Klaus Turowski
(Hrsg.)

Tagungsband

4.Workshop Modellierung und Spezifikation von Fachkomponenten

10.Oktober 2003

Bamberg

Gesellschaft für Informatik
Arbeitskreis 5.10.3
Komponentenorientierte betriebliche Anwendungssysteme

Tagungsleitung

Prof. Dr. Klaus Turowski
Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II
Universität Augsburg
Universitätsstraße 16, 86135 Augsburg
Phone: +49(821)598-4431; Fax : -4432
E-Mail: klaus.turowski@wiwi.uni-augsburg.de
URL: http://wi2.wiwi.uni-augsburg.de

Programmkomitee

Prof. Dr. S. Conrad
Ludwig-Maximilians-Universität München

Prof. Dr. G. Goos
Universität Karlsruhe

Prof. Dr. P. Loos
Johannes Gutenberg-Universität Mainz

Prof. Dr. E. Ortner
Universität Darmstadt

Stephan Sahm
itelligence AG

Prof. Dr. K. Turowski (Vorsitz)
Universität Augsburg

ISSN 1619-8980

Vorwort

Auf dem ersten Workshop *Modellierung und Spezifikation von Fachkomponenten*, der am 12. Oktober 2000 vom Arbeitskreis 5.10.3 *Komponentenorientierte betriebliche Anwendungssysteme* der Gesellschaft für Informatik (GI) im Rahmen der Fachtagung *Modellierung betrieblicher Informationssysteme (MobIS) 2000* an der Universität Siegen veranstaltet wurde, wurde vereinbart, einen Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten zu verfassen. Die Teilnehmer des Workshops waren aufgefordert, sich an der Erstellung eines gleichnamigen Memorandums zu beteiligen, das seit Februar 2002 in einer ersten konsolidierten Fassung vorliegt.

Für diesen vierten Workshop wurde darum gebeten, Diskussionsbeiträge für die Weiterentwicklung des Memorandums (http://www.fachkomponenten.de) einzureichen. Ausgehend von dieser Themenstellung, wurde um die Einreichung von Beiträgen gebeten, die insbesondere folgende Themen aufgreifen:

- die Spezifikation nicht-funktionaler Eigenschaften,
- Vorgehensmodelle zur memorandumskonformen Spezifikation sowie
- die Übertragbarkeit des Spezifikationsrahmens auf andere, nicht-betriebliche Anwendungsdomänen.

Der vorliegende Tagungsband enthält die schriftliche Fassung der zu Vortrag und Veröffentlichung angenommen Beiträge des vierten Workshops *Modellierung und Spezifikation von Fachkomponenten*. Der Workshop wurde vom Arbeitskreis WI-KobAs (5.10.3) und dem *Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II* der Universität Augsburg am 10. Oktober 2003 im Rahmen der *11. Fachtagung Modellierung betrieblicher Informationssysteme (MobIS 2003)* in Bamberg veranstaltet.

Abschließend sei allen gedankt, die durch die Einreichung eines Beitrags zum Gelingen des Workshops beigetragen haben. Besonderer Dank gebührt den Mitgliedern des Programmkomitees für die Begutachtung der eingegangenen Beiträge und Herrn Johannes Maria Zaha für die Mitwirkung bei der Organisation des Workshops.

Augsburg, im September 2003

Klaus Turowski

Inhaltsverzeichnis

Spezifikation von Fachkomponenten mit der Swiss Re Data Language

Hans Wegener[†], Gunnar Auth[‡]

[†]*Swiss Re, Mythenquai 50/60, 8022 Zürich, Schweiz, Hans_Wegener@swissre.com*

[‡]*DaimlerChrysler TSS GmbH, Lise-Meitner-Strasse 15, 89081 Ulm, Deutschland, gunnar.auth@daimlerchrysler.com*

Zusammenfassung: Die Swiss Re Data Language ist ein Mittel zur Spezifikation von Datenschnittstellen auf Basis vereinheitlichter Fachbegriffe. Unsere Erfahrungen mit dem nun produktiven Release 4 zeigen, dass ein auf formaler Spezifikation basierender Ansatz schnell auf praktische Grenzen trifft. Zudem muss die Ambiguität von Spezifikationen nach unserer Erfahrung immer im Verwendungskontext betrachtet werden. Daher schlagen wir vor, Wiederverwendung von zwischen Fachkomponenten ausgetauschten Daten nicht nur über technische, sondern zusätzlich über eine Kombination ablauf- und aufbauorganisatorischer Mittel zu verfolgen.

Schlüsselwörter: Referenzdaten, Attributtyp, Temporale Datenbank, Konfigurationsmanagement.

1 Einleitung

Ein wesentliches Problem beim Entwurf von Spezifikationssprachen ist die Balance zwischen der Beschreibungsfähigkeit auf der einen und Verständlichkeit für den Menschen auf der anderen Seite: üblicherweise sind in Grossunternehmen Menschen am Entwurf von Softwaresystemen beteiligt, die keine oder nur geringe Ausbildung im Bereich formaler Sprachen aufweisen. Dennoch sind sie mehr und mehr gezwungen, diese in ihrer Arbeit zu verwenden. Die Gebrauchstauglichkeit solcher Mittel wird dadurch zum Erfolgskriterium.

Wir berichten von den Erfahrungen, die wir im Rahmen eines jetzt abgeschlossenen Projekts mit dem Entwurf einer Spezifikationssprache für Referenzdaten gemacht haben. Ziel ist darzustellen, welche Techniken aus dem formalen Bereich durch welche informellen ergänzt werden können, so dass die resultierende Sprache die erforderliche Präzision nicht verliert.

Der Beitrag gliedert sich wie folgt: Zur Schaffung einer einheitlichen begrifflichen Grundlage werden in Abschnitt 2 zunächst die SDL und ihre Grammatik kurz vorgestellt. Die besondere Bedeutung des Change Managements und der organisatorischen Einbettung wird in Abschnitt 3 herausgestellt. Anschliessend beschreibt Abschnitt 4 die Anwendung der SDL im Rahmen der Spezifikation von Fachkomponenten. Einen zentralen Bestandteil des Beitrags bildet die Darstellung der bisher gemachten Erfahrungen in Abschnitt 5. Der Beitrag endet mit einer Zusammenfassung und einem Ausblick auf die zukünftige Entwicklung.

2 Die Grammatik der SDL

Die Swiss Re verfolgt einen strukturierten Ansatz zum Management ihrer Informationsarchitektur, der in [Mart2003] ausführlicher beschrieben ist. Als Grundlage für das Referenzdatenmanagement wurde die *Swiss Re Data Language* (SDL) zur einheitlichen Spezifikation von Daten und Datenschnittstellen entworfen. Innerhalb des Informationsmanagements ist die SDL das zentrale Mittel zur Beschreibung von applikationsübergreifenden Referenzdaten. Die Bezeichnung „Data Language“ deutet bereits

auf den angestrebten Zweck hin, Syntax und Semantik von Begriffen zu definieren, die bei der Datenverarbeitung und insbesondere dem Berichtswesen der Swiss Re eine Relevanz haben. Die SDL wird hier eingesetzt zur Spezifikation von Datenschnittstellen zwischen Fachapplikationen, genauer: von Attributen in (Kontext-)Entitätstypen.

Die Entwicklung und Betreuung der SDL ist organisatorisch im Referenzdatenmanagement der Swiss Re angesiedelt. In einem ersten Projekt zur Umsetzung des SDL-Konzepts wurde damit begonnen, die bisher getrennten Begriffsysteme der Bereiche Finance (FSA-DL) und Reinsurance (RSA-DL) in der SDL zusammenzuführen. Im Zuge der Integration wurden die ebenfalls getrennten Applikationen zur Verwaltung von FSA-DL und RSA-DL von einer neuen SDL-Applikation abgelöst. Die wichtigsten Bestandteile der technischen Lösung sind ein Repository zur Verwaltung von Glossaren und Taxonomien, sowie eine webbasierte Administrationskomponente und ein Frontend für die Suche nach Datenstandards. Im folgenden wird die Grammatik der SDL anhand lexikalischer, syntaktischer und semantischer Aspekte kurz beschrieben.

