabe des Anwendungsbereichs, der Komponententechnologie sowie die Art der Wiederverwendung (Prozedurfernaufruf vs. Auslieferung) vor und stellt hierfür jeweils spezielle, standardisierte Schlüsselworte zur Verfügung.

### 3.2 Blue Pages: Spezifikation der fachlichen Funktionalität

Mit den Blue Pages lässt sich die fachliche Funktionalität von Diensten beschreiben, die von einer Komponente entweder angeboten oder nachgefragt werden. Hierzu ist ein *Begriffssystem* zu spezifizieren, das die der Implementierung zugrunde liegende Fachsemantik als *Ontologie* beschreibt. Dabei sind die Begriffe eines Anwendungsbereichs zunächst zu definieren und anschließend in Beziehung zueinander zu setzen (vgl. Abb. 3). Die so spezifizierte Funktionalität ist bei der Auswahl von Komponenten von zentraler Bedeutung [VZJ03]. In Anlehnung an die Referenzmodellierung unterscheidet der UNSCOM Spezifikationsrahmen drei verschiedene *Begriffsklassen*: Informationsobjekte repräsentieren Dinge (Entitäten) aus dem Anwendungsbereich, Funktionen charakterisieren Handlungen und Prozesse stehen für komplexe Abläufe [Sch98]. Daneben werden vordefinierte *Beziehungstypen* bereitgestellt, mit denen Begriffe zueinander in Beziehung gesetzt werden können: Abstraktionen (Identität, Spezialisierung etc.) drücken eine gewisse Gleichheit zwischen Begriffen aus, während Kompositionen (Teil-Ganze-Beziehungen, Reihenfolgen etc.) verwendet werden, um Begriffe zu zusammengesetzten Strukturen zu verbinden.

|   |  |
|---|--|
| <b>Terminologie</b> (Typ: Lexikon)                        |  |
| <b>Begriff</b> (Typ: Informationsobjekt)                  | <b>Begriffswort:</b> Lager   |
|   | <b>Kurzdefinition:</b> Ein Lager ist ein Ort, an dem Artikel aufbewahrt werden.  |
| <b>Begriff</b> (Typ: Informationsobjekt)                  | <b>Begriffswort:</b> Lagerbereich  |
|   | <b>Kurzdefinition:</b> Ein Lagerbereich ist ein Ort, an dem Artikel zu einem bestimmten Zweck aufbewahrt werden.                 |
| <b>Begriff</b> (Typ: Informationsobjekt)                  | <b>Begriffswort:</b> Kommissionierbereich  |
|   | <b>Kurzdefinition:</b> Ein Kommissionierbereich ist ein Lagerbereich, an dem Artikel auftragsspezifisch zusammengestellt werden. |
| <b>Aussagensammlung</b>                                   |  |
| Ein Lager besteht aus 0 bis beliebig vielen Lagerbereich. |  |
| Ein Lager hat einen Name.                                 |  |
| Ein Kommissionierbereich ist ein Lagerbereich.            |  |
| Ein Reservebereich ist ein Lagerbereich.                  |  |

Abbildung 3: Spezifikation der Funktionalität von Komponenten in UNSCOM/T.

Durch die Vorgabe der Begriffsklassen und Beziehungstypen wird sichergestellt, dass bei der Spezifikation jeweils eine Ontologie mit gleichartigen Bestandteilen entsteht, die z.B. mit den Methoden des Semantic Web ausgewertet werden kann. Auf diese Weise wird es möglich, die spezifizierte Funktionalität einer Komponente einer Auswertung in semantischen Suchverfahren und automatisierten Kompatibilitätstests zugänglich zu machen. Zur Repräsentation des Begriffssystems wird in UNSCOM/T eine sog. *Normsprache* verwendet, die maschinenlesbar und wegen ihrer Nähe zur natürlichen Sprache gleichzeitig für Entwickler und Fachexperten verständlich ist. In UNSCOM/G werden hierfür UML 2 *Klassen- und Aktivitätsdiagramme* sowie spezielle Stereotypen eingesetzt [Ove06].

