Zur Weiterentwicklung der Spezifikation betrieblicher Softwarekomponenten

Jörg Ackermann

Lehrstuhl für Wirtschaftsinformatik und Systems Engineering
Universität Augsburg
Universitätsstraße 16, 86135 Augsburg
joerg.ackermann.hd@t-online.de

Zusammenfassung. Die präzise und eindeutige Spezifikation einer Softwarekomponente ist eine wesentliche Voraussetzung für ihre erfolgreiche Wiederverwendung. Dazu wurde 2002 vom Arbeitskreis WI-KobAS der GI ein Spezifikationsrahmen erstellt, wie die Spezifikation betrieblicher Softwarekomponenten erfolgen soll. Mittlerweile entsprechen die damaligen Empfehlungen allerdings nicht mehr in allen Punkten dem Stand der Forschung und dem Stand der Technik. Dieser Beitrag diskutiert bestehende Defizite und unterbreitet einen umfassenden Vorschlag, wie der Spezifikationsrahmen weiterentwickelt werden sollte.

1 Einleitung

Das grundlegende Paradigma komponentenorientierter Softwareentwicklung besteht darin, Anwendungen aus wiederverwendbaren Komponenten zu bauen – auf diese Weise wird die Erstellung von Komponenten (development for reuse) von der Erstellung ganzer Systeme aus Komponenten (development with reuse) entkoppelt [Br00, Sz02]. Dieser Ansatz verspricht u. a. kürzere Entwicklungszeiten, höhere Flexibilität und geringere Entwicklungskosten [DW98:397f., Gr98:583ff.].

Um erfolgreich wiederverwendet werden zu können, muss eine Komponente alle für ihren Einsatz relevanten Informationen in Form einer Komponentenspezifikation zur Verfügung stellen. Eine präzise und verlässliche Spezifikation unterstützt die Auswahl der passenden Komponente und das Vertrauen in die korrekte Arbeitsweise der Komponente [GG06:100]. Außerdem sind Komponentenspezifikationen Voraussetzung für den Erfolg von Komponentenmärkten [HT02] sowie für eine Kompositionsmethodologie und Werkzeugunterstützung [Ov04:4]. Aus diesen Gründen ist die Spezifikation von Softwarekomponenten ein kritischer Erfolgsfaktor bei der komponentenorientierten Entwicklung betrieblicher Anwendungssysteme.

Für betriebliche Softwarekomponenten steht mit dem Memorandum „Vereinheitlichte Spezifikation von Fachkomponenten“ [Tu02] ein Spezifikationsrahmen zur Verfügung, dessen Ziel in der Schaffung eines methodischen Standards für die Spezifikation besteht.

Mittlerweile entsprechen die Empfehlungen des Memorandums allerdings nicht mehr in allen Punkten dem Stand der Forschung und dem Stand der Technik.

Deshalb wird vorgeschlagen, dass der Arbeitskreis WI-KobAS eine überarbeitete Version 2.0 des Memorandums erstellt. Dieser Beitrag zeigt dazu auf, an welchen Stellen Handlungsbedarf besteht, und unterbreitet Vorschläge, wie sich die Defizite beseitigen lassen. Dazu wird zum einen auf eigene Forschungsergebnisse zurückgegriffen und zum anderen beschrieben, wie sich Ergebnisse anderer Autoren (vor allem [Ov06]) in das Memorandum (in seiner bisherigen Struktur) integrieren lassen. Die Arbeit ist wie folgt aufgebaut: Zunächst erfolgt eine Einführung zur Spezifikation betrieblicher Softwarekomponenten (Kapitel 2) sowie die Vorstellung einer Beispielkomponente (Kapitel 3). Danach erfolgen Vorschläge zu allgemeinen Aspekten der Spezifikation (z. B. Sprachauswahl und Erweiterbarkeit), die alle Ebenen betreffen (Kapitel 4). Anschließend werden für die sieben Spezifikationsebenen des Memorandums bestehende Defizite benannt und Lösungsvorschläge unterbreitet (Kapitel 5). Die Arbeit schließt mit dem zusammenfassenden Kapitel 6. Die Bedeutung der vorliegenden Arbeit besteht darin, dass ein umfassender Vorschlag unterbreitet wird, wie der Spezifikationsrahmen [Tu02] weiterentwickelt werden kann.

2 Spezifikation betrieblicher Softwarekomponenten

Unter *Spezifikation einer Softwarekomponente* wird die vollständige, widerspruchsfreie und eindeutige Beschreibung der Außensicht einer Komponente verstanden – die Spezifikation zeigt auf, welche Dienste eine Komponente in welchem Bedingungsrahmen bereitstellt [Tu02:3].

Trotz der großen Bedeutung von Komponentenspezifikationen gibt es zurzeit keinen Spezifikationsstandard, der allgemein anerkannt ist und alle relevanten Sachverhalte abdeckt. So konzentrieren sich viele der bestehenden Ansätze auf einzelne Entwicklungsphasen wie z. B. Entwurf und Entwicklung [DW98, CD01], Komposition [YS97] oder Auswahl [HL01]. Außerdem streben viele dieser Arbeiten eine Unterstützung von Entwicklung, Anpassung und Komposition aus technischer Sicht an – Suche und Auswahl von Komponenten (insbesondere aus fachlicher Sicht) werden dagegen nur unzureichend unterstützt [GG06]. Bisher existieren nur wenige Ansätze, deren Ziel die umfassende und vollständige Spezifikation von Softwarekomponenten ist – zu nennen sind dabei vor allem [Be99, Tu02, Ov06].

Die vorliegende Arbeit bezieht sich auf den Spezifikationsrahmen „Vereinheitlichte Spezifikation von Fachkomponenten“ (Memorandum) [Tu02]. Der Inhalt des Memorandums ist eine Empfehlung des Arbeitskreises „Komponentenorientierte betriebliche Anwendungssysteme“ (WI-KobAS) der Gesellschaft für Informatik und wurde gemeinsam von Vertretern aus Forschung und Praxis erstellt. Der Spezifikationsrahmen strebt an, einen *methodischen* Standard für die Spezifikation von betrieblichen Fachkomponenten zu schaffen. Dazu wurden die zu spezifizierenden Objekte ermittelt und verschiedenen Beschreibungsebenen zugeordnet und es wurden die auf den einzelnen Ebenen einzusetzenden Notationen festgeschrieben. [Tu02] sieht

insgesamt sieben Spezifikationsebenen vor: Syntax-, Verhaltens-, Abstimmungs-, Qualitäts-, Terminologie-, Aufgaben und Vermarktungsebene. In Kapitel 5 findet sich eine kurze Einführung zu Inhalt und Notation der einzelnen Ebenen.

Das Memorandum [Tu02] hebt sich von früheren Spezifikationsansätzen vor allem dadurch ab, dass es die Spezifikation aller relevanten Aspekte in einem vereinheitlichten Ansatz anstrebt, dazu sowohl technische als auch fachliche Aspekte betrachtet und berücksichtigt, dass die unterstützten betrieblichen Aufgaben sowie die verwendeten betrieblichen Begriffe auf fachlicher Ebene zu spezifizieren sind. Nachteile des Memorandums bestehen darin, dass die Spezifikationsvorschriften auf der Qualitätsebene unpräzise bleiben, dass die zu verwendenden Satzbaupläne auf Terminologie- und Aufgabenebene nicht verbindlich vorgegeben werden und dass der Spezifikationsrahmen von einem eingeschränkten Komponentenmodell (keine Mehrfachschnittstellen, keine benötigten Schnittstellen) ausgeht.