2.1 Lexikalische Aspekte

Die Festlegung lexikalischer Aspekte ist in einem global agierenden Unternehmen durchaus von Wichtigkeit. Da immer noch viele (so auch unsere) Betriebs- und Datenbanksysteme auf 8-Bit-Zeichensätzen beruhen, muss man exakt spezifizieren, wie Sonderzeichen (Umlaute und Akzentzeichen, Formelzeichen wie Indexierung und Exponentiation, Anführungsstriche) behandelt werden. Dies betrifft insbesondere Freitext in Begriffsbeschreibungen.

Sprachelemente der SDL basieren auf einer durch die Implementation bedingten Teilmenge des Zeichensatzes ISO 8859-1. Offizielle Sprache der Swiss Re ist Englisch, wodurch nur wenige Sonderzeichen unterstützt werden müssen. Für bestimmte Zeichen, die nicht Bestandteil der ISO 8859-1 sind, wurde spezifiziert, wie dorthin transformiert werden; z.B. werden die in Microsoft-Word-Dokumenten häufig auftretenden Anführungsstriche „“ nach "" umgewandelt.

2.2 Syntaktische Aspekte

In der SDL modellieren wir die Terminologie und Taxonomie unserer fachlichen Welt. Begriffe repräsentieren atomare fachliche Abstraktionen. Homonyme sind möglich. Begriffe teilen sich auf Repräsentanten fachlicher Wertmengen (Attribute) und die Werte selbst.

Beziehungen etablieren höherwertige Abstraktionen auf Basis von Begriffen, wobei folgende Beziehungstypen unterstützt werden:

- Begriffsverwandschaft: wird zwischen zwei Begriffen etabliert, wenn diese miteinander fachlich verwandt sind (z.B. „Currency“ und “Currency Block“). Die Beziehung ist ungerichtet und wird zum besseren Verständnis des Vokabulars eingesetzt.
- Wertehierarchie: wird zwischen mehreren Werten etabliert und drückt eine Organisation der Begriffswelt in Unter- und Oberbegriffe aus (z.B. „All Values of Line of Business“ wird aufgeteilt in „Life“ und „Non-Life“, wobei „Non-Life“ Oberbegriff für „Property“, „Marine“, „Liability“ und andere ist). Über Hierarchien werden insbesondere Aufzählungstypen (RDBMS, ROLAP) und Dimensionen (MOLAP) modelliert.
- Attributhierarchie: wird aus Attributen gebildet, um ein hierarchisches Datenmodell aufzubauen, wie es in der Finanzwelt auftaucht („General Ledger“). Zwei ausgewiesene Dimensionen, der Kontenplan und die so genannten Sichten ([Kontenplan-]Varianten) stellen die Grundstruktur her. Aus diesen wird in Form von Tripelgruppen für jede

auftretende Konto-Sicht-Kombination modelliert, welche finanziellen Fakten für eine Sicht (z.B. Halbjahresbericht, Abschlussbericht) zu berichten sind.

Die durch Werthierarchien hergestellten Artefakte werden von uns aufgrund ihrer Bedeutung für Referenzdaten als Referenzbäume bezeichnet. Attributhierarchien werden wegen ihrer spezialisierten Verwendung Kundenbäume genannt.

Die SDL beschreibt anhand von Begriffen Eigenschaften der Schnittstellenebene. Im Datenaustausch zwischen Applikationen werden Repräsentationen, so genannte Codes für die Konstituenten eines Fakts verwendet. Kontexte werden verwendet, um Begriffe zu ihren Codes zuzuordnen. Zum Beispiel ist in unserem Währungs- und Deviseninformationssystem der Begriff „Swiss Franc“ durch die Zahl „1000“ repräsentiert, während die ISO-4217 die Zeichenkette „CHF“ wählt.

Kontexte werden auch für die Definition von Synonymen verwendet. Für jeden Begriff kann in einem Kontext ein so genannter Alias vergeben werden, z.B. „Branche“ für „Line of Business“. Bei der Suche im SDL Tool wird, wenn ein Kontext selektiert wurde (vgl. Bild 1), zusätzlich zum eigentlichen Begriffsnamen auch der Alias zur Trefferermittlung beigezogen.

Die syntaktischen Einschränkungen wollen wir hier nicht in ihrer Gesamtheit aufführen, sondern nur die wichtigsten nennen:

- Repräsentant eines Referenzbaums ist ein Attribut, alle Knoten inklusive der Wurzel sind Werte.
- Ein Attribut ist maximal einem Referenzbaum zugehörig, Werte können in mehreren Referenzbäumen auftreten.
- Der gleiche Begriff darf nur einmal in einem Referenzbaum auftreten.
- Referenz- und Kundenbäume sind zyklenfrei.
- Innerhalb eines Kontexts sind die Codes der Begriffe in Bezug auf die sie enthaltenden Referenzbäume eindeutig.
- Ein Kundenbaum darf nur Referenzbäume enthalten, in denen sämtliche Begriffe im Kontext, mit dem der Kundenbaum assoziiert ist, codiert sind.

Wir unterscheiden zwischen harten („Fehler“) und weichen („Warnung“) Einschränkungen. Letztere können vom Administrator überschrieben werden.

2.3 Semantische Aspekte

Die Definition legt die dem Begriff zu Grunde liegende Semantik fest, die Beschreibung fügt informelle Erläuterungen an, wie etwa Einsatzbeispiele. Die Semantik des Begriffs ist dadurch aber noch nicht unbedingt kontextinsensitiv festlegbar. Beispielsweise ist die Interpretation der Definition eines Kontos abhängig von den verwendeten Rechnungslegungsgrundsätzen (z.B. Swiss GAAP und US GAAP). Hier helfen Kommentare, welche die kontextsensitiven Bestandteile der fachlichen Interpretation eines Begriffs enthalten, während die Definition allgemeingültig formuliert ist. Kommentartypen, denen ein Kommentar immer eindeutig zugeordnet ist, identifizieren den Interpretationskontext.

Werden Werte zusammengefasst, so beschreibt diese Menge eine fachlich bedingte Gruppe, die in der Realität auftritt. Es handelt sich um Aufzählungstypen, deren Handhabung durch die Form des Referenzbaums bestimmt wird. Die natürliche Ordnung orientiert sich an der Hierarchie des Baums. Die Interpretation in analytischen Systemen, insbesondere multidimensionalen Datenbanken, folgt dieser (Dimensions-)Hierarchie. Durch die Struktur

des Baums wird ausgedrückt, welche Fakten wie aggregiert werden. Fakten eines Unterbegriffs werden entlang der Relation aggregiert und so mit dem Oberbegriff assoziiert.

3 Change-Management und die SDL

Im Entwurf von SDL R4 spielte das Change-Management eine zentrale Rolle. Jedes Objekt in unserem Repository („SDL Tool“) erhält eine global eindeutige Identifikationsnummer (SDL-ID), die sich während des gesamten Lebenszyklus’ nicht ändert. Sowohl Fachapplikationen als auch Menschen greifen auf das Repository zu (vgl. Bild 1), aber die SDL-ID ist (wegen ihrer Eindeutigkeit) vor allem für die Applikationen von Bedeutung.

