Prozessorientierte Beschreibung von Fachkomponenten

Holger Jaekel, Thorsten Teschke

Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS), Escherweg 2, 26121 Oldenburg, Deutschland, Tel.: +49 (441) 97 22 - 1 25, Fax: - 1 02, E-Mail: {jaekel|teschke}@offis.de, URL: http://www.offis.de

Zusammenfassung. Die systematische Wiederverwendung von Fachkomponenten erfordert die Standardisierung ihrer Spezifikation, um die auf Komponentenmärkten angebotenen Fachkomponenten zu finden und im Hinblick auf ihre Eignung für eine betriebliche Aufgabe bewerten zu können. In diesem Beitrag werden die bisherigen Vorschläge zur Standardisierung von Fachkomponenten diskutiert und eine prozessorientierte Notation für die integrierte Beschreibung von Struktur und Verhalten von Fachkomponenten und komponentenbasierten Systemen vorgestellt.

Schlüsselworte: Komponente, Fachkomponente, formale Spezifikation, Statechart, π-Kalkül, Unified Modeling Language (UML), betriebliches Anwendungssystem

1 Einleitung

Die komponentenbasierte Softwareentwicklung verfolgt den Ansatz, qualitativ hochwertige, individuelle Lösungen durch Komposition und Konfiguration von bereits am Markt verfügbaren Commercial off-the-shelf Komponenten und eigenen Entwicklungen zu erstellen. Da Entwicklung und Verwendung von Komponenten nicht in einer Hand liegen, besteht die Notwendigkeit, Komponenten hinsichtlich ihrer Struktur und ihres Verhaltens zu beschreiben. In (Turowski 1999; Turowski 2001a) werden sechs Abstraktionsebenen unterschieden, auf denen Komponenten spezifiziert werden können: die Syntax-Ebene, die Verhaltens-Ebene, die Intra-Komponenten-Abstimmungs-Ebene, die Inter-Komponenten-Abstimmungs-Ebene, die Qualitäts-Ebene und die Fachkonzept-Ebene.

In diesem Beitrag sollen Vorschläge zur Vereinheitlichung der Spezifikation von Fachkomponenten gemacht werden. Dazu werden in Abschnitt 2 die bisherigen Ansätze diskutiert. In Abschnitt 3 wird dann eine prozessorientierte Notation zur integrierten Spezifikation von Fachkomponenten und daraus zusammengesetzten Systemen vorgestellt. Anschließend werden in Abschnitt 4 Vorschläge für Spezifikationen von Fachkomponenten auf der Syntax-Ebene und den Abstimmungs-Ebenen gemacht. In Abschnitt 5 erfolgt eine kurze Zusammenfassung.

2 Diskussion des Vorschlages zur Vereinheitlichung der Spezifikation von Fachkomponenten

Spezifikationen auf der Syntax-Ebene sollen die Schnittstellen von Fachkomponenten zu beschreiben, um die Kommunikation zwischen Komponenten auf einer technischen Ebene zu ermöglichen. In (Turowski 2001b) wird für die Spezifikation von Fachkomponenten auf der

Syntax-Ebene die Verwendung der von der Object Management Group (OMG) definierten OMG IDL (OMG 2001a) vorgeschlagen, wobei Komponenten durch das Interface-Konstrukt dieser IDL repräsentiert werden. Die Verwendung der OMG IDL für diese Zwecke ist jedoch mit Problemen behaftet:

- *Komponenten und Objekte:* Es ist durchaus denkbar, dass Komponenten neben einer direkten (prozeduralen) Schnittstelle auch Klassen enthalten, die eine indirekte (objektorientierte) Schnittstelle definieren (Szyperski 1998). Solche Komponenten sind jedoch mit der OMG IDL aufgrund der Abbildung von Komponenten auf Interfaces nicht spezifizierbar.
- *Kontrakte:* Für eine vollständige Beschreibung der Außensicht ist es notwendig, auch die von der Komponente benötigten Dienste anderer Komponenten zu spezifizieren (vgl. auch den Diskussionsbeitrag von Ackermann in (Turowski 2001b, S. 49f)). Um hier nicht auf konkrete Komponenten zu verweisen, müssten an dieser Stelle abstrakte Beschreibungen von Komponenten (Kontrakte) angegeben werden. Die OMG IDL bietet jedoch weder die Möglichkeit, benötigte Dienste anzugeben noch Kontrakte zu spezifizieren.
- *Frameworks:* Für die Zusammenarbeit von verschiedenen Komponenten sind die Dienste von Anwendungs- und System-Frameworks notwendig (Nierstrasz/Lumpe 1997). Diese Abhängigkeiten können jedoch bislang nicht ausgedrückt werden, da im Verlauf der bisherigen Diskussion keine Notation vorgeschlagen worden ist, wie Frameworks spezifiziert werden sollen.

Auf der Verhaltens-Ebene soll das Verhalten von Diensten der Fachkomponente im Allgemeinen und in Grenzfällen beschrieben werden. Adressaten dieser Verhaltensbeschreibungen sind Fachanwender, die diese Beschreibungen für das Verständnis und die Bewertung der Eignung einer Fachkomponente für eine bestimmte betriebliche Aufgabe nutzen. Die Anwendung von formalen Methoden basierend auf den Verhaltensbeschreibungen wie z.B. das Aufdecken von Inkompatibilitäten verschiedener Komponenten auf der Verhaltens-Ebene erscheinen aufgrund der hohen Komplexität eher unwahrscheinlich. Bei der Wahl einer Notation sollte deshalb die Verständlichkeit der Spezifikation für den Fachanwender berücksichtigt werden. Die in (Turowski 2001b) als Notation für Spezifikationen auf der Verhaltens-Ebene vorgeschlagene Object Constraint Language (OCL) ist für Fachanwender eher schwer verständlich. Auch die vorgeschlagene Umsetzung von OCL-Ausdrücken in eine normsprachliche Form führt nicht zu einer für Fachanwender angemessenen Notation (vgl. auch den Diskussionsbeitrag von Ackermann in (Turowski 2001b, S. 53f)).

Für die Spezifikationen auf den Abstimmungs-Ebenen wird in (Turowski 2001b) die Verwendung einer um temporale Operatoren erweiterten OCL (Conrad/Turowski 2000) vorgeschlagen. Die Erweiterungen erlauben es, in Vor- und Nachbedingungen von Diensten einer Komponente Angaben über die Reihenfolge der Verwendung der Dienste zu machen. Hierbei sehen wir jedoch ein Problem in der deklarativen Sichtweise dieses Ansatzes: Spezifikationen auf den Abstimmungs-Ebenen (insbesondere der Inter-Komponenten-Abstimmungs-Ebene) weisen einen engen Bezug zu den betrieblichen Abläufen in einem Unternehmen auf. Diese Abläufe werden häufig prozedural, z.B. in Form von Geschäftsprozessen, modelliert. Da diese Geschäftsprozessmodelle möglicherweise als Ausgangspunkt für die Auswahl und Bewertung von Fachkomponenten dienen sollen, sollte sichergestellt sein, dass ein Abgleich der Spezifikationen in ein fachlich orientiertes (prozedurales) Modell möglich ist.

3 Prozessorientierte Komponentenbeschreibungen

Die von uns vorgeschlagene Komponentenbeschreibungssprache *CDL* (*Component Description Language*, siehe (Teschke/Ritter 2000)) ermöglicht eine integrierte Beschreibung von Struktur und Verhalten von Komponenten und komponentenbasierten Systemen.

3.1 Das CDL-Komponentenmodell

Eine Komponente kann nach (Szyperski 1998, S. 41f) sowohl direkte (prozedurale) als auch indirekte (objektorientierte) Schnittstellen besitzen. In dem CDL-Komponentenmodell können deshalb *Komponenten* eine direkte Schnittstelle (*Operationen*) besitzen und gleichzeitig *Klassen* enthalten, deren Instanzen dann als Geschäftsobjekte die indirekten Schnittstellen der Komponente bereitstellen.

