Diskussionsbeitrag zur Vereinheitlichung der Spezifikation von Fachkomponenten

Jörg Ackermann

Von-der-Tann-Str. 42, 69126 Heidelberg, Deutschland, E-Mail: joerg.ackermann.hd@t-online.de

Zusammenfassung. Die geeignete Spezifikation einer Softwarekomponente ist eine wesentliche Voraussetzung für ihre erfolgreiche Wiederverwendung. Von der GI-Arbeitsgruppe 5.10.3. wird derzeit ein Vorschlag diskutiert, wie die Spezifikation für Fachkomponenten standardisiert werden kann. Dieser Beitrag enthält eine Reihe von Diskussionsbeiträgen zu diesem Vorschlag.

Schlüsselworte: Fachkomponente; Softwarekomponente; Spezifikation; Standardisierung

Die präzise und geeignete Spezifikation von Fachkomponenten ist eine wesentliche Voraussetzung, damit sich der Bau von Anwendungssystemen aus am Markt gehandelten Softwarekomponenten etablieren kann. Die Arbeitsgruppe 5.10.3. der Gesellschaft für Informatik erstellt derzeit einen Vorschlag, wie Aufbau und Inhalt einer solchen Spezifikation aussehen sollten. Für weitere Details dazu siehe das Memorandum „Vorschlag zur Spezifikation von Fachkomponenten“ (Turowski et al. 2001).

Dieser Beitrag enthält verschiedene Ergänzungen und Diskussionspunkte zu diesem Vorschlag. Es handelt sich einerseits um Vorschläge, die von mir schon ins Memorandum eingefügt wurden. Andererseits werden auch einige neue Aspekte diskutiert, und einige meiner Anmerkungen im Memorandum wurden als Ergebnis weiterer Diskussionen überarbeitet. Dieser Beitrag enthält zusammengefasst alle Ergänzungen und Änderungswünsche, die meiner Meinung nach im Memorandum berücksichtigt werden sollten. In allen anderen Punkten stimme ich den Vorschlägen im Memorandum zu.

Außerdem erstellte ich eine Fallstudie, in der eine Fachkomponente beispielhaft spezifiziert wird (siehe Ackermann 2001b). Viele meiner Diskussionsbeiträge sind aus der Fallstudie entstanden.

1 Inhalt und Aufbau

Dieser Abschnitt behandelt allgemeine Punkte zum Inhalt und zum Aufbau der Spezifikation einer Fachkomponente.

1. Die Spezifikation muss die Außensicht einer Fachkomponente *vollständig* beschreiben. Ich schlage vor, die Definition einer Fachkomponente folgendermaßen zu ändern:

> Unter der Spezifikation einer Fachkomponente wird eine vollständige, widerspruchsfreie und eindeutige Beschreibung ihrer Außensicht verstanden.

Dabei ist der Begriff “vollständig“ noch genauer zu fassen und stellt vorerst nur ein Leitbild dar. Er sollte aber an dieser Stelle nicht fehlen.

2. Eine Fachkomponente bietet anderen Komponenten ihre Dienste an. Im allgemeinen wird

sie aber auch auf die Dienste anderer Komponenten zurückgreifen. Deshalb müssen in der Spezifikation sowohl die angebotenen Dienste als auch die von der Komponente benötigten Dienste beschrieben werden. Das betrifft die Syntax-, die Verhaltens- und die Abstimmungsebene. Die benötigten Dienste sind durch eine geeignete Notation deutlich von den angebotenen Diensten zu trennen.

3. Um der Vollständigkeit gerecht zu werden, sind auch mögliche Parametrisierungen und Interaktionen sowie deren Einfluss auf das Verhalten der Komponente zu spezifizieren. Dies sollte als vorläufig offener Punkt aufgenommen werden.

4. Die bisherige Domänenebene sollte aufgeteilt werden: in eine *Terminologieebene*, die lexikonartig alle verwendeten Begriffe erläutert; und in eine *Funktionsebene*, welche Zweck und Verwendung der Fachkomponente beschreibt.

5. Ich stimme den bisherigen Diskussionen zu, dass nicht jeder neue Aspekt unbedingt eine neue Ebene der Spezifikation bilden sollte. Die relevanten Aspekte sollten zunächst gesammelt und danach zu Ebenen zusammengefasst werden.