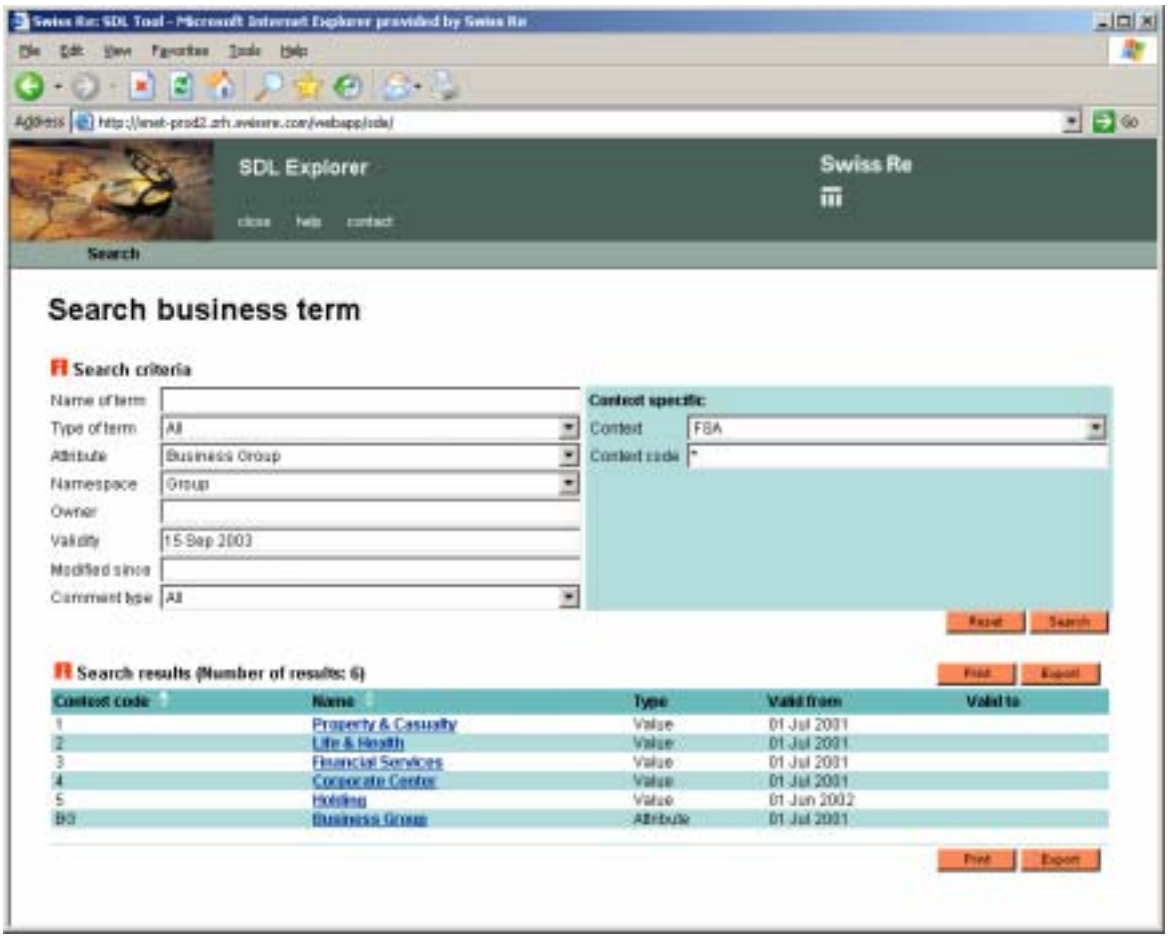


Bild 1: Der Inhalt des SDL Repositories ist im wesentlichen über zwei Modelle zugänglich; temporale Selektion (hier unter „Validity“ angegeben) wird vor allem von Fachbereichsvertretern genutzt; Applikationen nutzen sie ebenfalls, greifen aber zuweilen auch (über die nicht dargestellten Systemschnittstellen) auf die an Schnappschüssen orientierte Selektion zurück.

3.1 Objektgültigkeit und Objektversion bei Konfigurationsverzweigung

Das Versionenmodell verdient eine Erläuterung; es ist eine Mischung aus temporaler und schnappschussbasierter Handhabung. Die Gültigkeitsperiode eines Objekts wird neben der Versionsnummer zur Ermittlung der aktuell gültigen Version eines Objekts hinzugezogen, weil aufgrund von Korrekturen, Vorausarbeit etc. das Objekt mit der höchsten

Versionsnummer nicht automatisch das aktuell gültige ist (vgl. Abschnitt 3). Die aktuell gültige Objektversion ist:

- publiziert,
- hat ihren Gültigkeitsbeginn heute oder in der Vergangenheit und
- wenn es mehrere davon gibt, die höchste Versionsnummer, aber
- wenn ihr Gültigkeitsende in der Vergangenheit liegt, gibt es keine gültige Version.

Diese scheinbar umständliche Logik erklärt sich aus den konfligierenden Bedürfnissen von Fachbereich und IT. Aus Sicht der Fachabteilungen ist Zeit das wesentliche Kriterium für Selektion von Begriffs- und Baumversionen, da sich die geschäftlichen Abläufe − für ein Finanzserviceunternehmen nicht weiter verwunderlich − vor allem nach den Zeitpunkten von Transaktionen ausrichten. Das hat allerdings den unangenehmen Seiteneffekt, dass die Selektionsergebnisse zeitlich variieren. Um das im Umgang mit Referenzdaten zu vermeiden, bevorzugt die IT die schnappschussbasierte Variante. Ein Konflikt tritt auf, wenn konsekutive Versionen von Repositoryobjekten nicht konsekutiv in der Zeit geordnet sind.

Dies passiert relativ häufig beim Kontenplan (vgl. Bild 2). Konten wie „Accrued interest and rent“ oder „Real estate held for sale“ sind im Referenzbaum „Account“ enthalten. Versionen von Referenzbäumen und Begriffen können unabhängig voneinander publiziert werden. Gängige Praxis ist z.B., dass neue Konten geöffnet und alte geschlossen oder in der Semantik verändert werden (vgl. die sich ständig verändernden Rechnungslegungsgrundsätze). Dadurch wird nicht nur das Modell der gedachten Zukunft verändert (d.h. zukünftig gültige Versionen publiziert), sondern durchaus auch die Gegenwart (d.h. existierende gültige Versionen durch Publikation einer neuen überschrieben). Es kann also durchaus vorkommen, dass Version 1 und 3 heute gültig wären, Version 2 und 4 aber erst nächstes Jahr.

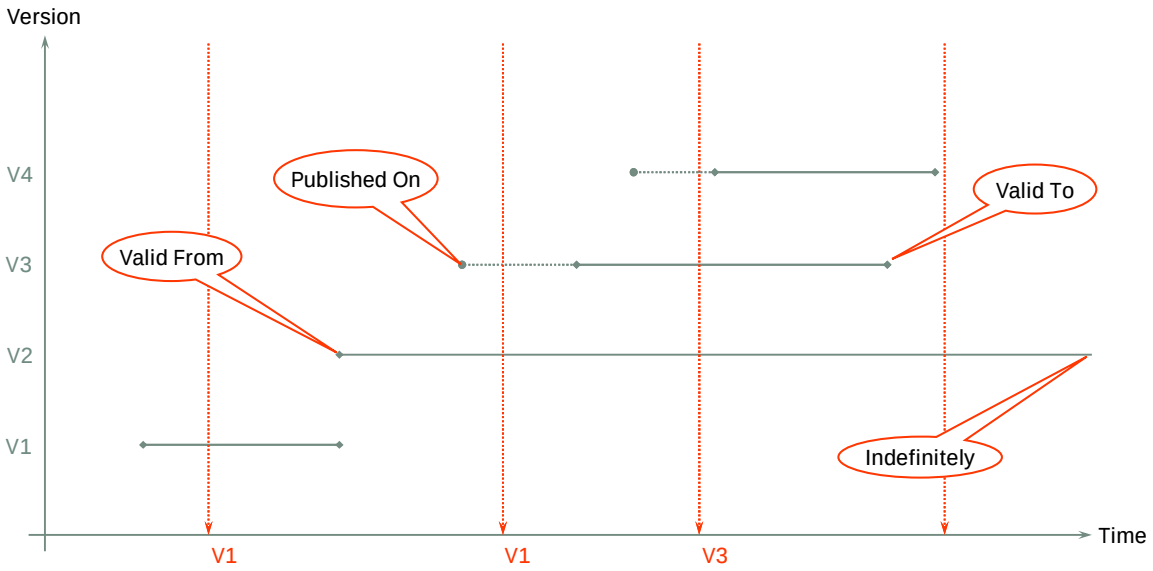


Bild 2: Veranschaulichung der Definition von „gültige Objektversion“. Zu den vier dargestellten Zeitpunkten ist die jeweils am unteren Rand bezeichnete Version gültig. Zum letzten Zeitpunkt ist keine Version gültig, weil Version 4 Version 1 maskiert; sie ist nicht mehr gültig, war aber in der Vergangenheit gültig.