Im CDL-Komponentenmodell werden *atomare Komponenten* und *komplexe Komponenten* unterschieden. Atomare Komponenten sind nicht zusammengesetzt, während komplexe Komponenten aus atomaren oder weiteren komplexen Komponenten zusammengesetzt sind. Atomare Komponenten können wiederum in *geschlossene Komponenten* und *Komponentenframeworks* eingeteilt werden. Während geschlossene Komponenten unabhängig von anderen Komponenten einsetzbar sind, verfügen Komponentenframeworks über Erweiterungspunkte (*Slots*), an denen Verbindungen zu anderen Komponenten hergestellt werden können, deren Dienste vom Komponentenframework benötigt werden. Komponentenframeworks sind somit der Ausgangspunkt für die Erstellung von komplexen Komponenten.

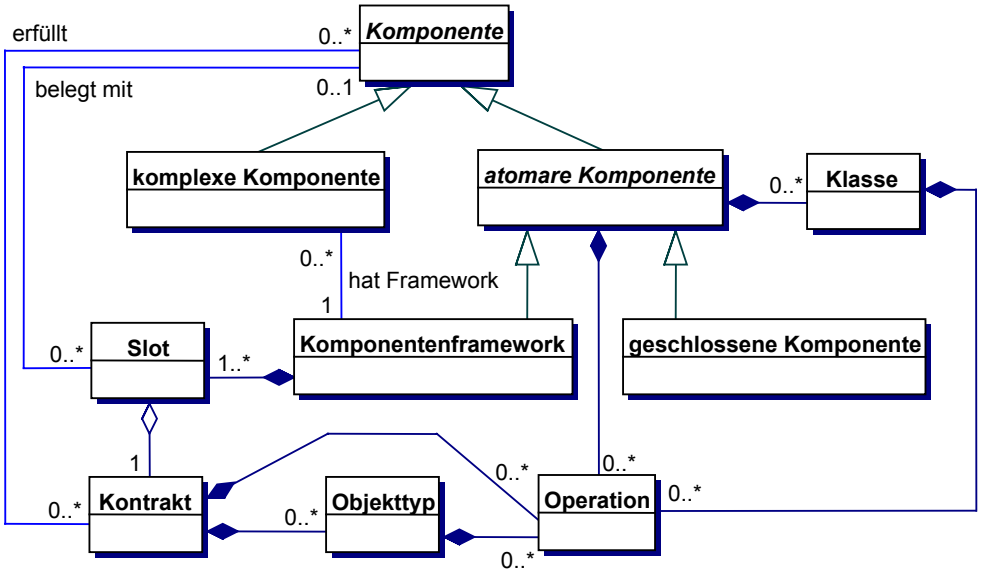

Abbildung 1: Das CDL-Komponentenmodell

Für die Typisierung der Slots werden im CDL-Komponentenmodell *Kontrakte* verwendet. Kontrakte beschreiben dabei die Anforderungen an Komponenten, die mit diesem Slot verbunden werden können. Analog zu den Beschreibungen von Komponenten werden für Kontrakte ebenfalls direkte und indirekte Schnittstellen beschrieben, den Klassen in den Komponenten entsprechen dabei *Objekttypen* in den Kontrakten. Ein Slot eines Komponentenframeworks kann mit einer Komponente belegt werden, wenn die Komponente den Kontrakt, mit dem der Slot typisiert ist, erfüllt, d.h. wenn sie mindestens die im Kontrakt angegebenen

Dienste bereitstellt und mindestens das vom Kontrakt spezifizierte Verhalten aufweist. Die Konzepte des CDL-Komponentenmodells werden zusammenfassend in Abbildung 1 dargestellt.