Die Reihenfolge der Ebenen sollte so gewählt werden, dass die Spezifikation gut lesbar ist. Vom derzeitigen Diskussionsstand ausgehend, schlage ich vor, die Ebenen wie folgt anzuordnen:

> Funktionsebene; Syntaxebene; Verhaltensebene; Abstimmungsebene; Leistungsebene; Administrationsebene (könnte auch am Anfang stehen); Terminologieebene

Die Terminologieebene sehe ich als ein Lexikon, welches die Begriffe aller Ebenen erklärt, und sollte deshalb am Ende erscheinen.

6. Die formale Spezifikation sollte auf allen Ebenen durch eine Dokumentation in Prosaform ergänzt werden. Dies ist zwar in gewissem Maße redundant, erhöht aber die Verständlichkeit deutlich und senkt die Hemmschwellen, sich mit der Spezifikation auseinander zu setzen. (Eine formale Aussage ist viel besser zu verstehen, wenn man zuvor weiß, was sie ausdrücken soll.)

Die formale Spezifikation bleibt die primäre Notation, und die Prosaform bildet lediglich eine Ergänzung. Dies bedeutet, dass bei Unklarheiten in der Prosaform oder sogar Widersprüchen immer die formale Notation bindend ist. Ich halte dies für einen guten und praktikablen Kompromiss zwischen Exaktheit und Verständlichkeit.

7. Für eine größere Einheitlichkeit und bessere Verständlichkeit sind für die Spezifikation bestimmte Konventionen (im Sinne von Best Practices) wünschenswert. Erste Erkenntnisse dazu wurden in meiner Fallstudie gesammelt und dokumentiert.

2 Objektorientierung und Komponenten

Das Zusammenspiel zwischen Objektorientierung (bzw. verschiedener Arten der Implementierung) und Komponentensoftware ist noch nicht in allen Facetten untersucht und derzeit Gegenstand intensiver Diskussionen (siehe z.B. Software Development Online). Meiner Meinung nach muss deshalb genauer untersucht werden, welche Auswirkungen die Verwendung von Objektorientierung auf die Spezifikation von Komponenten hat. An dieser Stelle wird ein erster Versuch dazu unternommen.

Es gibt mindestens drei verschiedene Arten, wie Objektorientierung bei Komponenten ver-

wendet werden kann:

a. Komponenteninterne objektorientierte Implementierung

b. Komponentenübergreifende objektorientierte Implementierung (d.h. es können von außerhalb der Komponente Objekte instanziiert werden und/oder es werden Objekte an den Schnittstellen der Dienste übergeben)

c. Modellierung und Beschreibung der Komponente mit objektorientierten Konstrukten

Diese drei Arten sind weitgehend unabhängig voneinander. Dabei kann jeder Punkt einzeln oder in fast beliebigen Kombinationen auftreten. (Lediglich Punkt b. ohne Punkt a. ist schwer vorstellbar.)

Für Komponentensoftware ist es unerheblich, ob eine Komponente intern objektorientiert implementiert wurde. Punkt a. hat demnach auf die Spezifikation einer Komponente keine Auswirkungen.

Anders ist die Sache bei Punkt b. Zunächst ist zu beachten, dass alle von der Komponente erlaubten Frameworks Objektorientierung soweit unterstützen müssen, dass diese Objektreferenzen an der Schnittstelle von Diensten übergeben können. Außerdem entstehen bei Punkt b. deutlich mehr technische Abhängigkeiten und auch die Semantik wird schwerer beherrschbar. So sind Techniken wie Vor- und Nachbedingungen bei Objekten nur begrenzt einsetzbar (siehe z.B. Szyperski 1998). Als Ergebnis wird die Spezifikation komplexer, und das Information Hiding beeinträchtigt.

Deshalb sollte eine Fallstudie erstellt werden, die eine Komponente mit komponentenübergreifender Objektübergabe untersucht und spezifiziert. Die in meiner Fallstudie untersuchte Komponente erlaubt keine Objektübergabe und liefert dementsprechend keine Erkenntnisse für den Fall b.

Noch einmal anders liegt der Fall bei Punkt c. Objektorientierte Modellierungskonstrukte können nicht nur bei objektorientierter, sondern auch bei funktionaler Implementierung verwendet werden. So können alle Dienste immer in der Form *Entität::Methode* (Beispiel *Flugreise::Anlegen*) dargestellt werden. (Bei prozessartigen Diensten lassen sich Prozess-Entitäten verwenden.)