Offensichtlich werden durch diese Modellierung implizit mehrere Konfigurationszweige verwaltet, weil das Werkzeug dazu keine entsprechende Abstraktion anbietet (zu den Gründen, warum das nicht so ist siehe Abschnitt 3.1). Ein Ausweg ist, Zeit und Version in der Bewertung der Gültigkeitsfrage zusammenzulegen, weil Zeit auch bei der Selektion eine dominante Rolle spielt. Ein Konfigurationszweig umfasst die Summe aller während einer gewählten Periode gültigen Repositoryobjekte.

3.2 Konsistenzsicherung im Rahmen des Change-Management

Das SDL Tool unterstützt zur Konsistenzsicherung bei Veränderungen einen bestimmten Workflow. Jede neue Version eines Objekts wird zunächst modelliert und dann zur fachlichen Prüfung eingereicht. Vor der Prüfung nimmt das Repository eine Konsistenzanalyse vor, die syntaktische Fehler und Warnungen (vgl. Abschnitt 2.2) erfasst. Nachdem diese bereinigt bzw. explizit überstimmt worden sind, prüft ein anderer Administrator („Vier-Augen-Prinzip“) die Änderung semantisch, d.h. nach fachlichen Kriterien. Hier werden insbesondere Doppelspurigkeiten und Widersprüchlichkeiten abgefangen, die sich einer technischen Analyse entziehen, und es kann noch einmal geprüft werden, ob die überstimmten Repositorywarnungen tatsächlich fachlicher Notwendigkeit entsprechen (sie werden vor der Publikation erneut angezeigt).

Diese doppelte Konsistenzsicherung hat einerseits den Zweck, die Definition der SDL zu entlasten (weniger Abstraktionen und Konsistenzregeln). Andererseits soll die Syntax nicht ohne weiteres so weit „abgeschwächt“ werden, dass jeder Administrator modellieren kann, was ihm bzw. ihr beliebt.

Tatsächlich gibt es z.B. Unterschiede in der Behandlung ungültig gewordener Begriffe. Während die Finanzabteilung (aus Gründen der Darstellung des Geschichtsverlaufs) selbige im Referenzbaum belässt, ist es im Bereich Reinsurance üblich, eine neue Baumversion anzulegen, in welcher der ungültige Begriff nicht mehr auftaucht. Da sich beide Bereiche gewisse Attribute (d.h. Referenzbäume) teilen, ist es unabdingbar, dass die Abstimmung im Rahmen eines Geschäftsprozesses abläuft, weil Technik diese Komplexität nur zu sehr hohen Kosten abzubilden vermag.

4 Spezifikation von Fachkomponenten mit der SDL

Folgende Aspekte der (Daten-)Schnittstelle einer Fachkomponente können durch einfachen Verweis auf die SDL hinreichend festgelegt werden:

- Fachliche Bedeutung: durch die Definition, Beschreibung und Kommentare eines Begriffs wird die fachliche Semantik eines Werts (zum Teil kontextsensitiv) angegeben. Unsere Finanzabteilung publiziert bereits seit längerem ein Glossar auf Basis der SDL, das weltweit als fachliche Grundlage für Daten liefernde Systeme dient.
- Validierungsregeln: durch den Aufbau eines Attributs bzw. eines Referenzbaums wird implizit formuliert, welche Werte eine Datenschnittstelle ablehnen kann bzw. annehmen muss. Der Operational Data Store im Geschäftsbereich Property & Casualty arbeitet bereits heute nach diesem Prinzip. Weiterhin wird über die Gültigkeitsperiode ersichtlich, welche Fakten zu welchem Zeitpunkt geladen werden können bzw. als ungültig abgelehnt werden müssen. So ist es beispielsweise nicht mehr möglich, Neugeschäft in der Währung „German Mark“ zu buchen, wohl aber Altgeschäft.

- Aggregationsregeln: alle Applikationen, die als „SDL-compliant“ bezeichnet werden, aggregieren Dimensionsdaten in gleichförmiger Weise, nämlich entlang der Taxonomie des Referenzbaums.
- Repräsentation: mittels Kontexten entkoppelt man die fachliche Ebene (den Begriff) von der technischen (dem Code). Der Kontextcode eines Werts kann eindeutig mit selbigem assoziiert werden, wenn bekannt ist, welches Codesystem die liefernde Komponente verwendet.
- Change-Management: durch Angabe der Wahl des Selektionsmechanismus' für Objektversionen wird auch der Umgang mit Veränderungen vollständig oder teilweise festgelegt.

Wiederverwendung von Datenstandards wird mit der SDL also auf verschiedenen Ebenen ermöglicht bzw. vereinfacht (vgl. [Krue1992]):

- Abstraktion: die auf fachsprachlicher Ebene vorhandene Assoziation von Attributen mit Wertmengen wird mit Referenzbäumen formalisiert und damit technisch handhabbar gemacht. Über Kontexte wird die technische Repräsentation von der fachsprachlichen Seite der Begriffe entkoppelt.
- Selektion: das SDL Tool ermöglicht die flexible und effiziente Suche nach vorhandenen Daten(typ)standards und ein präzises Verständnis ihrer fachlichen Bedeutung.
- Integration: die Bereitstellung temporaler und schnappschussbasierter Behandlung von Objektversionen ermöglicht es Eigentümern von Komponenten, sich schnell und eindeutig auf die Modalitäten des Umgangs mit Veränderungen an der Datenschnittstelle zu einigen.
- Spezialisierung: geschieht im Bereich der Werte durch Aufbau von Taxonomien.

Die SDL deckt selbstverständlich nur Attributtypen ab, wenn man von den Kundenbäumen absieht; Entitätstypen hingegen stehen nicht im Fokus. Wiederverwendung ist hier Gegenstand anderer, semiformeller Massnahmen, vgl. Abschnitt 6.

In der Swiss Re sind zurzeit Untersuchungen im Gange sind, um die Möglichkeit der Spezifikation weiterer Abbildungsregeln im Rahmen des Datenaustauschs zu klären. Da die Abtrennung zur Funktionalität klassischer ETL-Tools allerdings schwierig ist, haben wir noch keinen Entwurfsentscheid getroffen.

Zur Veranschaulichung werden im folgenden zwei kurze Beispiele für die Spezifikation von Datenschnittstellen mit der SDL beschrieben

4.1 Beispiele

Die Analyseapplikation „Group Performance Management“ hat eine Datenschnittstelle beschrieben, die eine (aus etwa 20 Entitätstypen gebildete) Kollektion von 12 Dimensionen und einer Masszahl (sowie zwei Hilfsattributen) umfasst. Diese können grundsätzlich ohne Angabe ihrer konkreten Werte genannt werden. Allerdings sind bei dieser Applikation alle Werte unterhalb „Life“ aus dem Referenzbaum „Line of Business“ ausgeschlossen, genauso wie bestimmte Werte im Bereich „Non-Life“. Diese müssen nur aufgezählt werden.

Die Analyseapplikation „Technical Steering Values“ hat eine Datenschnittstelle beschrieben, die 12 Attribute, 3 Masszahlen und 2 Daten umfasst. Als Repräsentation werden die Codes der RSA-DL gewählt. Auch hier wird durch den Verweis auf definierte Datenstandards und Repräsentationsformen der Aufwand für die Spezifikation signifikant reduziert.

Beide Applikationen basieren auf Schnappschüssen der Referenzdaten; Daten liefernde Applikationen müssen sich entweder auf diese einstellen oder ihre Daten vorher entsprechend transformieren.

5 Erfahrungen

Die hier beschriebenen Erfahrungen beziehen sich auf die Periode von Februar 2002 bis September 2003. Während dieses Zeitraums wurde in der Swiss Re bereits das bestehende Werkzeug SDL R3 betrieben und als Grundlage für den Entwurf von SDL R4 genutzt. Letzteres ist seit Juni 2003 im produktiven Einsatz und wird global genutzt. Insgesamt drei Administratoren in den Vereinigten Staaten und sechs in der Schweiz verwalten Daten für das Corporate Center und den Geschäftsbereich Property & Casualty.