Zu erwähnen bleibt, dass auf den Ideen von [Tu02] aufbauend eine Weiterentwicklung der Komponentenspezifikation durch Arbeiten von Overhage erfolgte, in deren Ergebnis der Spezifikationsrahmen UnSCom [Ov06] entstand. Die vorliegende Arbeit unterscheidet sich von [Ov06] dadurch, dass hier Vorschläge zur Weiterentwicklung unterbreitet werden, die Struktur und Aufbau des Memorandums (weitgehend) erhalten, während sich [Ov06] bzgl. Spezifikationsebenen und Notationstechniken zum Teil deutlich von [Tu02] unterscheidet.

3 Beispielkomponente *Lagerverwaltung*

Zur Illustration der weiteren Ausführungen wird in diesem Abschnitt eine Beispielkomponente *Lagerverwaltung* eingeführt. Die Komponente unterstützt die Abwicklung einer sehr einfachen Lagerhaltung – zur Vereinfachung wurden Funktionalität und Komplexität auf das Nötigste reduziert. Die Komponente ist komplex genug, um als Beispiel in dieser Arbeit zu dienen, aber in dieser Form kaum praxistauglich – so unterstützt die Komponente z. B. nur die manuelle Einlagerung.

Die Komponente *Lagerverwaltung* unterstützt die Lagerabwicklung in genau einem Lager. Die von der Komponente verwalteten Daten werden in Abb. 1 durch ein Datenmodell dargestellt. Das Lager besteht aus einer Anzahl von *Lagerplätzen*, auf denen die Materialien physisch gelagert werden. *Lagerbestand* repräsentiert die an einem Lagerplatz vorhandene Menge eines Lagermaterials (beispielsweise drei Paletten von Lagermaterial ABC-XYZ). *Lagermaterial* enthält die lagerspezifischen Eigenschaften eines Materials. Wird auf einem Lagerplatz ausschließlich ein bestimmtes Lagermaterial gelagert, kann man dem Lagerplatz dieses Lagermaterial zuweisen. Im Beispiel werden keine Lagereinheitentypen verwendet und es wird stattdessen angenommen, dass innerhalb des Lagers jeder Lagerplatz für ein Material geeignet ist.

Die Beispielkomponente unterstützt die Ausführung folgender zentraler Lageraktivitäten: Lagermaterial einlagern, Lagermaterial auslagern, Lagerbestand abfragen (Schnittstelle *IStockManagement*). Darüber hinaus existieren Funktionen, mit

denen sich Lagerplätze und Lagermaterialien verwalten lassen (Schnittstellen *IStorageBin* und *IWarehouseMaterial*). Außerdem ermöglicht die Komponente, bei einer (außerhalb der Komponente liegenden) Materialverwaltungskomponente die Existenz eines Lagermaterials abzufragen (benötigte Schnittstelle *IMaterial*).

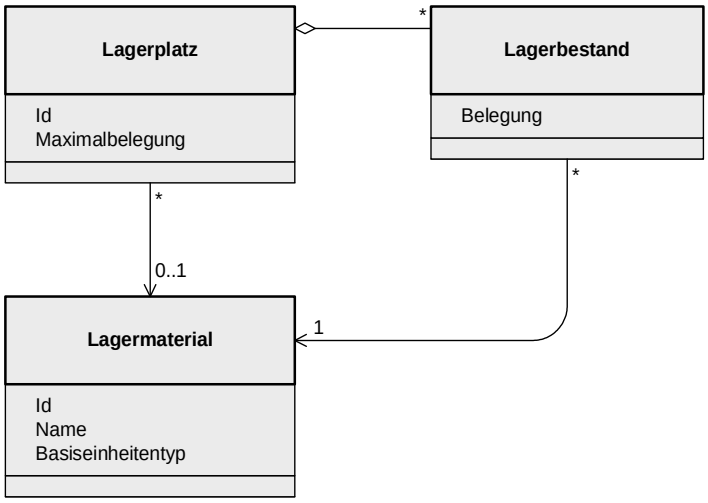

Abb. 1: Von der Komponente *Lagerverwaltung* verwaltete Daten

4 Ebenenübergreifende Aspekte

Dieses Kapitel fasst offene Punkte und Vorschläge zusammen, die sich auf den Spezifikationsrahmen [Tu02] im Ganzen beziehen und sich damit keiner der Spezifikationsebenen zuordnen lassen – für die einzelnen Ebenen selbst siehe Kapitel 5.

4.1 Aufbau und Spezifikationsebenen

Der Aufbau des Spezifikationsrahmens an sich und insbesondere die einzusetzenden Spezifikationsebenen bedürfen einer nochmaligen gründlichen Überprüfung. So wird z. B. in [Ov06:67] kritisiert, dass der Identifikation der Ebenen in [Tu02] nur ein empirisches Vorgehen zugrunde lag, dessen Ergebnis sich nicht in intersubjektiv nachvollziehbarer Weise begründen lässt. Im Gegensatz dazu werden die Spezifikationsebenen von UnSCom aus einem Klassifikationsschema abgeleitet, welches nach Modellperspektiven (statisch, operational, dynamisch) und Abstraktionsebenen (fachlich, logisch, physisch) unterscheidet [Ov06:125ff.]. Ein Nachteil der UnSCom-Lösung liegt jedoch darin, dass drei Spezifikationsebenen für die physische Komponentenbeschreibung entstehen, obwohl (aufgrund das Black-Box-Charakters) nur vergleichsweise wenige Informationen zu erfassen sind – die Beschreibung der nicht-funktionalen Eigenschaften wirkt damit unangemessen aufgebläht und könnte stattdessen auch auf einer zusammengefassten Ebene (= Qualitätsebene in [Tu02]) erfolgen.

4.2 Explizite Benennung der spezifikationsrelevanten Sachverhalte

Ein Defizit von [Tu02] ist, dass die spezifikationsrelevanten Sachverhalte nicht genau identifiziert werden. Eine explizite und vollständige Benennung aller spezifikationsrelevanten Sachverhalte ist jedoch sowohl für einen Spezifikationsersteller (welche Aspekte sind in der Spezifikation zu beschreiben und welche nicht?) als auch für den Hersteller eines Spezifikationstools (welche Aspekte muss man erfassen können?) unverzichtbar.

Daher wird vorgeschlagen, zu jeder Spezifikationsebene alle spezifikationsrelevanten Sachverhalte explizit aufzuführen. Als *spezifikationsrelevanter Sachverhalt* wird dabei alles bezeichnet, was die Außensicht der Komponente betrifft. Dies können Dinge (z. B. Attribute), deren Eigenschaften (z. B. Wertebereich) und zugehörige Ausprägungen (z. B. Festwerte *A* und *B*) sein. Um die Breite des Spektrums auszudrücken, wurde der allgemeine Begriff *Sachverhalt* gewählt. Beispiele solcher Sachverhalte sind Name und Dienste einer Schnittstelle oder Name und Wertebereich eines Qualitätsmerkmals.

Objekte	**Spezifikationsrelevante Sachverhalte**
Komponente	Name, Zugeordnete Schnittstellen (angeboten/benötigt)
Schnittstelle	Name, Dienste (Name, Parameter)
Datentypen	Name, Attribute (Name, Datentyp, Kardinalität) bzw. Aufzählungswerte
Typ im Spezifikations-datenmodell	Name, Attribute (Name, Datentyp, Kardinalität)
Beziehung im Spezifikationsdatenmodell	Beziehungsart, Beteiligte Typen, Kardinalitäten, Rollen

Abb. 2: Spezifikationsrelevante Sachverhalte auf der Schnittstellenebene

Die spezifikationsrelevanten Sachverhalte lassen sich entweder tabellarisch erfassen [Ac07a:97f.] oder in Form von Metaschemata abbilden [Ov06:133]. Abb. 2 zeigt exemplarisch, wie eine tabellarische Darstellung der spezifikationsrelevanten Sachverhalte für die Schnittstellenebene aussieht. Um die Übersichtlichkeit der Einträge zu erhöhen, werden die Sachverhalte nach Bezugsobjekten geordnet dargestellt [Ac04:141]. Motivation dafür ist, dass spezifikationsrelevante Sachverhalte nicht isoliert existieren, sondern meist zu einem übergeordneten Bezugsobjekt gehören.