### 3.3 Green Pages: Spezifikation der logischen Architektur

Green Pages beschreiben die logische Architektur der von einer Komponente angebotenen bzw. nachgefragten Dienste als Programmierschnittstellen. Diese Informationen werden für den korrekten Aufruf von Diensten benötigt und unterstützen somit hauptsächlich den Kopplungsprozess. Die Definition der Schnittstellen erfolgt durch Angabe von *Signaturlisten*, mit denen sich Typen, Konstanten, Attribute, Methoden, Ausnahmen und Ereignisse vereinbaren lassen (vgl. Abb. 4). Da die Angabe von Signaturlisten für den korrekten Aufruf im Allgemeinen nicht hinreichend ist, lassen sich mit dem UNSCOM Spezifikationsrahmen zusätzliche Architekturmerkmale angeben. Hierzu gehören die beim Aufruf von Diensten geltenden Vor- und Nachbedingungen, die als *Dienstverträge* zur Kompositionszeit jedoch nur heuristisch ausgewertet werden können [ZW97], sowie *Reihenfolgebeziehungen* zwischen Diensten, die als reguläre Automaten zu spezifizieren sind.

```

interface IBestandsvwt {
    exception KontoUnguelteig ();

    void setzeMindestbestand(in long konto, in long bestand) raises (KontoUnguelteig);
    long aktuellerBestand(in long konto) raises (KontoUnguelteig);
    long physischerBestand(in long konto) raises (KontoUnguelteig);
};

eventtype MindestbestandUnterschritten (long artikel);

component Lagermanagement {
    provides IBestandsvwt __IBestandsvwt;
    publishes MindestbestandUnterschritten __MindestbestandUnterschritten;
};

context IBestandsvwt::aktuellerBestand(konto: Integer): Integer
post: result >= 0
-- Der frei verfügbare Bestand eines Artikels ist nicht kleiner als Null

context IBestandsvwt::physischerBestand(konto: Integer): Integer
post: result >= 0
-- Der physische Bestand eines Artikels ist nicht kleiner als Null

```

Abbildung 4: Spezifikation der Architekturmerkmale von Komponenten in UNSCOM/T.

Zur Darstellung von Signaturlisten wird in UNSCOM/T die *OMG Interface Definition Language* (IDL) des CORBA Standards genutzt. In UNSCOM/G erfolgt die Repräsentation durch ein UML 2 *Komponentendiagramm*. Spezifizierte Dienstverträge werden in beiden Formaten durch die *OMG Object Constraint Language* (OCL) dargestellt. Zur Angabe von Reihenfolgebeziehungen ist in UNSCOM/T ein proprietäres Format zu nutzen, das in UNSCOM/G als UML 2 *Zustandsdiagramm* dargestellt werden kann [Ove06].

### 3.4 Grey Pages: Spezifikation der physischen Qualität

Mit den Grey Pages lässt sich die Qualität einer Komponente und der von ihr angebotenen bzw. nachgefragten Dienste beschreiben. Diese Informationen sind für die Auswahl und Kopplung von Komponenten gleichermaßen von Bedeutung. Basierend auf dem Qualitätsmodell des ISO 9126 Standards [ISO01, ISO03] unterscheidet der UNSCOM Spezifikationsrahmen dabei zwischen statischen, operationalen und dynamischen *Merkmalskategorien*: zu den statischen Merkmalskategorien zählen die *Verwendbarkeit*, *Wartbarkeit* und *Portabilität* einer Komponente, die operationale Merkmalskategorie wird durch die *Funktionalität* bestimmt und zu den dynamischen Merkmalskategorien gehören die *Zuverlässigkeit* sowie die *Effizienz*. Für jede dieser Merkmalskategorien wurde aus dem ISO

Standard eine Reihe messbarer *Kennzahlen* übernommen und zu einer Bibliothek standardisierter Qualitätsattribute zusammengefasst (vgl. Abb. 5 oben).

```

type Usability = contract {
  documentationUnderstandability : increasing numeric;
  meanTimeToUse : decreasing numeric m;
};

type Efficiency = contract {
  responseTime : decreasing numeric s;
  throughput : increasing numeric calls/s;
  memoryUtilization : decreasing numeric b;
  discUtilization : decreasing numeric b;
};

LagermanagementVerwendbarkeit for Lagermanagement = profile {
  require Usability contract {
    documentationUnderstandability > 0.9;
    meanTimeToUse (mean < 60 m);
  };
};

BestandsvwtNormallast for IBestandsvwt = profile {
  from aktuellerBestand, physischerBestand require Efficiency contract {
    responseTime (mean < 2 s);
    throughput (mean > 15 calls/s; percentile 80 > 10 calls/s);
  };
};

BestandsvwtHochlast for IBestandsvwt = profile {
  from aktuellerBestand, physischerBestand require Efficiency contract {
    responseTime (mean < 1 s);
    throughput (mean > 20 calls/s; percentile 80 > 15 calls/s);
  };
};

LagermanagementNormallast for Lagermanagement = servicelevel {
  require BestandsvwtNormallast, DatenbankNormallast;
};

LagermanagementHochlast for Lagermanagement = servicelevel {
  require BestandsvwtHochlast, DatenbankHochlast;
};

```

Abbildung 5: Spezifikation der Qualitätseigenschaften von Komponenten in UNSCOM/T.