3.2 CDL-Verhaltensbeschreibungen

In CDL kann für jedes Element, das seinen Zustand im Laufe der Zeit ändert, eine Verhaltensspezifikation angegeben werden. Dies gilt für die Komponenten selber (die ihren Zustand durch Aufrufe der direkten Schnittstelle ändern können) und für in Komponenten enthaltene Klassen. Analog dazu kann für Kontrakte und darin enthaltene Objekttypen ein Verhalten spezifiziert werden.

Eine Spezifikation von Verhalten kann grundsätzlich deklarativ oder operational erfolgen. Den deklarativen Ansatz verfolgen Formalismen wie beispielsweise die OCL, die Vor- und Nachbedingungen für Operationen spezifizieren. Deklarative Modelle eignen sich jedoch weniger als Ausgangsbasis für anwendungsorientierte Modelle, die in den meisten Fällen einen operationalen Charaker besitzen. Eher dafür geeignet scheinen Ansätze auf Basis von Prozessalgebren wie beispielsweise CCS (Milner 1989), CSP (Hoare 1985) oder das π-Kalkül (Milner/Parrow/Walker 1992) mit einer operationalen Sichtweise von Prozessen und Interaktionen sowie Automatenmodelle wie beispielsweise Petri-Netze und Statecharts. Aus diesem Grund wurde für die Verhaltensbeschreibungen in CDL der polyadische π-Kalkül (Milner 1993) in Verbindung mit Statecharts verwendet.

Die Beschreibungen von Operationen verwenden neben den aus dem π-Kalkül bekannten Methodenaufrufen auch Konstrukte für Fallunterscheidungen und Schleifen. Diese sollten jedoch nicht dafür eingesetzt werden, die entsprechende Operation „auszuprogrammieren“, sondern dienen in erster Linie der Unterscheidung von betriebswirtschaftlich relevanten Fällen (z.B. *Kontostand* < 0). Weiterhin kann unterschieden werden, ob in der Operation verwendete Objekte neu erzeugt werden (*new*) oder ob bereits existierende Objekte referenziert werden (*existing*).

Die Abbildung 2 zeigt ein Beispiel für die Verwendung von Verhaltensspezifikationen in CDL. Dargestellt ist die CDL-Beschreibung einer Komponente *Auftragsabwicklung*, die eine Klasse *Auftrag* enthält (vgl. auch Beispiel in (Turowski 2001b, S. 56)). Die direkte Schnittstelle der Komponente enthält eine Methode *auftragAnnehmen()* zum Erzeugen eines neuen Auftrags. Die Komponente ist initial im Zustand *created()*, in dem die (in diesem Fall leere) Initialisierung der Komponente erfolgt. Anschließend wechselt die Komponente in den Zustand *initialized()*. Die Verhaltensspezifikation der Komponente *Auftragsabwicklung* erlaubt im Zustand *initialized()* den Aufruf der Methode *auftragAnnehmen()* und bleibt in diesem Zustand, so dass weitere Aufträge erzeugt werden können.

Die Verhaltensbeschreibung der Klasse *Auftrag* erlaubt nach der Initialisierung der Klasse den wiederholten Aufruf der Methode *rechnungDrucken()*. Der Aufruf der *stornieren()*-Methode führt zu einem Wechsel in den Zustand *storniert()*, in dem keine weiteren Aufrufe möglich sind. Der Operator + beschreibt dabei eine nichtdeterministische Auswahlmöglichkeit, d.h. die Klasse ist bereit, einen der beiden Dienste auszuführen. Diese Auswahl wird jedoch vom Benutzer der Klasse getroffen. Insgesamt ergibt sich damit das gewünschte Verhalten, dass nach der Erzeugung eines Auftrags Rechnungen gedruckt werden können, solange keine Stornierung stattgefunden hat.

```
component Auftragsabwicklung {

  class Auftrag {

    rechnungDrucken() {...};
    stornieren() {...};

    created()     = initialized();
    initialized() = rechnungDrucken?().initialized()
                    + stornieren?().storniert();
    storniert()   = ();
  };

  auftragAnnehmen(out at: Auftrag) {
    new(Auftrag at);
    return(at);
  };

  created()     = initialized();
  initialized() = auftragAnnehmen?(at).initialized();
};
```

Abbildung 2: Spezifikation einer Komponente *Auftragsabwicklung*

4 Notationsvorschläge

Das von uns vorgeschlagene CDL-Komponentenmodell erweitert die Modellierungsfähigkeiten der UML (OMG 2001b) im Bereich der Komponentenmodellierung. Diese Erweiterungen könnten in eine vereinheitlichte Spezifikation von Fachkomponenten einfließen.