(*Qualitätsmerkmal* ist ein Beispiel für ein Bezugsobjekt, welches u. a. die spezifikationsrelevanten Sachverhalte *Name* und *Wertebereich* umfasst.)

In [Ac07a:98-123] findet sich die vollständige Auflistung der spezifikationsrelevanten Sachverhalte aller Spezifikationsebenen, auf deren Darstellung in dieser Arbeit aus Platzgründen verzichtet werden muss.

4.3 Ablage ebenenübergreifender Informationen

Das Memorandum schlägt vor, die Beschreibung von Komponenten auf verschiedenen Spezifikationsebenen vorzunehmen [Tu02:3]. Durch eine solche Aufteilung der Spezifikationsinhalte auf verschiedene thematische Beschreibungsebenen wird die Komplexität reduziert – Spezifikationen werden damit für den Produzenten einfacher zu erstellen und für den Konsumenten leichter zu verstehen.

Allerdings hat sich herausgestellt, dass eine Spezifikation auch Informationen enthalten muss, die Sachverhalte verschiedener Ebenen betreffen. Beispiele dafür sind die Zuordnung von Sachverhalten aus fachlicher und technischer Sicht [Tu02:5] oder die ebenenübergreifende Zusammenfassung von Parametrisierungsauswirkungen [Ac07a:227ff.]. Solche ebenenübergreifenden Informationen ließen sich in der bisherigen Struktur des Memorandums nur unzureichend abbilden – so wurden z. B. die Zuordnungstabellen für die fachliche und technische Sicht einfach der Schnittstellenebene zugeordnet [Tu02:5]. Die Zuweisung solcher übergreifenden Informationen zu einer einzelnen Spezifikationsebene ist jedoch dann unangemessen, wenn die Ebenenauswahl willkürlich ist und die Informationen von Charakter und Notationstechnik nicht zu den übrigen Informationen der Ebene passen.

Aus diesem Grund wird vorgeschlagen, dass der Spezifikationsrahmen [Tu02] um einen weiteren Bereich für ebenenübergreifende Informationen erweitert wird. Dieser Bereich kann bei Bedarf für unterschiedliche Aspekte nochmal unterteilt werden. Wichtig ist dabei, dass bei jedem Kandidaten für den ebenenübergreifenden Bereich untersucht wird, ob die Informationen tatsächlich ebenenübergreifend sind (wie bei einer Zuordnung von Sachverhalten verschiedener Ebenen) oder ob ein falscher Zuschnitt der Ebenen die Ursache dafür ist, dass sich die betroffenen Informationen nicht einer Spezifikationsebene zuordnen lassen.

4.4 Mehrsprachige Komponentenspezifikationen

Eine Komponentenspezifikation ist eine formalisierte Dokumentation einer Softwarekomponente. Analog zu anderen Dokumentationen und Produkthandbüchern ist davon auszugehen, dass Komponentenanwender eine Spezifikation in ihrer Landessprache bevorzugen. Damit stellt sich die Frage nach der Mehrsprachigkeit von Komponentenspezifikationen, wenn eine Komponente in verschiedenen Ländern vertrieben wird. Damit muss auch ein Spezifikationsrahmen und darauf aufbauende Spezifikationstools die Möglichkeit vorsehen, eine Spezifikation in mehreren Sprachen

zu erstellen – diese Problematik wurde von den gängigen Spezifikationsansätzen [DW98, CD01, Tu02, Ov06] bisher nicht diskutiert.

Bei einer Analyse der Sprachabhängigkeit der Spezifikationsartefakte [Ac07a:95f.] ergibt sich, dass manche Artefakte einsprachig sein müssen (Namen von Schnittstellen und Datentypen), andere mehrsprachig sein könnten (Begriffe und Aufgaben) sowie die Sprachkonstrukte von OCL und QML einsprachig auf Englisch sind.

Im Spezifikationsrahmen [Tu02] sind die Spezifikationsbeispiele in der deutschen Version komplett auf Deutsch und wurden für die englische Version komplett ins Englische übersetzt – damit wird als Lösung des Sprachproblems suggeriert, eine Spezifikation einfach komplett in andere Sprachen zu übersetzen. Diese auf den ersten Blick naheliegende Lösung ist jedoch nicht möglich, da Schnittstellen- und Datentypnamen (beim derzeitigen Stand der Technik) eindeutig und damit einsprachig sein müssen!

Stattdessen wird nachfolgend vorgeschlagen, wie durch eine geeignete Sprachauswahl das Ziel einer verständlichen und konsistenten Spezifikation erreicht werden kann:

- Auf den technischen Ebenen (Schnittstellen, Verhalten, Abstimmung, Qualität) besteht das Problem, dass Teile der Spezifikation einsprachig sein müssen (Schnittstellen), andere mehrsprachig sein können (Datenmodell) und andere fest in Englisch sind (z. B. OCL). Da eine Mischung von Sprachen die Verständlichkeit der Spezifikation beeinträchtigt, wird vorgeschlagen, die Spezifikation auf diesen Ebenen einsprachig in Englisch zu erstellen. Dieser Vorschlag ist angemessen, da bei Technikexperten (Zielgruppe dieser Ebenen) englische Sprachkenntnisse typischerweise vorhanden sind und technische Spezifikationen sowieso häufig in Englisch erstellt werden.

- Für die fachlichen Ebenen (Terminologie, Aufgaben) und die Vermarktungsebene wird vorgeschlagen, die Spezifikation in allen benötigten Sprachen zur Verfügung zu stellen. Dies begründet sich dadurch, dass zum einen ausreichende englische Sprachkenntnisse bei Fachexperten (Zielgruppe dieser Ebenen) nicht vorausgesetzt werden können und dass zum anderen die vollständige Übersetzung aller Spezifikationsinhalte dieser Ebenen technisch möglich ist. Diese Mehrsprachigkeit muss vom Spezifikationsrahmen (sprachabhängige Satzbaupläne) und von darauf aufbauenden Tools (sprachabhängige Spezifikationsinhalte) explizit vorgesehen werden.

Diesen Richtlinien folgend sind in dieser Arbeit alle Spezifikationsbeispiele auf den technischen Ebenen in Englisch und auf den fachlichen Ebenen in Deutsch.

4.5 Erweiterbarkeit des Spezifikationsrahmens

Aus dem Ziel des Spezifikationsrahmens [Tu02], einen Standard zur Spezifikation von betrieblichen Fachkomponenten zu schaffen, ergibt sich, dass dieser Spezifikationsvorschriften für alle spezifikationsrelevanten Sachverhalte enthalten muss.

Daher sind auch spezielle Aspekte zu berücksichtigen (wie z. B. die Spezifikation von parametrisierbaren Komponenten [Ac07a] oder von Komponenten mit graphischen Benutzeroberflächen [Ov06:275]), die zwar für manche, aber nicht für alle Komponenten relevant sind. Werden alle diese Vorschriften in [Tu02] direkt aufgenommen, ist zu befürchten, dass der Spezifikationsrahmen bald unübersichtlich wird und damit an Akzeptanz verliert.

Aus diesem Grund wird vorgeschlagen, bei der Weiterentwicklung von [Tu02] zwischen einem Kernstandard und ergänzenden Zusatzstandards zu unterscheiden. Sinnvoll wäre in diesem Zusammenhang, wenn schon der Kernstandard Möglichkeiten zur Erweiterung vorsieht, so dass zusätzliche Aspekte einfach ergänzt werden können, ohne jeweils den Kernstandard erweitern zu müssen. So sollte z. B. der Kernstandard erlauben, die verwendeten UML-Elemente zu stereotypisieren oder zusätzliche Satzbaupläne aufzunehmen.