5.1 Zur Homonymproblematik: Kontextsensitivität der Begriffsintension

Das Auftreten von Homonymen wurde von uns in früheren Entwürfen von SDL R4 verboten [Wege2003]. Um verschiedenen Vokabularen Rechnung zu tragen, wurden sie in disjunkte Namensräume unterteilt. Dies stellte sich als nicht machbar heraus. So treten im Kontenplan äusserst häufig Namensdubletten (wie etwa „Other Assets“) auf. Die Semantik eines jeden solchen Begriffs ergibt sich erst aus seiner Position in der Hierarchie. Auf die Vergegenständlichung dieses Zusammenhangs in der Grammatik der SDL wurde aus Komplexitätsgründen verzichtet, gleichfalls kam eine Umbenennung („Other Assets 1..n“) nicht in Frage. Zudem werden Konten mit hierarchisch aufgebauten Codes versehen, wodurch für den Menschen wieder akzeptable Eindeutigkeit hergestellt wird.

Diese Erfahrung wirft die Frage auf, was ausser dem Namen eines Fachbegriffs als Merkmal für die eindeutige Bestimmung der Intension angegeben werden muss und welche Mittel akzeptabel sind. Üblicherweise geht eine formale Spezifikation davon aus, dass das darin verwendete Glossar homonymfrei ist. Falls das nicht so ist, muss noch der Zusammenhang angegeben werden, um Eindeutigkeit herzustellen. Bei uns wird die Intension über mehrere Aspekte näher bestimmt, nämlich die Definition, die Beschreibung, die Lage des Begriffs im Referenzbaum sowie allfällig vorhandene Kommentare. (Man beachte, dass ein Begriff auch in mehreren Referenzbäumen auftreten kann.) Die fachsprachliche Realität stellt sich also als relativ komplex dar (vgl. Bild 3).

Eine grosse Herausforderung bei der Bereitstellung von Modellierungsmitteln zur formalen Spezifikation von Komponenten ist die Identifikation der angemessen Zahl und Granularität. Eine zu nahe an der fachsprachlichen Realität orientierte Welt wird wegen ihrer Komplexität wenig oder keine Benutzerakzeptanz finden, während eine zu realitätsfremde Welt vielleicht noch verwendet werden wird, aber keine Aussagen von Interesse zu treffen vermag. Nach unserer Erfahrung sind bei der Güterabwägung vor allem folgende (bei uns beobachtbare) Effekte nutzbar:

1. Es ist zumeist klar, in welchem Fachgebiet man sich bewegt. Je kleiner die fachlichen Überschneidungen zwischen den Gebieten, desto geringer ist die Bedeutung von Homonymen.
2. Innerhalb eines Verwendungskontexts treten Homonyme seltener bei Attributen als bei Werten auf. Von 81 Attributen tritt bei uns nur ein einziges doppelt auf (1.2%), und das nur auf Grund eines Missverständnisses während der Datenmigration. Von 4089 Werten sind allerdings 357 Dubletten bzw. Tripletts zu vermelden (8.7%).

3. Innerhalb eines Fachgebiets ist so viel (teilweise redundante) Information vorhanden, dass Homonyme keine grossen Ambiguitäten erzeugen.

Diese Effekte erlauben es uns, bei den Abstraktionen der SDL auf einen Teil der formalsprachlichen Präzision zu verzichten, ohne dass dadurch die Fähigkeit zum Verstehen der fachlichen Bedeutung signifikant leidet.

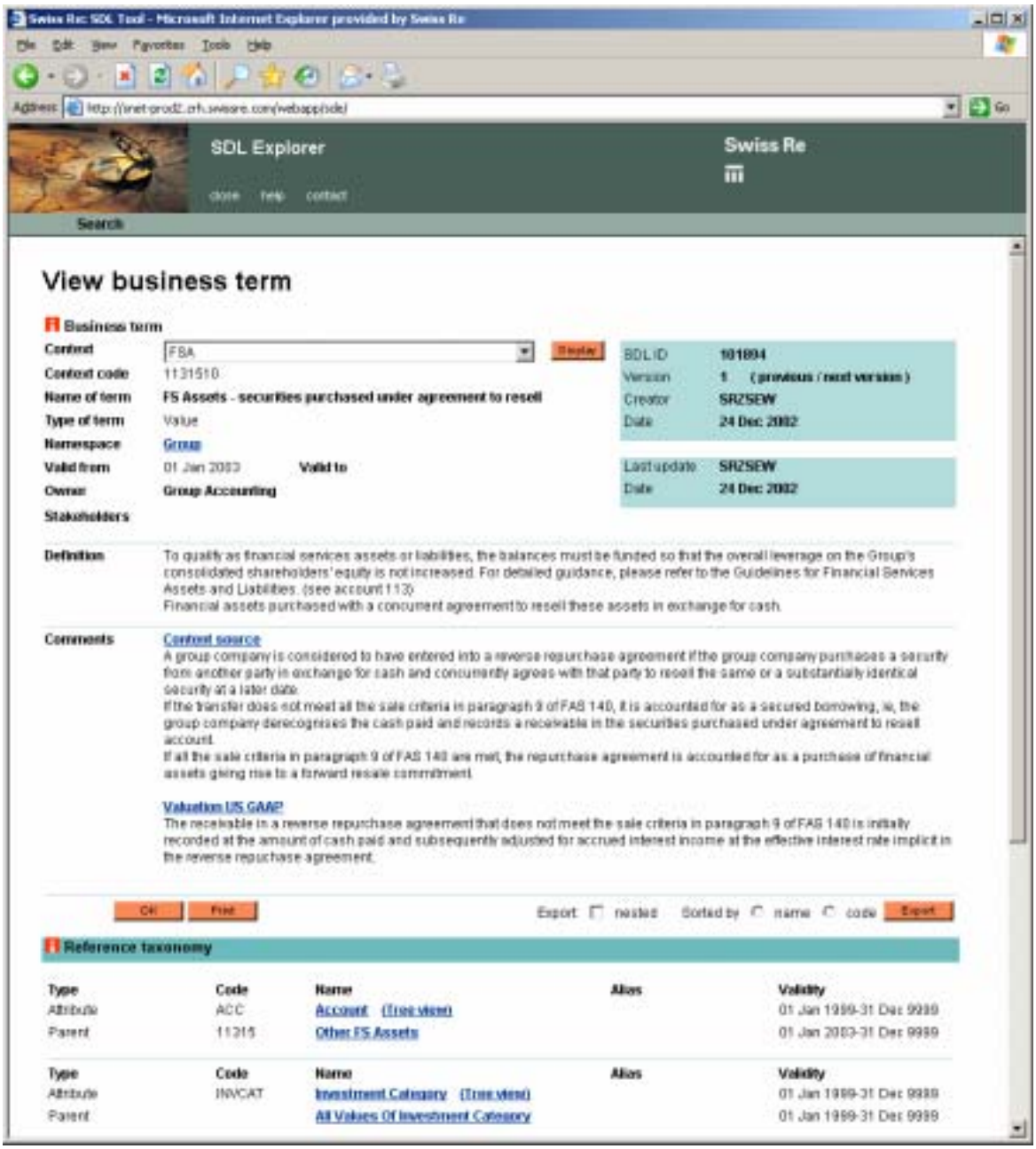


Bild 3: Die kontextsensitiven Aspekte von Fachbegriffen werden in der SDL über die hier abgebildete Sicht angezeigt: Definition, Beschreibung (bei diesem Begriff nicht vorhanden), Kommentare, Verortung in der bzw. den Referenzstruktur(en) und den aus Platzgründen nicht gezeigten Kundenstruktur(en).