Die Verwendung objektorientierter Modellierungskonstrukte ermöglicht eine Reihe von Vorteilen:

- deutlich bessere Strukturierungsmöglichkeiten (gerade bei vielen Diensten notwendig)
- höhere Verständlichkeit
- bessere Ausdrucksmöglichkeiten in den eher objektorientierten Notationssprachen (OMG IDL, OCL)
- einfachere Wiederverwendung von Invarianten
- bessere Trennung zwischen allgemeinen und dienstspezifischen Bedingungen

Aufgrund dieser Vorteile wird die Verwendung objektorientierter Modellierungskonstrukte bei der Spezifikation von Fachkomponenten allgemein empfohlen.

3 Funktionsebene

Meiner Meinung nach ist eine deutliche Trennung zwischen einer Terminologieebene und einer Funktionsebene notwendig. Die *Terminologieebene* erläutert lexikonartig alle verwendeten Begriffe. Die *Funktionsebene* hingegen beschreibt Zweck und Verwendung der Fachkomponente. Dadurch erhalten wir eine Trennung zwischen den (in der Spezifikation) verwendeten Begriffen und den von der Komponente unterstützten Aufgaben. Wird z.B. auf der Terminologieebene eine betriebliche Aufgabe definiert, dann trifft dies noch keine Aussage darüber, ob die Fachkomponente diese Aufgabe erfüllt oder aber vielleicht von einer anderen Fachkomponente benötigt.

Die Funktionsebene sollte aus folgenden Teilen bestehen:

a. Überblicksdokumentation über Zweck und Verwendung (Prosaform)

b. Liste der unterstützten betrieblichen Aufgaben (z.B. aus Referenzmodellen)

c. Beispiele, wie die Komponente verwendet werden kann (Prosaform)

d. Überblicksdokumentation der einzelnen Dienste (Prosaform)

Zu a.: Die Spezifikation sollte eine Überblicksdokumentation in Prosaform enthalten. Diese beschreibt, welchen Aufgabenbereich die Komponente abdeckt und welche betrieblichen Aufgaben unterstützt werden. Ziel dieser Dokumentation ist, dass ein potentieller Verwender eine weitere Entscheidungshilfe hat, ob die Komponente für ihn prinzipiell interessant ist. Um dieses Ziel zu unterstützen, sollten in der Dokumentation die standardisierten fachlichen Begriffe und betrieblichen Aufgaben verwendet werden.

Dies ist zwar in gewissem Maße redundant, da unter b. eine Liste unterstützter (standardisierter) betrieblicher Aufgaben hinterlegt wird. Jedoch wird vielen Verwendern eine Prosaform im ersten Schritt lieber sein. Außerdem ist davon auszugehen, dass nicht alle Facetten möglicher Funktionalität standardisierbar sind und es Möglichkeiten zur Differenzierung geben muss. (Beispiel: Eine Komponente verwendet für eine betriebliche Aufgabe A ein neues Verfahren. Solange diese neue Variante noch nicht in der Liste betrieblicher Aufgaben als z.B. A1 standardisiert wurde, muss diese Komponente unter A klassifiziert werden und kann sich nicht von anderen Komponenten abheben.)

Zu b. Dieser Teil enthält, wie im Memorandum vorgeschlagen, die Liste aller unterstützten betrieblichen Aufgaben. Diese können sich z.B. aus Referenzmodellen ergeben.

Zu c. Es wäre sehr wünschenswert, wenn zu jeder Fachkomponente einige Beispielszenarien angegeben werden. Diese beschreiben, wie die Fachkomponente verwendet werden kann. Ein potentieller Verwender kann dadurch einschätzen, welche Möglichkeiten zur Anwendung existieren und ob diese auf seinen Fall übertragbar sind. Diese Idee folgt der Metapher eines Kochbuchs (Quelle?): Die Angabe von Farbe, Geschmack, Festigkeit, Größe, etc. einer Möhre (Spezifikation der Möhre) hilft einem Verwender nur bedingt. Werden stattdessen einige Gerichte angegeben und erklärt, wie die Möhre verwendet wird, kann der Verwender dies auf seine Situation übertragen.