5 Ebenenspezifische Weiterentwicklungsvorschläge

In diesem Kapitel werden Vorschläge zur Weiterentwicklung des Memorandums unterbreitet, die sich konkret auf die einzelnen Spezifikationsebenen beziehen. Dabei orientiert sich die Darstellung an den derzeit bestehenden Ebenen von [Tu02] – ggf. werden sich diese im Rahmen der Weiterentwicklung ändern (vgl. dazu die Diskussion in Abschnitt 4.1).

5.1 Schnittstellenebene

Die Schnittstellenebene dient dazu, grundlegende Vereinbarungen zur technischen Kommunikation zu treffen [Tu02:5]. Dazu gehören die Namen der Dienste, die eine Komponente anbietet oder benötigt, die Definition der verwendeten Datentypen, die Signaturen der Dienste sowie die Deklaration von Fehlermeldungen. Die daraus resultierenden Vereinbarungen garantieren, dass Dienstnehmer und Dienstgeber auf technischer Ebene miteinander kommunizieren können – semantische Aspekte bleiben dagegen weitgehend unberücksichtigt.

Die im Memorandum [Tu02] vorgesehenen Spezifikationsvorschriften für die Schnittstellenebene weisen allerdings einige eklatante Einschränkungen und Probleme auf: es wird von einem eingeschränkten Komponentenmodell mit nur einer angebotenen Schnittstelle ausgegangen, der Name der Komponente kann nicht explizit definiert werden (sondern ergibt sich aus dem Namen der Schnittstelle), benötigte Dienste werden durch einen Workaround mithilfe einer Schnittstelle *Extern* deklariert und durch die vorgeschlagene Verwendung der OMG IDL als Notationstechnik kommt es zu einem Methodenbruch bei der Verhaltensebene (in UML OCL formulierte Constraints erfordern ein UML-Modell – der Bezug auf OMG IDL-Ausdrücke ist syntaktisch falsch [Ac03:26f.]).

Die bei der Erstellung von [Tu02] gültige Version 1.4 der UML enthielt zwar das Modellierungskonstrukt *Komponente*, war jedoch ausschließlich für die Modellierung von physischen Komponenten vorgesehen. Bei einer Spezifikation sind jedoch logische Komponenten einzusetzen [Ac03:24], die von UML 1.4 nicht explizit unterstützt wurden. Diese Einschränkung entfällt mit der aktuellen Version UML 2.0 [OMG05b] – damit ist nun möglich, das Konstrukt *Komponente* zur Modellierung logischer Komponenten zu verwenden.

Zur Weiterentwicklung des Spezifikationsrahmens [Tu02] wird vorgeschlagen, auf der Schnittstellenebene die UML 2.0 als primäre Notationstechnik einzusetzen:

- Zur Spezifikation einer Komponente wird ein Komponentendiagramm eingesetzt, aus welchem der Name der Komponente sowie die Schnittstellen der Komponente hervorgehen – vgl. dazu Abb. 3. Die UML erlaubt die Angabe mehrere Schnittstellen pro Komponente und (anhand unterschiedlicher Notationen) eine Unterscheidung zwischen angebotenen (z. B. *IStockManagement*) und benötigten Schnittstellen (*IMaterial*).

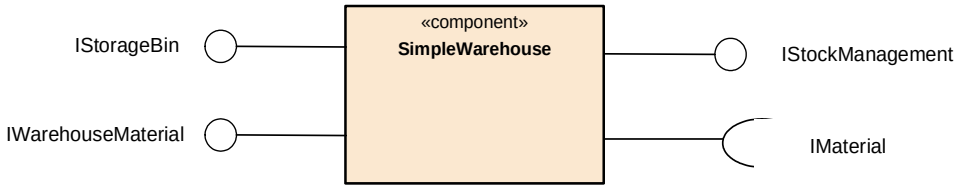

Abb. 3: Beispielkomponente *Lagerverwaltung* mit Schnittstellen

- Die genaue Spezifikation der Schnittstellendienste erfolgt mit Hilfe des UML-Konstrukts *Schnittstelle* (engl. *interface*) und wird für die Schnittstelle *IStockManagement* beispielhaft in Abb. 4 dargestellt. Dabei werden zu jedem Dienst die Parameter (mit Parameterrichtung (in, out, inout), Name, Datentyp und Kardinalität), ein optionaler Returnparameter (mit Datentyp) und eventuell eingesetzte Ausnahmen (engl. *exceptions*) spezifiziert.

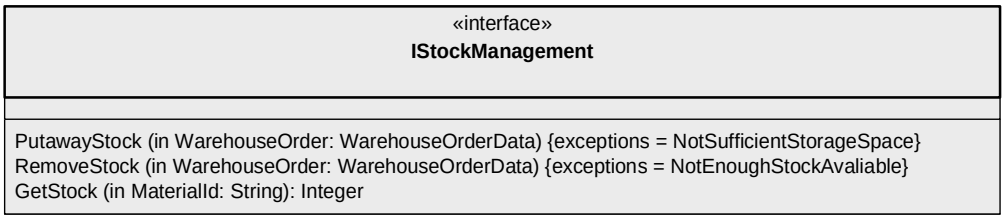

Abb. 4: Detaillierte Spezifikation der Schnittstelle *IStockManagement*

- Werden neben UML-Standarddatentypen (wie *String* oder *Integer*) auch eigene Datentypen und Ausnahmetypen verwendet, so müssen diese ebenfalls mit Hilfe der UML spezifiziert werden. Dazu können die UML-Modellierungskonstrukte *dataType* und *exception* eingesetzt werden.

Durch den Einsatz der UML in der angegebenen Form ergeben sich folgende Vorteile für die Spezifikation auf der Schnittstellenebene: 1. Schnittstellen-, Verhaltens- und Abstimmungsebene verwenden mit der UML eine einheitliche Notationstechnik und es kann auf eine Notationstechnik (OMG IDL) verzichtet werden. (Bei Bedarf lässt sich eine IDL-Spezifikation aus der UML-Beschreibung ableiten.) 2. Der bisher vorkommende Methodenbruch zwischen Schnittstellen- und Verhaltensebene sowie die anderen oben aufgeführten Einschränkungen werden vermieden. 3. Der Spezifikationsrahmen bleibt konform zur aktuellen UML-Version und die Spezifikation profitiert von den erweiterten Ausdrucksmöglichkeiten von UML 2.0. Damit wird also vorgeschlagen, die UML konsequent auf den Ebenen einzusetzen, wo sie die Spezifikation verbessern kann und wo sie international als Modellierungsstandard anerkannt ist. Betriebliche Fachkomponenten lassen sich jedoch nicht vollständig mit der UML spezifizieren, da auch spezifikationsrelevante Objekte und Spezifikationsebenen (z. B. Terminologie und betriebliche Aufgaben) zu betrachten sind, die von der UML nicht abgedeckt werden.