5.2 Temporale und versionenbasierte Welt

Die Modellierung eines leistungsfähigen, aber verständlichen Versionskonzepts erwies sich als enorm schwierig. Die Tatsache, dass sich verschiedene Artefakte (Begriffe, Bäume, Kontexte) unabhängig voneinander ändern können und zudem noch die Gültigkeitszeiträume zur Berechnung der aktuell zu verwendenden Version hinzugezogen werden müssen, stellt grosse Anforderungen an die Fachvertreter. Wir mussten diverse Vereinfachungen vornehmen, die im Interesse der Allgemeingültigkeit des Konzepts als bedauernswert zu beurteilen sind. Wir mussten feststellen, dass die Einfachheit des Entwurfs eine wesentliche Rolle für die Kundenakzeptanz spielt, auch wenn die formale Eindeutigkeit darunter leidet.

Heute wird die implizite Erstellung von Konfigurationen (A bezieht sich auf die aktuell gültige Version von B) der expliziten (A bezieht sich auf Version 17 von B) vorgezogen. Seiteneffekte sind so schlechter zu erkennen; gleichfalls leidet die Fähigkeit zur Konsistenzanalyse.

Diverse Gespräche mit Applikationsvertretern zeigten uns, dass zeitlich äquidistante (z.B. quartalsweise) Publikation von Artefakten den Change-Prozess in der IT vereinfacht, aber in den Fachgebieten trotzdem nicht zu Konflikten führt. Teilweise ist es dort (z.B. im Finanzbereich) ohnehin schon aus anderen Gründen gängige Praxis. Diese idiomatische Lösung funktioniert relativ zuverlässig, ohne dass der Sachverhalt vergegenständlicht werden müsste.

Es wurde von uns erwogen, den Entwurf der SDL radikal zu vereinfachen, um die Verständlichkeit zu erhöhen, aus Akzeptanzgründen aber nicht umgesetzt. Gleichfalls wurde erwogen, Verzweigungen der Versionshistorie anzubieten, um von der etwas eigenartigen Modellierung von Varianten in Kundenbäumen wegzukommen. Ebenfalls dachten wir über ein bitemporales Datenmodell nach [Snod1999]. Dies wurde aber sowohl von den Fachabteilungen (aus Komplexitätsgründen) als auch von den Geldgebern (wegen des Projektrisikos) abgelehnt.

6 Zusammenfassung und Ausblick

Schon die verhältnismässig einfache SDL entwickelte zusammen mit dem Versionskonzept eine beachtliche Komplexität und wir gelangten an die Grenzen des Akzeptablen. Die Modellierung einer Fachsprache mit wenigen Ausdrucksmitteln gestaltete sich als der zu erwartende Kompromiss. Wir glauben, dass er uns gut gelungen ist. Insbesondere nutzen wir die menschliche Fähigkeit zum intelligenten Umgang mit Homonymen und erzielen so eine Entlastung der Konzeptwelt der SDL, ohne zu stark an Klarheit einzubüssen. Unsere Erfahrung zeigt auch, dass temporalen Aspekten von Fachbegriffen – vielleicht auch bedingt durch unsere Betonung der ökonomisch-analytischen Welt – grosse praktische Relevanz zukommt.

Als Diskussionsbeitrag stellen wir die Frage, wie weit die “vollständige, widerspruchsfreie und eindeutige Beschreibung“ [Turo2002] von Fachkomponenten auf Basis von Fachterminologie praktisch beherrschbar ist, und inwieweit kontextinsensitive Eindeutigkeit überhaupt nötig ist. Da im Memorandum des GI AK 5.10.3 (unter anderem) auf Wiederverwendung abgezielt wird, möchten wir die Frage aufwerfen, ob allein technische Mittel zur Realisierung von Wiederverwendungspotenzial hinreichend bzw. nötig sind. Insbesondere stellen wir diese Frage, weil

1. es beinahe unmöglich ist, sich Kosten- und Zeitdruck zu entziehen und Altlasten selten zu eliminieren sind,

2. allerhöchste formelle Genauigkeit in einer mit vielen Ambiguitäten behafteten Fach(sprach)welt nicht der entscheidende Erfolgsfaktor (sondern manchmal sogar hinderlich) ist und
3. unsere Erfahrungen mit einer repositorybasierten Lösung die praktischen Verständlichkeits- bzw. Komplexitätsgrenzen deutlich aufzeigen.

Bei Swiss Re konzentrieren wir uns daher insbesondere bei der Wiederverwendung von Entitätstypen verstärkt auf (semiformelle) Ansätze im methodischen Bereich und verwenden dazu (vgl. [Mart2003]):

1. ein architekturelles Rahmenwerk als Richtschnur zur Datenmodellierung,
2. ein enges organisatorisches Netz als Projekthilfe und -korrektiv und
3. technische Massnahmen zur Abbildung existierender Lösungen auf die SDL (vgl. Abschnitt 4).

Wir stehen dabei erst am Anfang, können aber sagen, dass durch die höhere Flexibilität die Akzeptanz trotz geringerer formaler Strenge höher ist; damit, so hoffen wir, wird der Wiederverwendungsgedanke auf fruchtbaren Boden fallen.

7 Literatur

[Turo2002] *Turowski, K.*: Vereinheitlichte Spezifikation von Fachkomponenten. http://www.fachkomponenten.de, Abruf am 2003-05-15.

[Krue1992] *Krueger, C. W.*: In: Software Reuse. ACM Computing Surveys, 24 (1992), S. 132-183. 1992

[Mart2003] *Marti, R.*: Information Integration in a Global Enterprise. Some Experiences from a Financial Services Company. *In: Weikum, G., Schöning, H., Rahm, E. (Hrsg.)*: Tagungsband der BTW 2003, http://www.btw2003.de, Abruf am 2003-05-15.

[Wege2002] *Wegener, H., Auth, G.*: Die Swiss Re Data Language – Erfahrungen mit Terminologiemanagement im Rahmen des Data Warehousing. *In: von Maur, E., Winter, R. (Hrsg.)*: Tagungsband der DW 2002, Heidelberg 2002.

[Snod1999] *Snodgrass, R.*: Developing Time-Oriented Database Applications in SQL. San Francisco 1999.

Entwicklung eines Metamodells für die „Vereinheitlichte Spezifikation von Fachkomponenten“

Peter Fettke, Peter Loos

Johannes Gutenberg-Universität Mainz, Lehrstuhl für Wirtschaftsinformatik und BWL, ISYM - Information Systems & Management, D-55099 Mainz, Germany, Tel.: +49 6131 39-22018, Fax: -22185, E-Mail: {fettke|loos}@isym.bwl.uni-mainz.de, WWW: http://www.isym.bwl.uni-mainz.de

Zusammenfassung. Metamodelle sind allgemein bekannte Instrumente zur Systemgestaltung. In diesem Beitrag wird ein Metamodell für die „Vereinheitlichte Spezifikation von Fachkomponenten“ der Arbeitsgruppe 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik (Memorandum) entwickelt. Das mit Hilfe des Entity-Relationship-Modells formulierte Metamodell enthält weit mehr als 50 Konstrukte (Entitäts- und Beziehungstypen). Im Metamodell können verschiedene Beziehungen zwischen den Spezifikationsebenen des Memorandums aufgezeigt werden. Ebenso ist die Identifikation von Interpretationsspielräumen möglich. In künftigen Arbeiten können das vorgestellte Metamodell verfeinert sowie die unterbreiteten Gestaltungsspielräume zur Weiterentwicklung des Memorandums aufgegriffen werden.

Schlüsselworte: Komponenten, Metaisierung, Komponenten-Repositorium, Lehre, Modellierung

1 Ausgangssituation und Motivation

Die komponentenbasierte Softwareentwicklung verspricht die Effektivität und Effizienz der Systementwicklung zu verbessern [Same97; Szyp02; Grif98; Turo01]. Während notwendige Technologien für eine komponentenbasierte Entwicklung weitgehend verfügbar sind, bspw. Enterprise Java Beans von Sun oder .NET von Microsoft, besteht ein Entwicklungsbedarf bei der Spezifikation von Fachkomponenten. Einen Beitrag zur Überwindung dieser Problematik leistet das von der Arbeitsgruppe 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik entwickelte Memorandum „Vereinheitlichte Spezifikation von Fachkomponenten“ [Acke+02] (siehe Abschnitt 2.1, das Wort „Memorandum“ bezeichnet im Folgenden diesen Spezifikationsvorschlag).