4.1 Notationsvorschlag für die Syntaxebene

Für die Spezifikation der von der Fachkomponente angebotenen Dienste und Klassen sowie der von ihr benötigten Dienste werden Beschreibungen nach dem CDL-Komponentenmodell auf Basis der UML vorgeschlagen. Dafür müssen die Modellierungselemente des CDL-Komponentenmodells auf die in der UML vorhandenen Modellierungselemente abgebildet werden. Eine Übersicht über die Zusammenhänge zwischen dem CDL-Komponentenmodell und der UML ist in Abbildung 3 dargestellt. Die meisten Elemente aus dem CDL-Komponentenmodell haben eine direkte Entsprechung in der UML, für einige waren jedoch Erweiterungen notwendig. Alle Erweiterungen der UML wurden mittels der Mechanismen *TaggedValue* und *Stereotype* durchgeführt.

Bei der Abbildung der Modellierungselemente auf die UML wird nicht zwischen geschlossenen Komponenten und Frameworks unterschieden, beide Konzepte werden in der UML durch eine *Component* repräsentiert. Dabei wird die Vereinbarung getroffen, dass eine Komponente eine geschlossene Komponente repräsentiert, wenn sie keine Slots enthält, andernfalls ist sie ein Framework.

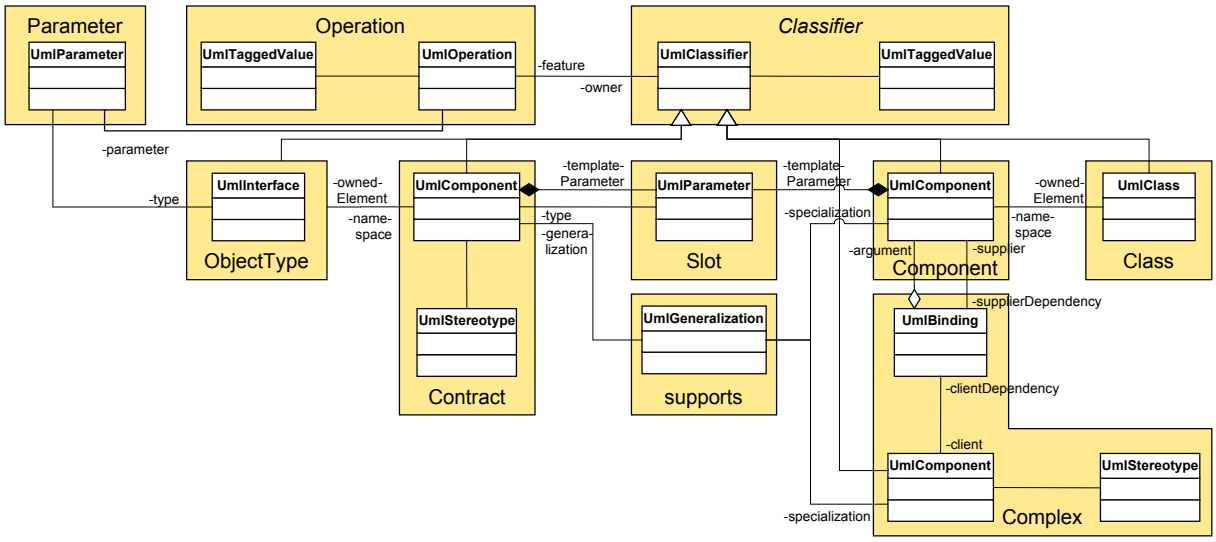

Abbildung 3: Abbildung der CDL-Modellierungselemente auf die UML

Für komplexe Komponenten und Kontrakte existiert in der UML kein Modellelement, so dass auch diese Modellierungselemente auf die *Component* mit dem Stereotyp „*Complex*“ bzw. „*Contract*“ abgebildet wurden. Die von einer Komponente unterstützten Kontrakte können über ein *Generalization*-Beziehungsobjekt mit der Komponente verbunden werden. Komponenten werden damit als Spezialisierung der Kontrakte modelliert, die sie unterstützen.