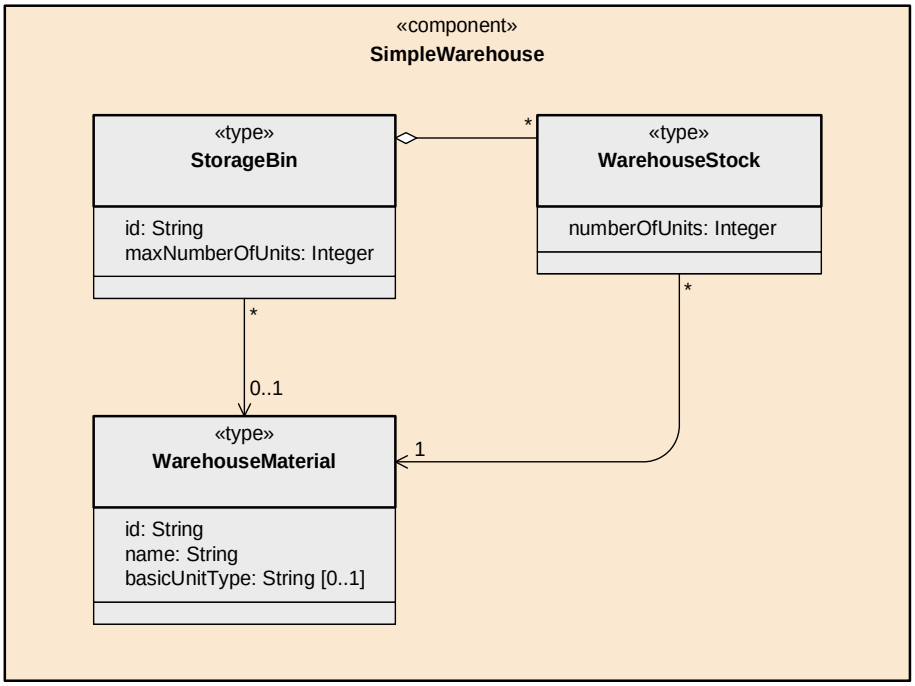

Abb. 5: Spezifikationsdatenmodell der Beispielkomponente *Lagerverwaltung*

Betriebliche Fachkomponenten verwalten häufig betriebliche Daten – so erlaubt die Komponente *Lagerverwaltung* beispielsweise Lagerplätze anzulegen, zu speichern und zu einem späteren Zeitpunkt wieder anzuzeigen oder zu verändern. Um solches Verhalten vollständig und syntaktisch korrekt spezifizieren zu können, muss die Spezifikation mit einem geeigneten Datenmodell versehen werden. Zu diesem Zweck wurde das Konzept eines *Spezifikationsdatenmodells* eingeführt. (Das Modell modelliert nur Daten und keine Operationen und ist daher kein Objektmodell.) Ein solches Modell

repräsentiert die von einer Komponente verwalteten Daten auf konzeptioneller Ebene – es enthält dazu die vertragsrelevanten Daten und abstrahiert von den internen Implementierungsdetails. Die Darstellung dieses Modells erfolgt mithilfe des UML-Konstrukts *component* und wird in Abb. 5 beispielhaft für die Komponente *Lagerverwaltung* gezeigt. In [Ac07b] findet sich eine ausführliche Darstellung des Konzepts inklusive der Begründung seiner Notwendigkeit und einer Diskussion, welche Daten in einem solchen Modell abzubilden sind und welche nicht.

Es wird vorgeschlagen, ein solches (optionales) Spezifikationsdatenmodell der Schnittstellenebene zuzuordnen. Für diese Zuordnung spricht, dass das Modell die vertragsrelevanten Daten der Komponente deklariert und damit gut zu den anderen Deklarationen auf der Schnittstellenebene passt, dass die eingesetzten Notationen (UML-Modelle) übereinstimmen und dass das Modell sowohl auf der Verhaltens- als auch auf der Abstimmungsebene benötigt wird und sich daher keiner der beiden Ebenen eindeutig zuordnen lässt.

5.2 Verhaltensebene

Aufgabe der Verhaltensebene ist, das Verhalten einer Komponente näher zu beschreiben – dazu wird spezifiziert, wie sich die Komponente im Allgemeinen und insbesondere bei Grenzfällen verhält [Tu02:7]. So kann beispielsweise angegeben werden, dass die Summe aller Bestände auf einem Lagerplatz die für den Lagerplatz festgeschriebene Maximalbelegung nicht überschreiten darf oder dass die Einlagerung von Lagermaterialien den Bestand auf dem verwendeten Lagerplatz entsprechend erhöht. Wichtig ist dabei, dass das Verhalten der Komponente vollständig zu beschreiben ist.

Als primäre Notationstechnik wird auf der Verhaltensebene die *UML Object Constraint Language* (OCL) [OMG05a] eingesetzt. Verhaltensbeschreibungen mithilfe der OCL erfolgen durch die Angabe von allgemeinen Invarianten sowie von dienstspezifischen Vor- und Nachbedingungen.

In dieser Arbeit wird weitgehend den vom Memorandum vorgegebenen Spezifikationsvorschriften gefolgt [Tu02:7-9]. Es ergeben sich jedoch folgende Detailverbesserungen:

- Das neu eingeführte Spezifikationsdatenmodell ermöglicht, Invarianten und mit persistierten Daten zusammenhängende Vor- und Nachbedingungen syntaktisch korrekt zu formulieren.

- Es vereinfacht sich die Verhaltensspezifikation der benötigten Dienste, da diese direkt der Komponente zugeordnet sind und der Workaround über eine gedachte Komponente *Extern* entfällt.

- Alle in OCL formal spezifizierten Bedingungen sind zusätzlich (im Sinne einer sekundären Notationstechnik) natürlichsprachlich anzugeben – es wird vorgeschlagen, dafür die von der OCL vorgesehenen Kommentare zu verwenden.

```
context StorageBin
  inv: self.maxNumberOfUnits > 0
    -- attribute maxNumberOfUnits must be greater than 0

  inv: self.WarehouseStock.numberOfUnits->sum() <= self.maxNumberOfUnits
    -- sum of all stock can not exceed attribute maxNumberOfUnits
```

Abb. 6: Invarianten für den Typ *StorageBin*

Abb. 6 zeigt exemplarisch zwei Invarianten. Diese legen fest, dass die Maximalbelegung eines Lagerplatzes immer größer als null sein muss und dass die Summe aller auf einem Lagerplatz gelagerten Bestände nicht größer sein kann als die für den Lagerplatz vorgesehene Maximalbelegung.

5.3 Abstimmungsebene

Vereinbarungen auf der Abstimmungsebene regeln die Reihenfolge, in der Dienste aufgerufen werden können, sowie die Synchronisationserfordernisse zwischen Diensten [Tu02:10]. Beispielsweise wird hier festgelegt, dass ein Lagermaterial zunächst definiert werden muss, bevor eine Einlagerung für dieses Lagermaterial durchführbar ist.

Mögliche Notationsalternativen auf dieser Ebene sind die um temporale Operatoren erweiterte OCL [CT01] (vorgeschlagen in [Tu02:10]) sowie Zustandsautomatenmodelle (z. B. vorgeschlagen in [Ov06:189ff.]). Für die Verwendung von Automaten spricht, dass diese (z. B. im Rahmen von Kompatibilitätstests oder Qualitätsvorhersagen) statisch auswertbar sind. Der Nachteil von Automaten ist jedoch, dass diese alle erlaubten Aufrufreihenfolgen explizit spezifizieren – eine vollständige Erfassung ist jedoch nicht immer praktikabel. In einem solchen Fall haben temporale Operatoren einen Vorteil, da sie lediglich Einschränkungen bei der Reihenfolge von Dienstaufrufen festlegen. Ein Nachteil der temporalen OCL in der Form von [CT01] liegt darin, dass die formalen Grundlagen der Erweiterung nicht expliziert werden [ZG03:351].

Daher wird für die Abstimmungsebene vorgeschlagen, noch einmal die genaue Zielstellung der Spezifikation festzulegen und darauf aufbauend die Wahl der geeigneten Notationstechnik zu treffen.

5.4 Qualitätsebene

Ziel der Qualitätsebene ist die Spezifikation der nicht-funktionalen Eigenschaften einer Komponente – Beispiele dafür sind die Verfügbarkeit, das Performanzverhalten und die Wartbarkeit eines durch die Komponente angebotenen Dienstes [Tu02:12]. Dazu muss die Qualitätsebene zu beschreibende Qualitätskriterien, verwendbare Messgrößen und Methoden zu deren Quantifizierung sowie ggf. durch die Komponenten einzuhaltende Service Level (Qualitätsgrenzen zur Laufzeit) enthalten. Darüber hinaus ist festzulegen,

in welcher Form diese Informationen dem Benutzer einer Komponente zur Verfügung gestellt werden.