Zu d. Hier werden die Dienste der Fachkomponente in Prosaform dokumentiert und deren Verwendung erklärt. An dieser Stelle kann man auch Informationen aufnehmen, die auf den anderen Ebenen nicht beschrieben werden können.

4 Syntaxebene

Auf der Syntaxebene ergeben sich für mich noch einige prinzipielle Fragen.

Zunächst einmal ist die OMG IDL eine sehr implementierungsnahe Sprache. Daraus ergeben sich Einschränkungen (siehe z.B. Ackermann 2001b), die insbesondere in der Nicht-CORBA-Welt zu Problemen führen können. Es stellt sich die Frage, ob die OMG IDL die geeignete Sprache für die Syntaxebene ist.

Ein alternativer Notationsvorschlag ist die Web Service Definition Language (WSDL). Die WSDL entwickelt sich gerade zu einem Industriestandard zur Beschreibung von (Web-)Services. Dabei werden die Services im XML-Format nach vorgeschriebenen Richtlinien beschrieben. Die verwendeten Datentypen werden als XML-Schema definiert.

Für die WSDL sprechen meiner Meinung nach die folgenden Punkte:

- konzeptuell besser geeignet, da wie bei Komponenten Dienste und keine Interfaces beschrieben werden
- bessere Abstraktion von Implementierungsdetails
- bessere Ausdrucksmöglichkeiten als die OMG IDL (die in der Fallstudie aufgetretenen Probleme lassen sich mit der WSDL vermeiden)
- direkte Einbindung der Dokumentation möglich.

Für mich ergibt sich daraus die allgemeinere Frage, was das genaue Ziel der Beschreibung auf der Syntaxebene ist.

a. Ist die Syntaxebene für einen menschlichen Leser bestimmt (im Sinne einer Dokumentation)?
b. Oder enthält die Syntaxebene auch schon die exakte Aufrufsyntax der Dienste?

(In meiner Fallstudie beschreibt die Syntaxebene die Dienste hinreichend genau, damit ein Leser eine gute Vorstellung von der Syntax der Dienste erhält. Die dargestellte OMG IDL-Syntax ist jedoch nicht so exakt, dass damit auch ein Aufruf möglich wäre.)

Beschränkt sich das Ziel auf Punkt a, dann ist meiner Meinung nach die OMG IDL zu implementierungsnah und zu einschränkend. Stattdessen wäre z.B. die WSDL besser geeignet.

Bezieht sich das Ziel auf den Punkt b, sollte die Verwendung der OMG IDL auch noch einmal überdacht werden. Denn mit der Festlegung der OMG IDL als Spezifikationssprache fordern wir eine OMG-kompatible Implementierung. Dies schränkt aus meiner Sicht die Komponente sehr ein. Eine Alternative könnte in diesem Falle eine Notation sein, die Komponenten-Framework spezifisch ist. Der große Vorteil wäre, dass die Notationssprache der Syntax der Komponente angepasst ist. Der Nachteil wäre, dass die Einheitlichkeit verloren geht. Außerdem müsste ein Komponentenintegrator die Syntax des Komponenten-Framework kennen. Dies ist allerdings kein kritischer Punkt, da er bei der Integration diese Kenntnisse sowieso benötigt.

5 Verhaltensebene

Die Verhaltensebene sollte ein UML-Modell enthalten. Dieses stellt alle wesentlichen Spezi-

fikations-Entitäten und deren Beziehung zueinander dar. Das Modell bildet die Grundlage für die auf der Verhaltensebene formulierten OCL-Bedingungen. Die Verwendung eines Modells bringt eine Reihe von Vorteilen mit sich. Für eine ausführliche Diskussion dazu siehe die Fallstudie (Ackermann 2001b).

In manchen Fällen ist es möglich, eine Bedingung entweder auf der Verhaltensebene oder der Abstimmungsebene zu beschreiben. Ein Beispiel hierfür findet sich ebenfalls in meiner Fallstudie. Ohne weitergehende Konventionen ergibt sich damit ein Interpretationsfreiraum, auf welcher Ebene bestimmte Bedingungen auszudrücken sind. Dies sollte meiner Meinung nach vermieden werden. Ich schlage deshalb vor, dass eine Bedingung möglichst immer auf der Verhaltensebene mit Hilfe der OCL formuliert wird. Nur wenn eine Bedingung nicht mit Hilfe der OCL formulierbar ist, sollte diese mit Hilfe der temporalen Operatoren auf der Abstimmungsebene ausgedrückt werden. Für weitere Details und eine Diskussion der Vor- und Nachteile siehe wieder die Fallstudie.