Verhaltensbeschreibungen können im CDL-Komponentenmodell für Komponenten, Kontrakte, Klassen, Objekttypen und Operationen angegeben werden. In der UML können diese Beschreibungen in ihrer textuellen Repräsentation als *TaggedValue* an *Classifier* bzw. Operationen angehängt werden.

4.2 Notationsvorschlag für die Verhaltens-Ebene

Die Beschreibung von Fachkomponenten auf der Verhaltens-Ebene sollte fachlich ausgerichtet sein, damit Fachanwender entscheiden können, ob das Verhalten der Komponente den Anforderungen entspricht. Hierbei sind insbesondere die betriebswirtschaftlichen Dienste relevant, die von der Fachkomponente angeboten werden. Die Spezifikation dieser Dienste könnte über die Namen der Komponenten, Klassen und Operationen in Verbindung mit einer betriebswirtschaftlichen Terminologie erfolgen.

Eine solche Terminologie könnte ein betriebswirtschaftliches Vokabular sowie Definitionen und semantische Beziehungen zwischen Begriffen des Vokabulars (z.B. Generalisierungen oder Aggregationen) spezifizieren. Als Grundlage einer solchen betriebswirtschaftlichen Terminologie könnten die Ergebnisse der KEBBA-Projektes (Kaufmann et al. 1999) dienen.

4.3 Notationsvorschlag für die Abstimmungs-Ebenen

Für die Spezifikationen auf den Abstimmungs-Ebenen wird eine prozessorientierte Notation auf Basis der UML vorgeschlagen. Eine prozessorientierte Beschreibung des Verhaltens von Fachkomponenten ist insbesondere dann sinnvoll, wenn man das Zusammenspiel der Komponenten im Rahmen der Geschäftsprozesse eines Unternehmens betrachtet.

Vereinbarungen auf der Intra-Komponenten-Abstimmungs-Ebene spezifizieren die Reihen-

folge, in der Dienste einer Fachkomponente verwendet werden können. In den CDL-Verhaltensbeschreibungen werden diese Informationen in den Verhaltensbeschreibungen der Komponenten und Klassen bzw. Kontrakte und Objekttypen angegeben. Ein solches Modellierungselement ist bereit, einen Dienst auszuführen, wenn sich der mit dem Element assoziierte Automat in einem Zustand befindet, dessen Prozessterm die Kommunikation des entsprechenden Dienstes zulässt. Da die CDL-Verhaltensbeschreibungen u.a. auf Statecharts basieren, lassen sich die Beschreibungen leicht auf die in der UML vorhandenen Modellierungselemente abbilden. Eine Verhaltensbeschreibung entspricht dabei einer *StateMachine*, ein Zustand aus einer Verhaltensbeschreibung entspricht einem *SimpleState*. Zustandsübergänge werden als *Transition* abgebildet, wobei der zugehörige Methodenaufruf als eine *CallAction* mit der Transition verbunden wird. Die Verwendung von Zuständen mit Parametern ist in den UML-Statecharts nicht vorgesehen sind. Solche Parameter können jedoch mittels des UML-Erweiterungsmechanismus *TaggedValue* angegeben werden.