Von zentraler Bedeutung ist die Spezifikation der dienstbezogenen, dynamischen Qualitätsmerkmale, die auch als *Quality of Service* bezeichnet und in Form sog. *Service Level Agreements* angegeben werden. Die Qualität der angebotenen Dienste hängt dabei in der Regel von der Qualität der bei der Dienstauführung nachgefragten Dienste, dem Nutzungsprofil sowie der Hardware-Umgebung ab. Diese Größen lassen sich deshalb bei der Spezifikation eines Service Level Agreements explizit einbeziehen (vgl. Abb. 5 unten). Zur Repräsentation von Qualitätsmerkmalen und Service Level Agreements wird in UNSCOM/T und UNSCOM/G auf die *Quality Modeling Language* (QML) zurückgegriffen.

## 4 Schlussbetrachtung

Im Rahmen dieses Beitrags wurden Konzepte bzw. Methoden zur Spezifikation von Komponenten dargestellt und zum UNSCOM Spezifikationsrahmen zusammengefasst. Bei der Entwicklung des Spezifikationsrahmens wurde ein besonderes Augenmerk auf die Erfüllung der eingangs genannten grundlegenden Anforderungen gelegt, damit eine möglichst gute Unterstützung für die Auswahl und Kopplung von Komponenten gegeben ist. So wurde durch die systematische Herleitung von Vertragsebenen der Forderung nach *Vollständigkeit* Rechnung getragen. Zugleich verfügt der UNSCOM Spezifikationsrahmen über einen modularen Aufbau und besitzt die geforderte *Veränderbarkeit*. Mit seinen normativen, von konkreten Technologien unabhängigen Vorgaben hinsichtlich des zu beschreibenden Inhalts und der einzusetzenden Notationen erfüllt er ferner die Kriterien der *Verbindlichkeit* und *Universalität*. Da der Spezifikationsrahmen vorwiegend etablierte, industriefähige aber formale Notationen einsetzt, besitzt er die gewünschte *Praktikabilität* und

*Verständlichkeit*. Das in diesem Beitrag nicht näher dargestellte Metaschema mit seinen Konsistenzregeln sorgt schließlich für die geforderte *Abgestimmtheit*. Das Kriterium der *Angemessenheit* wird derzeit durch Anwendung des Spezifikationsrahmens überprüft.

Zur Zeit wird daran gearbeitet, industriefähige *Werkzeuge* für die Spezifikation von Komponenten nach dem UNSCOM Spezifikationsrahmen bereitzustellen. Unter Einbeziehung von Wissenschaft, Industrie und Standardisierungsgremien wird ferner eine Weiterentwicklung und *Standardisierung* des Spezifikationsrahmens innerhalb des GI-Arbeitskreises *Komponentenbasierte betriebliche Anwendungssysteme* angestrebt [Tur02]. Schließlich hat der Aufbau einer umfassenden Werkzeugunterstützung für den Kompositionsprozess begonnen, der in Kooperation mit anderen Partnern vorangetrieben werden soll. Unter dem Projektnamen CompoNex (Component Nexus) soll dabei eine *Konstruktionsmethodik* entstehen, deren Methoden und Werkzeuge wie in den anderen Ingenieursdisziplinen auf einem gemeinsamen, standardisierten Spezifikationsrahmen aufsetzen.

## Literatur

- [BJPW99] A. Beugnard, J.-M. Jezequel, N. Plouzeau und D. Watkins. Making Components Contract Aware. *IEEE Computer*, 32(7):38–45, 1999.
- [Bro00] A. W. Brown. *Large-Scale, Component-Based Development*. Prentice Hall, NJ, 2000.
- [CD94] S. Cook und J. Daniels. *Designing Object Systems. Object-Oriented Modelling with Synropy*. Prentice Hall, Englewood Cliffs, NJ, 1994.
- [CD01] J. Cheesman und J. Daniels. *UML Components. A Simple Process for Specifying Component-Based Software*. Addison-Wesley, NJ, 2001.
- [Crn02] I. Crnkovic. Component-Based Software Engineering - New Challenges in Software Development. *Software Focus*, 2(4):127–133, 2002.
- [DW99] D. F. D’Souza und A. C. Wills. *Objects, Components, and Frameworks with UML. The Catalysis Approach*. Addison-Wesley, NJ, 1999.
- [GAO95] D. Garlan, R. Allan und J. Ockerbloom. Architectural Mismatch: Why Reuse Is So Hard. *IEEE Software*, 12(6):17–26, 1995.
- [ISO01] ISO/IEC. Software Engineering - Product Quality - Quality Model. ISO Standard 9126-1, International Organization for Standardization, 2001.
- [ISO03] ISO/IEC. Software Engineering - Product Quality - External Metrics. ISO Standard 9126-2, International Organization for Standardization, 2003.
- [McI68] M. D. McIlroy. Mass Produced Software Components. In *Software Engineering: Report on a Conference by the NATO Science Committee*, Seiten 138–150, Brussels, 1968.
- [Mey92] B. Meyer. Applying “Design by Contract”. *IEEE Computer*, 25(10):40–51, 1992.
- [OHMR91] T. W. Olle, J. Hagelstein, I. G. MacDonald und C. Rolland. *Information Systems Methodologies. A Framework for Understanding*. Addison-Wesley, Wokingham, 1991.
- [OMG03] OMG. UML 2.0 Superstructure Specification. Adopted Specification ptc/03-08-02, Object Management Group, 2003.