Der Spezifikationsrahmen [Tu02] erkennt zwar die Notwendigkeit der Spezifikation nicht-funktionaler Eigenschaften, enthält jedoch nur ein eher allgemeines Vorgehensmodell, aber kein Qualitätsmodell und keine konkreten Spezifikationsvorschriften.

Deutlich konkretere Vorgaben zur Spezifikation nicht-funktionaler Eigenschaften sind im Spezifikationsrahmen UnSCom enthalten. Besonders hervorzuheben sind dabei folgende Ergebnisse [Ov06:212-236]:

- Es wird ein abstraktes Qualitätsmodell entwickelt, welches alle spezifikationsrelevanten Sachverhalte strukturiert in Form eines Metamodells darstellt. Dabei muss eine Komponentenspezifikation sowohl die Definition der verwendeten Qualitätskategorien und -merkmale (auf Typebene) als auch die für die Komponente konkreten Kennzahlen (Ausprägungen) der Merkmale enthalten.

- Es werden sechs Qualitätskategorien (*Verwendbarkeit*, *Wartbarkeit*, *Portabilität*, *Funktionalität*, *Zuverlässigkeit* und *Effizienz*), insgesamt 49 Qualitätsmerkmale sowie geeignete Messverfahren vorgegeben, welche bei der Spezifikation einer Softwarekomponente einsetzbar sind.

- Es wird vorgeschlagen, die Spezifikation mithilfe der *Quality of Service Modeling Language (QML)* [FK98] durchzuführen. Erwähnenswert ist dabei, dass die QML sowohl für die Definition der Kategorien und Merkmale als auch für die Angabe der Kennzahlen einsetzbar ist.

Abb. 7 zeigt beispielhaft eine QML-Spezifikation für die Qualitätskategorie *Effizienz* (*efficiency*). Im oberen Teil werden z. B. für das Merkmal *Antwortzeit* (*responseTime*) die Eigenschaften rationaler Wertebereich, Maßeinheit *Sekunden* und die Präferenz für kleinere Werte festgelegt. Danach wird (für die Beispielkomponente) z. B. festgelegt, dass die Antwortzeit des Dienstes *GetStock* im Mittel kleiner als eine Sekunde ist.

```
type Efficiency = contract {
  responseTime : decreasing numeric s;
  throughput : increasing numeric calls/s; ... };

StockManagement for IStockManagement = profile {
  from GetStock require Efficiency = contract {
    responseTime {mean < 1 s;};
    throughput {mean > 25 calls/s; percentile 80 > 18 calls/s;}; }; };
```

Abb. 7: Beispielhafte Spezifikation für die Qualitätskategorie *Effizienz*

Es wird vorgeschlagen, die Ergebnisse aus [Ov06] auf den Spezifikationsrahmen [Tu02] zu übertragen. Zu beachten ist dabei, dass die in [Ov06] auf mehrere Spezifikationsebenen verteilten Sachverhalte gemeinsam in die Qualitätsebene zu integrieren sind.

5.5 Terminologieebene

Die Terminologieebene dient als zentrales Begriffsverzeichnis einer betrieblichen Fachkomponente [Tu02:16]. Dazu werden in einem Lexikon alle (wesentlichen) Fachbegriffe mit ihrer Definition, erklärenden Beispielen und ihrer Beziehung zu anderen Begriffen gespeichert. Die Terminologieebene wendet sich hauptsächlich an Fachexperten (zur Komponentenauswahl), aber auch an Technikexperten (zur Komponentenanpassung und -komposition), und hat zum Ziel, alle relevanten Fachbegriffe eindeutig und allgemein verständlich zu definieren.

Dazu wird als Notationstechnik der Einsatz von Normsprachen vorgeschlagen, bei denen es sich um Ontologiedefinitionssprachen handelt, welche sowohl für einen menschlichen Benutzer als auch maschinell lesbar sind [OS96]. Dazu besteht eine Normsprache aus einem Lexikon mit geklärten Fachbegriffen und verwendet zur Bildung von Aussagen eine rationale Grammatik (rekonstruierte Grammatik der Umgangssprache), die eine Reihe von Satzbauplänen als Schablonen vorgibt. Die Verwendung von Satzbauplänen hat zwei Vorteile gegenüber der Verwendung natürlicher Sprache: Spezifikationen werden präziser und das entstehende Lexikon bildet eine einfache Ontologie, welche das automatische Auffinden von Zusammenhängen vereinfacht. Der Vorteil von Satzbauplänen gegenüber anderen Ontologienotationen (wie z. B. dem Resource Description Framework (RDF)) liegt darin, dass die entstehenden Sätze auch für Fachexperten verständlich sind, die keine Erfahrung mit formalen Modellen oder Ontologienotationen haben.

Das Defizit des Spezifikationsrahmens [Tu02] besteht darin, dass die Spezifikationsvorschriften insgesamt vage bleiben und keine Satzbaupläne angegeben werden. Die einzusetzenden Satzbaupläne sollten allerdings vom Spezifikationsrahmen vorgegeben werden, da zum einen nur durch deren Standardisierung das Ziel erreicht werden kann, Ungenauigkeiten und Vagheiten zu vermeiden [Za05:41ff.], und zum anderen nicht jedem Komponentenhersteller zugemutet werden kann, eigene Satzbaupläne zu entwerfen.

Zur Präzisierung der Vorschriften auf der Terminologieebene wird vorgeschlagen, dass die Spezifikation eines Fachbegriffs aus drei Teilen besteht:

- A) der eigentliche Lexikoneintrag, der zur Klärung des Begriffs dient;
- B) eine Aussagensammlung, die den Begriff in Beziehung zu anderen Begriffen (des Lexikons) setzt und damit eine umfassende Spezifikation des Begriffsgeflechts innerhalb der Komponente (bzw. der unterstützten Domäne) erlaubt;
- C) Restriktionen, welche Abhängigkeiten zwischen Begriffen spezifizieren.

Zu A) Die Definition eines Begriffs erfolgt in Form einer Tabelle [Tu02:18; Ov06:164]. Diese enthält neben der genauen Bezeichnung des Begriffs (Begriffswort) eine natürlichsprachliche Kurzdefinition, (optional) eine ausführlichere, ebenso natürlichsprachliche Langdefinition sowie erklärende Beispiele und ggf. Gegenbeispiele. Ein Beispiel dazu findet sich in den ersten vier Tabellenzeilen von Abb. 8.

Zu B) Der Spezifikationsrahmen UnSCom enthält eine Liste vordefinierter Satzbaupläne zur Spezifikation von Begriffen [Ov06:171]. Es wird vorgeschlagen, diese Satzbaupläne für die Terminologieebene von [Tu02] zu übernehmen. Mit Hilfe dieser Satzbaupläne lässt sich eine Aussagensammlung erstellen, welche die Zusammenhänge zwischen Begriffen standardisiert beschreibt und (unter der gleichnamigen Rubrik) Teil der Definition eines Begriffs ist. So wird z. B. in Abb. 8 festgelegt, dass ein Lagerplatz aus 0 bis beliebig vielen Lagerbeständen besteht.