Das Memorandum schlägt eine Spezifikation von Fachkomponenten auf unterschiedlichen Spezifikationsebenen vor, wobei unterschiedliche Notationen für die einzelnen Spezifikationsebenen eingeführt werden (bspw. Interface Definition Language (IDL) der Object Management Group (OMG), Object Constraint Language (OCL), Tabellenform). Auf den ersten Blick lässt die gewählte aspektorientierte Spezifikation einer Fachkomponente vermuten, dass die einzelnen Spezifikationsebenen unabhängig voneinander sind. Indes zeigt eine detailliertere Analyse, dass zwischen den einzelnen Konstrukten unterschiedlicher Spezifikationsebenen Abhängigkeiten bestehen können. Beispiele für Abhängigkeiten sind: 1. Auf der Verhaltensebene können nur diejenigen Dienste spezifiziert werden, die auf der Schnittstellenebene bereits definiert worden sind. 2. Die Dienste, die auf der Schnittstellenebene eingeführt worden sind, werden Aufgaben zugeordnet, die auf der Aufgabenebene definiert werden. 3. Gene-

rell sollen auf der Terminologieebene sämtliche Begriffe definiert werden, die auf den anderen Spezifikationsebenen verwendet werden.

Die genannten Beispiele zeigen, dass die vom Memorandum eingeführten Spezifikationsebenen einer Fachkomponente nicht vollständig unabhängig voneinander sind, sondern gegenseitige Abhängigkeiten untereinander aufweisen. Aus Sicht der Autoren leidet hierdurch einerseits die Verständlichkeit für die einzelnen Spezifikationsebenen einer Fachkomponente. Andererseits, kann dies dazu führen, dass die Spezifikation einer Fachkomponente Inkonsistenzen, Redundanzen bzw. Unvollständigkeiten aufweist, die bei separater Betrachtung einzelner Spezifikationsebenen *nicht* ersichtlich werden, sondern erst bei einer ebenenübergreifenden Analyse aufgedeckt werden können.

Um Klarheit über diese Sachverhalte zu schaffen, halten die Autoren es für sinnvoll, ein einheitliches und integriertes Metamodell für die im Memorandum vorgeschlagenen Notationen zu entwickeln (vgl. die Argumentation von [SüEb97, 1f.] im Kontext der Booch-Methode). In einem Metamodell können die einzelnen Konstrukte der verwendeten Notationen und deren Zusammenhang verdeutlicht werden. Die verschiedenen Notationen sollten in einem Metamodell integriert werden, sodass vorhandene Abhängigkeiten und Beziehungen zwischen den Konstrukten der verschiedenen Notationen expliziert werden.

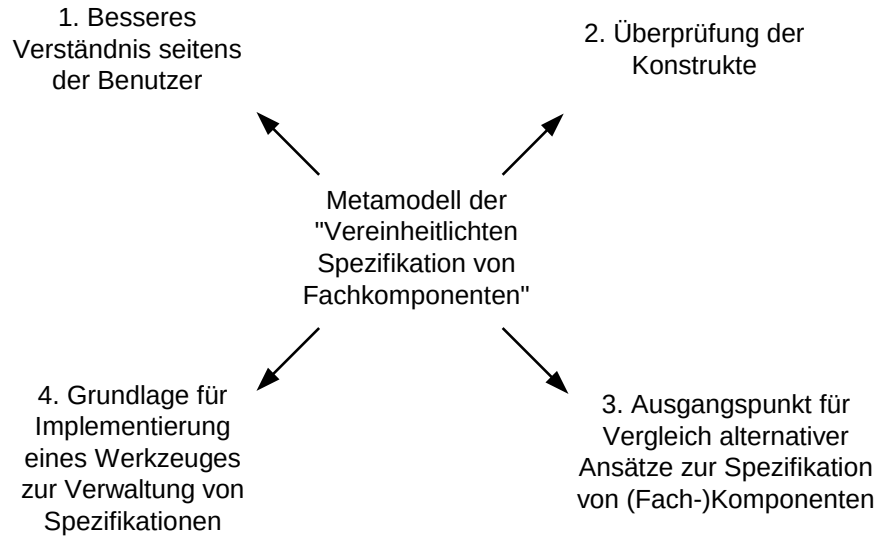


Bild 1: Nutzenpotenziale des Metamodells

Aus einer allgemeinen Perspektive eröffnet ein integriertes Metamodell für die einzelnen Notationen des Memorandums verschiedene Nutzenpotenziale (vgl. Bild 1):

1. Besseres Verständnis für die verschiedenen Konstrukte der Notationen des Memorandums: Ein Metamodell kann die Kommunikation zwischen Nutzern des Memorandums unterstützen, indem sämtliche Konstrukte der Spezifikationsrahmens und ihrer Beziehungen expliziert werden. Hierdurch kann ebenso die Lehrbarkeit des Spezifikationsansatzes in Theorie und Praxis erleichtert werden. Durch Verweis auf bestimmte Konstrukte des Metamodells, kann ferner die Weiterentwicklung des Memorandums und seiner Spezifikationsaspekte systematisch unterstützt werden.
2. Überprüfung der Konstrukte der Notationen des Memorandums: Durch die Erstellung eines Metamodells können Zusammenhänge zwischen verschiedenen Konstrukten der

verschiedenen Notationen aufgedeckt werden. Hierdurch wird es möglich, Konsistenz und Widerspruchsfreiheit zu überprüfen. Ebenso können Abhängigkeiten und Zusammenhänge zwischen Konstrukten verschiedener Ebenen aufgedeckt werden.

3. Ausgangspunkt für Vergleich alternativer Ansätze zur Spezifikation von (Fach-)Komponenten: Neben dem Memorandum existieren alternative Ansätze zur Spezifikation von (Fach-)Komponenten [DSWi98; BJPW99; Fisc00; Han99]. Auf der Grundlage der Metamodelle der verschiedenen Ansätze wird es möglich, Ähnlichkeiten und Gemeinsamkeiten zwischen den Ansätzen systematisch darzustellen und zu untersuchen.
4. Grundlage für die Implementierung eines Werkzeuges zur Verwaltung von Spezifikationen: Ein Metamodell der Notationen des Memorandums kann als ein Ausgangspunkt für die Entwicklung eines Fachkonzeptes eines Werkzeuges zur Verwaltung von Komponentenspezifikationen verwendet werden.

Ziel des Beitrages ist es, ein Metamodell für die im Memorandum definierten Notationen zu entwickeln. Der Beitrag ist folgendermaßen aufgebaut: Nach dieser Einleitung werden im nächsten Abschnitt das Memorandum, der Begriff des Metamodells und verwandte Forschungsarbeiten knapp rekapituliert. Abschnitt 3 stellt das entwickelte Metamodell vor. Im abschließenden vierten Abschnitt werden verschiedene Schlussfolgerungen gezogen und ein Ausblick auf weitere Fragestellungen gegeben.