6 Abstimmungsebene

Als Notation auf der Abstimmungsebene wurde die OCL, erweitert um einige temporale Operatoren, vorgeschlagen. Dazu sind meiner Meinung nach einige Präzisierungen notwendig.

Durch die temporalen Operatoren entstehen einige Einschränkungen bei der OCL, die explizit formuliert werden sollten:

- Verzicht auf die Auswertbarkeit eines OCL-Ausdrucks zu einem bestimmten Zeitpunkt
  - Der Wert eines OCL-Ausdrucks muss jederzeit (bei Vor- und Nachbedingungen bei der Methodenausführung) ermittelbar sein. Dies ist bei zukunftsgerichteten Operatoren nie möglich und auch bei vergangenheitsgerichteten Operatoren oft nur theoretisch möglich.
- Zulassen von Nicht-Query-Methoden in temporalen OCL-Ausdrücken
  - Auf der Abstimmungsebene müssen oft Abhängigkeiten zwischen ändernden Methoden beschrieben werden. Eine Einschränkung auf Query-Methoden (wie in OCL notwendig) ist nicht sinnvoll.
- Als Konsequenz ergibt sich, dass die temporalen Operatoren in der Spezifikation nur als Modellierungskonstrukt für die Mensch-Mensch-Kommunikation verwendet werden. Eine maschinelle Auswertbarkeit ist zunächst nicht geplant.

Wegen dieser Einschränkungen sehe ich die temporalen Operatoren nicht als eine Erweiterung innerhalb der OCL. Stattdessen schlage ich vor, die OCL und eine „Temporale Form der OCL“ als verschiedene Sprachen zu positionieren (die syntaktisch und semantisch weitgehend übereinstimmen). Entsprechend sollte auf der Verhaltensebene einer Fachkomponente immer die OCL selbst (ohne obige Einschränkungen) und auf der Abstimmungsebene die „Temporale Form der OCL“ verwendet werden. (Vielleicht findet sich ja noch ein besserer Name für die temporale Form.)

7 Administrationsebene

Die Spezifikation einer Fachkomponente sollte Hinweise zu Installation und Betrieb enthalten. Darunter fallen insbesondere die Systemvoraussetzungen zum Betrieb der Komponente

(wie benötigte Systemarchitektur oder mögliche Komponenten-System-Frameworks, auf denen die Komponente lauffähig ist). Dazu hatte ich im Memorandum eine Installationsebene angeregt.

Ich schließe mich aber dem Gegenvorschlag an, diese Informationen auf der Administrationsebene anzusiedeln.

8 Terminologieebene

Im Memorandum wird kontrovers diskutiert, ob die Terminologieebene alle oder nur die von einem Standard abweichenden Begriffe enthalten sollte. Meiner Meinung nach müssen alle in der Spezifikation verwendeten Begriffe erklärt werden. Bei standardisierten Begriffen genügt ein einfacher Verweis auf den Standard. Alle anderen Begriffe müssen komponentenlokal definiert werden. Dies erscheint sinnvoll, da es derzeit verschiedene Standardisierungsbemühungen gibt (z.B. für verschiedene Branchen) und wohl auf absehbare Zeit ein Begriff immer nur mit seinem Standard eindeutig sein wird.

Literatur

Ackermann, J.: Fallstudie zur Spezifikation von Fachkomponenten. In: K. Turowski (Hrsg.): Modellierung und Spezifikation von Fachkomponenten: 2. Workshop, Bamberg, 2001.

Software Development Online: Beyond Objects. Discussion column by different authors. URL: http://www.sdmagazine.com./features/uml/beyondobjects/?topic=uml. Abruf am 2001-08-03.

Szyperski, C.: Component Software: Beyond Object-Oriented Programming. 2. Aufl., Addison-Wesley, Harlow 1998.

Turowski, K., et al.: Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten. Diskussionsvorschlag des GI-Arbeitskreises 5.10.3, 2001. URL: http://www.fachkomponenten.de. Abruf am 2001-08-03.