Begriffswort: LAGERPLATZ
Kurzdefinition: Ein *Lagerplatz* ist die kleinste adressierbare Raumeinheit innerhalb eines LAGERS.
Langdefinition: Ein *Lagerplatz* ist die kleinste adressierbare Raumeinheit innerhalb eines LAGERS. An einem LAGERPLATZ werden Bestände von LAGERMATERIAL gelagert.
Beispiel: Lagerplatz 015-07-02 (Reihe 015, Regal 07, Ebene 02)
Aussagensammlung: Ein LAGERPLATZ hat eine ID und eine MAXIMALBELEGUNG. Eine ID ist eine Zeichenkette. Eine MAXIMALBELEGUNG ist eine Ganzzahl. Ein LAGERPLATZ besteht aus 0 bis beliebig vielen LAGERBESTAND. Ein LAGERPLATZ steht in Beziehung zu 0 bis 1 LAGERMATERIAL.
Restriktionen: Die Summe aller Bestände (LAGERBESTAND) an einem LAGERPLATZ darf den Wert von MAXIMALBELEGUNG nicht überschreiten. Ist einem LAGERPLATZ ein LAGERMATERIAL zugeordnet, kann der LAGERPLATZ nur Bestände (LAGERBESTAND) dieses Materials enthalten.

Abb.8: Spezifikation des Begriffs *Lagerplatz*

UnSCom beinhaltet Satzbaupläne für die Beschreibung der folgenden Beziehungsarten: Inklusion, Merkmalsvereinbarung, Aggregation, Konnexion und Relation. Beispielsweise lassen sich mit Hilfe der *Merkmalsvereinbarung* (auch *Partizipation*) die Merkmale eines Begriffs beschreiben – dafür dient der Satzbauplan „(Ein | Eine) *A* hat

(einen | eine | ein) *B* (und (einen | eine | ein) *C*)+“. Ein Anwendungsbeispiel dafür findet sich in der ersten Aussage (Rubrik *Aussagensammlung*) in Abb. 8, die festgelegt, dass ein Lagerplatz durch die Merkmale ID und Maximalbelegung näher beschrieben wird. Durch entsprechende Erweiterungen der Satzbaupläne lassen sich auch Rollen und Kardinalitäten berücksichtigen – so handelt es sich bei der letzten Aussage in Abb. 8 um eine Relation mit der Kardinalität [0..1]. Für die vollständige Liste aller verfügbaren Satzbaupläne vgl. [Ov06:171].

Zu C) Die letzte Tabellenzeile von Abb. 8 zeigt, dass (über die Aussagensammlung hinausgehende) Abhängigkeiten zwischen Begriffen bestehen können. Dabei kann es sich um Einschränkungen bei der Komponentenverwendung handeln, die einem (die Komponente evaluierenden) Fachexperten zur Verfügung zu stellen sind. Aus diesem Grund sind solche Restriktionen ebenfalls in die Definition eines Begriffs aufzunehmen und werden unter der Rubrik *Restriktionen* in der tabellarischen Darstellung berücksichtigt [Ac07a:114]. Der Einsatz von vorgegebenen Satzbauplänen ist an dieser Stelle schwierig, da die auftretenden Restriktionen beliebig komplex sein können – stattdessen wird vorgeschlagen, die Restriktionen natürlichsprachlich zu formulieren. Eine formalere Spezifikation dieser Restriktionen wäre in Zukunft wünschenswert. Dies könnte z. B. durch eine automatische Konvertierung formaler Bedingungen (in UML OCL) in natürliche oder normierte Sprache (vgl. beispielweise [Hä02] für erste Ansätze) oder durch die Verwendung einer Fachnormsprache für prädikatenlogische Aussagen (wie in [Ov06:173] angedacht) erfolgen.

5.6 Aufgabenebene

Ziel der *Aufgabenebene* ist die Beschreibung der von einer Komponente unterstützten betrieblichen Aufgaben sowie ggf. deren Zerlegung in mehrere Teilaufgaben auf fachlichem (konzeptionellem) Niveau [Tu02:19]. Die Aufgabenebene wendet sich hauptsächlich an Fachexperten und soll eine Beurteilung erlauben, ob die Komponente aus fachlicher Sicht für einen konkreten Anwendungsfall einsetzbar ist. Analog zur Terminologieebene wird festgelegt, dass die Spezifikation mit Hilfe von Normsprachen erfolgt.

Wie auch auf der Terminologieebene bleibt [Tu02] bei der Aufgabenebene vage und verzichtet auf die vollständige Angabe aller einsetzbaren Satzbaupläne. Daher sind bei einer Weiterentwicklung von [Tu02] die Spezifikationsvorschriften für Aufgaben zu präzisieren. Dazu wird vorgeschlagen, die Ergebnisse aus [Ov06:158-160,172-174] auf die Aufgabenebene zu übertragen. Zentrales Element ist dabei der Vorschlag, Begriffe und Aufgaben auf ähnliche Weise zu spezifizieren – durch eine solche Vereinheitlichung wird die Spezifikation sowohl für menschliche Benutzer (durch Systematisierung) als auch für Tools vereinfacht. Analog zu Begriffen besteht damit auch die Aufgabendefinition aus drei Teilen:

- A) Die Definition der betrieblichen Aufgabe erfolgt durch Angabe ihres Namens, einer Kurzdefinition, einer optionalen Langdefinition sowie von Beispielen und ggf. Gegenbeispielen. Der Name einer betrieblichen Aufgabe besteht aus einem Substantiv (dem primären Objekt der Handlung), einem optionalen Adjektiv (zur

Präzisierung der Handlung) und einem Verb (dem Handlungsbezeichner) – Beispiele dafür sind die betrieblichen Aufgaben *Lagerbestand abfragen* und *Lagermaterial manuell einlagern*.

- B) Die Aussagensammlung spezifiziert, in welcher Beziehung die Aufgabe zu anderen betrieblichen Aufgaben sowie zu den auf der Terminologieebene beschriebenen Begriffen steht. Im Rahmen der Aussagensammlung kann die Zerlegung einer Aufgabe in Teilaufgaben (Komposition), die Angabe von Aufgabenvarianten (Inklusion) sowie die Beschreibung von Aufgabenmerkmalen (Merkmalsvereinbarung), mit denen die Beziehung der Aufgabe zu Begriffen angegeben wird, beschrieben werden. Für eine vollständige Übersicht über alle verfügbaren Satzbaupläne vgl. [Ov06:176].

- C) Für die Ausführung betrieblicher Aufgaben können Restriktionen bestehen – so kann beispielsweise nur ein Lagerungsauftrag der Art *Einlagerung* zum Ausführen der Aufgabe *Lagermaterial einlagern* verwendet werden. Einschränkungen solcher Art sind für Fachexperten bei der Komponentenbewertung relevant und müssen deshalb (als Restriktion unter der gleichnamigen Rubrik) in die Aufgabenspezifikation aufgenommen werden. Wie auf der Terminologieebene erfolgt die Spezifikation von Restriktionen in natürlicher Sprache.

Ein Beispiel für die Anwendung der Spezifikationsvorschriften findet sich in Abb. 9.

Aufgabe: LAGERMATERIAL EINLAGERN
Kurzdefinition: Das *Einlagern von Lagermaterial* umfasst die Aufnahme einer bestimmten Anzahl von LAGERMATERIAL in das LAGER und dessen physische Lagerung an einem (oder mehreren) LAGERPLATZ.
Beispiel: Drei Paletten von Lagermaterial ABC-XYZ ins Lager aufnehmen und auf Lagerplatz 015-07-02 lagern
Aussagensammlung: Ein LAGERVERWALTER tut LAGERMATERIAL EINLAGERN mit einem LAGERUNGSAUFTRAG.
Restriktionen: Die AUFTRAGSART von LAGERUNGSAUFTRAG ist Einlagerung. Der LAGERUNGSAUFTRAG enthält (EINHEITENANZAHL viele) LAGERPLÄTZE.