Auf der Inter-Komponenten-Abstimmungs-Ebene werden die Reihenfolgebeziehungen der Verwendung von Diensten verschiedener Komponenten definiert. In CDL werden diese Beziehungen in den Verhaltensbeschreibungen der Operationen festgelegt: Für eine Operation wird dabei beschrieben, welche Dienste (auch anderer Komponenten, die über Slots referenziert werden) bei der Abarbeitung der Operation aufgerufen werden. Die Verhaltensbeschreibung einer Operationen kann wiederum durch eine *StateMachine* in der UML abgebildet werden. Die sequentielle Ausführung von Methodenaufrufen in den Operationen werden durch verkettete *PseudoState*s abgebildet.

5 Zusammenfassung

Für die weitere Verbreitung der komponentenbasierten Softwareentwicklung, besonders im betrieblichen Umfeld, ist eine standardisierte Spezifikation von Fachkomponenten unerlässlich. In diesem Beitrag wurden die in (Turowski 2001b) diskutierten Ansätze für eine vereinheitlichte Spezifikation auf der Syntax-Ebene, der Verhaltens-Ebene und den Abstimmungs-Ebenen diskutiert. Dabei wurden insbesondere die fehlenden objektorientierten Schnittstellen und die fehlende Kompositionalität der Beschreibungsmodelle auf der Syntax-Ebene, die unzureichende fachliche Orientierung auf der Verhaltens-Ebene und deklarative Sichtweise der Spezifikationen auf den Abstimmungsebenen thematisiert.

Die von uns für die Spezifikation von Fachkomponenten auf diesen Ebenen vorgeschlagene Notation für eine integrierte Beschreibung von Struktur und Verhalten von Komponenten und komponentenbasierten Systemen verfolgt einen prozessorientierten Ansatz. Das CDL-Komponentenmodell lässt sich auf die UML abbilden, so dass beispielsweise CDL-Komponentenmodelle in UML-Repositories abgelegt werden können. Neben diesen Beiträgen zur Syntax-Ebene liefert die CDL Beiträge zu den Abstimmungs-Ebenen.

Literatur

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

Hoare, C. A. R.: Communicating Sequential Processes. Prentice Hall, New York 1985.

Kaufmann, T.; Schmitzer, B.; Ließmann, H.; Lohmann, M.; Mertens, P.: ICF-System - Ein Werkzeug zur Anforderungsanalyse. In: Proceedings of KnowTechForum '99 (1999).

Milner, R.: Communication and Concurrency. Prentice Hall, New York 1989.

Milner, R.: The Polyadic π-Calculus: A Tutorial. In: *F. L. Bauer; W. Brauer; H. Schwichtenberg (Hrsg.)*: Logic and Algebra of Specification. Springer-Verlag, Berlin 1993.

Milner, R.; Parrow, J.; Walker, D.: A Calculus of Mobile Processes, part I and II. In: Information and Computation 100 (1992) 1, S. 1-77.

Nierstrasz, O.; Lumpe, M.: Komponenten, Komponentenframeworks und Gluing. In: HMD - Theorie und Praxis der Wirtschaftsinformatik 34 (1997) 197, S. 8-23.

OMG (Hrsg.): The Common Object Request Broker: Architecture and Specification: Revision 2.4.2. OMG, Framingham, M.A., Februar 2001a.

OMG (Hrsg.): Unified Modeling Language Specification (Version 1.4 draft). OMG, Framingham, M.A., Februar 2001b.

Szyperski, C.: Component Software: Beyond Object-Oriented Programming. ACM Press and Addison Wesley, New York 1998.

Teschke, T.; Ritter, J.: Towards a Foundation of Component-Oriented Software Reference Models. Presented at net.objectdays2000, sub-conference Generative and Component-based Software Engineering. Erfurt, Germany 2000

Turowski, K.: Standardisierung von Fachkomponenten: Spezifikation und Objekte der Standardisierung. In: *A. Heinzl (Hrsg.)*: 3. Meistersingertreffen. Schloss Thurgau 1999

Turowski, K.: Spezifikation und Standardisierung von Fachkomponenten. In: Wirtschaftsinformatik 43 (2001a) 3.

Turowski, K. (Hrsg.): Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten. Gesellschaft für Informatik, Arbeitskreis 5.10.3, 3. August 2001b.