- [OT05] S. Overhage und P. Thomas. WS-Specification: Ein Spezifikationsrahmen zur Beschreibung von Web-Services auf Basis des UDDI-Standards. In O. K. Ferstl et al., Hrsg., *Wirtschaftsinformatik 2005*, Seiten 1539–1558. Physica, 2005.
- [Ove06] S. Overhage. *Vereinheitlichte Spezifikation von Komponenten: Grundlagen, UnSCom Spezifikationsrahmen und Anwendung*. Dissertation, Universität Augsburg, 2006.
- [PD91] R. Prieto-Diaz. Implementing Faceted Classification for Software Reuse. *Communications of the ACM*, 34(5):89–97, 1991.
- [Rop99] G. Ropohl. *Allgemeine Technologie*. Hanser, München, 1999.
- [Sam97] J. Sametinger. *Software Engineering with Reusable Components*. Springer, Berlin, Heidelberg, 1997.
- [Sch98] A. W. Scheer. *ARIS: Vom Geschäftsprozess zum Anwendungssystem*. Springer, Berlin, Heidelberg, 3.. Auflage, 1998.
- [SGM02] C. Szyperski, D. Gruntz und S. Murer. *Component Software. Beyond Object-Oriented Programming*. Addison-Wesley, Harlow, 2. Auflage, 2002.
- [Tur02] K. Turowski. Standardized Specification of Business Components. Bericht, GI, 2002.
- [UDD00] UDDI Organization. UDDI Technical White Paper. Public draft, Universal Description, Discovery, and Integration Standards Organization, 2000.
- [VZJ03] P. Vitharana, F. Zahedi und H. Jain. Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis. *IEEE Transactions on Software Engineering*, 29(7):649–664, 2003.
- [Wal03] K. C. Wallnau. A Technology for Predictable Assembly from Certifiable Components. Bericht CMU/SEI-2003-TR-009, Software Engineering Institute, 2003.
- [Wey01] E. J. Weyuker. The Trouble with Testing Components. In G. T. Heineman und W. T. Councill, Hrsg., *Component-Based Software Engineering. Putting the Pieces Together*, Seiten 499–512. Addison-Wesley, NJ, 2001.
- [ZW97] A. M. Zaremski und J. M. Wing. Specification Matching of Software Components. *ACM Transactions on Software Engineering and Methodology*, 6(4):333–369, 1997.



**Sven Overhage** (Jahrgang 1975) studierte Wirtschaftsinformatik an der Technischen Universität (TU) Darmstadt. Nachdem er sein Diplom mit Auszeichnung abgeschlossen hatte, arbeitete er als Doktorand am Fachgebiet Entwicklung von Anwendungssystemen der TU Darmstadt sowie am Lehrstuhl für Wirtschaftsinformatik und Systems Engineering der Universität Augsburg. Im Jahre 2006 promovierte er an der Universität Augsburg mit der Note summa cum laude über das in diesem Beitrag beschriebene Thema. Seine wissenschaftlichen Arbeiten wurden durch ein Graduiertenstipendium der Friedrich-Ebert-Stiftung gefördert und auf Konferenzen sowie durch die Software-Industrie mehrfach ausgezeichnet.

Sven Overhage ist Mitgründer und stellvertretender Sprecher der GI-Fachgruppe Software-Architektur. Am Lehrstuhl für Wirtschaftsinformatik und Systems Engineering leitet er die Forschungsgruppe Component und Service Engineering. Daneben ist er als Lehrbeauftragter an der Hochschule Liechtenstein sowie als Gutachter für internationale Konferenzen und Journale tätig.