Abb. 9: Spezifikation der betrieblichen Aufgabe *Lagermaterial einlagern*

5.7 Vermarktungsebene

Zweck der Vermarktungsebene ist die Spezifikation solcher Merkmale, die benötigt werden, um die Komponente *betriebswirtschaftlich-organisatorisch* handhabbar zu machen. Die auf dieser Ebene spezifizierten Merkmale sind insbesondere für Verkäufer und Einkäufer sowie für Assemblierer und Qualitätssicherer von Interesse [Tu02:21]. Als primäre Notationstechnik wird die Tabellenform vorgeschlagen – außerdem werden die im Rahmen der Vermarktungsebene zu spezifizierenden Merkmale explizit vorgegeben. Als Erweiterung wird vorgeschlagen, bei allen vorgegebenen Merkmalen zu überprüfen, ob die Angabe mehrerer Merkmalswerte möglich sein sollte. So ist es denkbar, dass eine Komponente mehrere Komponententechnologien (typischerweise mehrere Versionen derselben Technologie) unterstützt.

6 Zusammenfassung

Diese Arbeit beschäftigte sich mit der Weiterentwicklung der Spezifikation betrieblicher Softwarekomponenten. Ausgangspunkt der Untersuchungen war der Spezifikationsrahmen für betriebliche Fachkomponenten (Memorandum) [Tu02]. Es wurde vorgeschlagen, diesen Spezifikationsrahmen weiterzuentwickeln – dazu wurden bestehende Defizite identifiziert und es wurde ein umfassender Vorschlag unterbreitet, wie das Memorandum weiterentwickelt werden sollte. Die Ergebnisse dieses Beitrags können damit als Ausgangspunkt für Diskussionen zu einem „Memorandum 2.0“ dienen. Neben einer Überarbeitung des Spezifikationsrahmens an sich sollte in Zukunft ein Spezifikationstool erstellt werden, welches auf dem Memorandum basiert, und es sollten Möglichkeiten zur Vereinfachung der Komponentenspezifikation (z. B. durch den Einsatz von Spezifikationsmustern [AT06]) identifiziert und umgesetzt werden.

Literaturverzeichnis

[Ac03] *Ackermann, J.*: Spezifikation von Fachkomponenten mit der UML 2.0. In: *Turowski, K. (Hrsg.)*: 4. Workshop Modellierung und Spezifikation von Fachkomponenten. Bamberg 2003, S. 23-30.

[Ac04] *Ackermann, J.*: Zur Beschreibung datenbasierter Parametrisierung von Softwarekomponenten. In: *Turowski, K. (Hrsg.)*: Architekturen, Komponenten, Anwendungen – Proceedings zur 1. Verbundtagung Architekturen, Komponenten, Anwendungen (AKA 2004). LNI Band P-57. Augsburg 2004, S. 131-149.

[Ac07a] *Ackermann, J.*: Spezifikation der parametrisierungsbezogenen Eigenschaften betrieblicher Fachkomponenten. Dissertation. Augsburg 2007.

[Ac07b] *Ackermann, J.*: Using a Specification Data Model for Specification of Black-Box Software Components. In: Enterprise Modelling and Information Systems Architectures 2 (2007) 1, S. 3-13.

[AT06] *Ackermann, J.; Turowski, K.*: A Library of OCL Specification Patterns for Behavioral Specification of Software Components. In: *Dubois, E.; Pohl, K. (Hrsg.)*: CAiSE 2006. Springer-Verlag LNCS 4001. Luxemburg 2006, S. 255-269.

[Be99] *Beugnard, A.; Jézéquel, J.-M.; Plouzeau, N.; Watkins, D.*: Making Components Contract Aware. In: IEEE Computer 32 (1999) 7, S. 38-44.

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

[CD01] *Cheesman, J.; Daniels, J.:* UML Components. Addison-Wesley. Boston 2001.

[CT01] *Conrad, S.; Turowski, K.*: Temporal OCL: Meeting Specification Demands for Business Components. In: *Siau, K.; Halpin, T. (Hrsg.)*: Unified Modeling Language: Systems Analysis, Design and Development Issues. Idea Group. Hershey 2001, S. 151-165.

[DW98] *D'Souza, D.F.; Wills, A.C.*: Objects, Components, and Frameworks with UML: The Catalysis Approach. Addison-Wesley. Reading 1998.

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

[GG06] *Geisterfer, C.J.M., Ghosh, S.*: Software Component Specification: A Study in Perspective of Component Selection and Reuse. In: Proceedings of the 5th International Conference on COTS-Based Software Systems (ICCBSS'06). IEEE Computer Society Press. Orlando 2006, S. 100-108.

[Gr98] *Griffel, F.*: Componentware. Konzepte und Techniken eines Softwareparadigmas. dpunkt Verlag. Heidelberg 1998.

[Hä02] *Hähnle, R.; Johannisson, K.; Ranta, A.*: An Authoring Tool for Informal and Formal Requirements Specifications. In: *Kutsche, R.-D.; Weber, H. (Hrsg.)*: Fundamental Approaches to Software Engineering: 5th International Conference (FASE 2002). Springer-Verlag LNCS 2306. Grenoble 2002, S. 233-248.

[HL01] *Hemer, D.; Lindsay, P.*: Specification-based retrieval strategies for module reuse. In: *Grant, D.; Sterling, L. (Hrsg.)*: Proceedings 2001 Australian Software Engineering Conference. IEEE Computer Society. Canberra 2001, S. 235-243.

[HT02] *Hahn, H.; Turowski, K.*: General Existence of Component Markets. In: *Trappl, R. (Hrsg.)*: Sixteenth European Meeting on Cybernetics and Systems Research (EMCSR). Vol. 1. Vienna 2002, S. 105-110.

[OMG05a] *OMG (Hrsg.):* Object Constraint Language. Version 2.0, formal/06-05-01. URL: http://www.omg.org/technology/documents, Abruf am 2008-07-07.

[OMG05b] *OMG (Hrsg.):* Unified Modeling Language: Superstructure. Version 2.0, formal/05-07-04. URL: http://www.omg.org/technology/documents, Abruf am 2008-07-07.

[OS96] *Ortner, E., Schienmann, B.*: Normative Language Approach: A Framework for Understanding. In: *Thalheim, B. (Hrsg.)*: Conceptual Modeling. ER '96, 15th International Conference on Conceptual Modeling. Springer-Verlag LNCS 1157. Cottbus 1996, S. 261-276.

[Ov04] *Overhage, S.*: Zur Spezifikation von Komponenten der Informationssystementwicklung mit Softwareverträgen. In: *Rebstock, M. (Hrsg.)*: Tagungsband Modellierung betrieblicher Informationssysteme – MobIS 2004. LNI Band P-40. Essen 2004, S. 3-20.

[Ov06] *Overhage, S.*: Vereinheitlichte Spezifikation von Komponenten – Grundlagen, UnSCom Spezifikationsrahmen und Anwendung. Dissertation. Augsburg 2006.

[Sz02] *Szyperski, C.; Gruntz, D.; Murer, S.*: Component Software: Beyond Object-Oriented Programming. 2. Aufl. Addison-Wesley. Harlow 2002.

[Tu02] *Turowski, K. (Hrsg.):* Vereinheitlichte Spezifikation von Fachkomponenten – Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme. Universität Augsburg. Augsburg 2002. URL: http://www.fachkomponenten.de. Abruf am 2008-07-07.

[YS97] *Yellin, D.; Strom, R.*: Protocol Specifications and Component Adaptors. In: ACM Transactions on Programming Languages and Systems 19 (1997), S. 292–333.

[Za05] *Zaha, J.M.*: Automatisierte Kompatibilitätstests für fachliche Software-Komponenten. Dissertation. Augsburg 2005.

[ZG03] *Ziemann, P.; Gogolla, M.*: OCL Extended with Temporal Logic. In: *Broy, M.; Zamulin A.V. (Hrsg.)*: Perspectives of Systems Informatics – 5th International Andrei Ershov Memorial Conference. Springer-Verlag LNCS 2890. Novosibirsk 2003, S. 351-357.