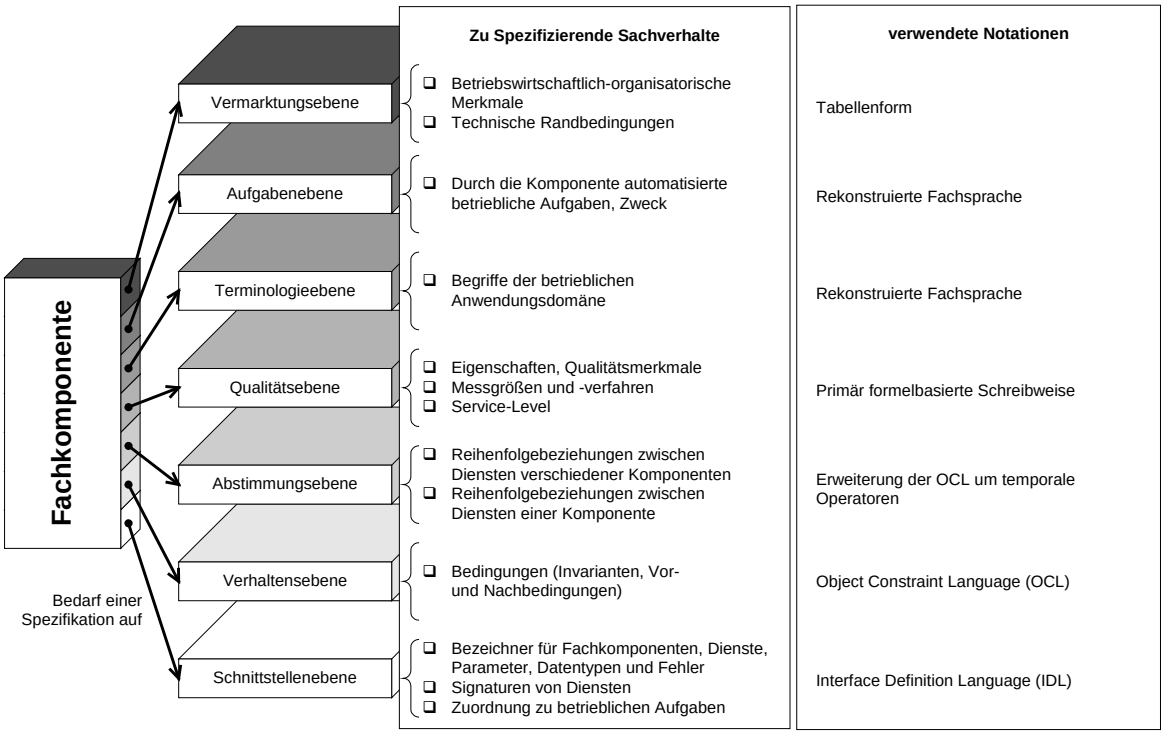


Bild 2: Spezifikationsebenen und –aspekte sowie verwendete Notationen [Acke+02]

2 Theoretischer Hintergrund

2.1 Vereinheitlichte Spezifikation von Fachkomponenten

Das Memorandum verwendet sieben Ebenen zur Spezifikation (Bild 2) [Acke+02]. Jede Ebene fokussiert einen bestimmten Aspekt der Spezifikation einer Fachkomponente und berücksichtigt die Anforderungen einer Rolle im Entwicklungsprozess. Auf den einzelnen Ebenen werden z T. unterschiedliche Notationen verwendet (OCL, OMG IDL etc.). Für eine ausführliche Diskussion der verschiedenen Spezifikationsaspekte sowie der verwendeten Notationen sei auf die zitierte Literatur verwiesen.

2.2 Metamodelle

Der Begriff Metamodell wird in der Literatur nicht einheitlich verwendet. Allgemein kann ein Metamodell als ein Modell eines Modells verstanden werden. Aus Sicht des Metamodells wird das beschriebene Modell Objektmodell genannt. Bei der Metamodellbildung können zwei Metaisierungsprinzipien unterschieden werden [Stra96, S. 17-28]:

- Ein sprachbasiertes Metamodell repräsentiert die bei der Konstruktion des Objektmodells verwendete Modellierungssprache.
- Ein prozessbasiertes Metamodell repräsentiert die bei der Konstruktion des Objektmodells durchgeführten Modellierungsschritte.

Für den hier verfolgten Zweck sind ausschließlich sprachbasierte Metamodelle von Relevanz, sodass das Wort „Metamodell“ ausschließlich in diesem Sinne im Folgenden verwendet wird. Demnach stellt ein Metamodell die in einem Modell verwendeten Konstrukte und deren Beziehungen dar. Diese Beziehungen definieren, welche Konstrukte in welchen Zusammenhängen erlaubt sind. Ein sprachbasiertes Metamodell ist vergleichbar mit der Grammatikdefinition einer Programmiersprache.

2.3 Verwandte Forschungsarbeiten

Allgemein werden Metamodelle in unterschiedlichen Forschungsbereichen verwendet. Anwendungsbeispiele finden sich bei der Evaluierung von objektorientierten Analysemethoden [Stra96], der Modell Driven Architecture [KlWB03], der Auswahl von Workflowmanagementsystemen [Mühl99] oder der Darstellung und dem Vergleich von Ontologien [DGR02].

Ein spezielles Metamodell für das Memorandum ist den Autoren nicht bekannt, obgleich für einzelne Sichten bereits syntaktische Regelwerke existieren, die als Metamodelle verstanden werden können (Bild 3). Indes sind vorhandene Arbeiten bisher nicht in einem Gesamtmodell integriert dargestellt. In der Arbeit von [Turo01] wird auf allgemeiner Ebene der Zusammenhang zwischen den Begriffen Fachkomponente, Dienst, Schnittstelle, betriebliche Aufgabe, betriebliches Anwendungssystem, Implementierung und Eigenschaft mit Hilfe eines UML-Klassendiagramms geklärt. Dieses Modell kann als ein grobes Metamodell verstanden werden, das als Ausgangspunkt für ein detailliertes Metamodell dienen kann.

Spezifikationsebene	Quellen
Vermarktungsebene	[FeLo01]
Aufgabenebene	[Ortn97]
Terminologieebene	[Ortn97]
Qualitätsebene	-
Abstimmungsebene	[CoTu00]
Verhaltensebene	[OMG01b]
Schnittstellenebene	[OMG01a]
alle Ebenen	[Acke+02]

Bild 3: Ausgangspunkte für die Erstellung eines integrierten Metamodells

3 Entwicklung des Metamodells

In diesem Abschnitt wird das Metamodell für das Memorandum entwickelt (Bild 4). Als Modellierungssprache zur Repräsentation des Metamodells wird das Entity-Relationship-Model (ERM) verwendet. Die Auswahl kann dahingehend begründet werden, dass diese Modellierungssprache relativ weit verbreitet und verständlich ist. Es werden im Folgenden die einzelnen Spezifikationsebenen betrachtet, wobei mit der Schnittstellenebene begonnen wird.

Die Schnittstellenebene besteht aus zwei „Interface“-Spezifikationen: Die eine Spezifikation beschreibt die Dienste, die von der Komponente angeboten werden, die andere Spezifikation (interface extern) die Dienste, die von der Komponente nachgefragt werden. Eine „Interface“-Spezifikation wiederum enthält einfache sowie strukturierte Datentypen, Ausnahmezustände sowie verschiedene Dienste. Die Dienste der Schnittstellenebene können einzelnen Aufgaben der Aufgabenebene zugeordnet werden bzw. Datentypen sind mit Begriffen auf der Terminologieebene zu assoziieren.

Auf der Verhaltensebene werden Bedingungen definiert, die das Verhalten einzelner Dienste festlegen. Auf der Ebene können eine Reihe von Bedingungen spezifiziert werden, wobei jede einzelne Bedingung eindeutig einer Komponente zuzuordnen ist. Der Kontext einer Bedingung kann durch die Angabe eines Dienstes präzisiert werden, auf den sich die Bedingung bezieht. Zur Formulierung der Bedingungen können die auf der Schnittstellenebene verwendeten Datentypen verwendet werden. Die angeführten Beziehungen zu Diensten und Datentypen der Schnittstellenebene dürfen sich nur auf die eigene und nicht auf die externe Schnittstelle beziehen. Diese Bedingung ist nicht explizit im Metamodell formuliert.

Die Abstimmungsebene besteht aus einer Reihe von Bedingungen, die sich zwingend auf einen Dienst der Schnittstellenebene beziehen. Neben der Kontextdefinition einer Bedingung, kann eine Bedingung weiterhin Notwendigkeiten zur Abstimmung zu anderen Diensten der Komponente definieren (Intra-Abstimmungsbedingung). Andererseits können Abstimmungen zu Diensten anderer Komponenten beschrieben werden (Inter-Abstimmungsbedingung). Während die Beziehungen „Kontext“ und „Intra-Abstimmung“ zwingend auf Dienste der eigenen Schnittstelle verweisen, bezieht sich die Beziehung „Inter-Abstimmung“ zwingend auf Dienste der externen Schnittstelle (diese Bedingung ist nicht im Metamodell formuliert).

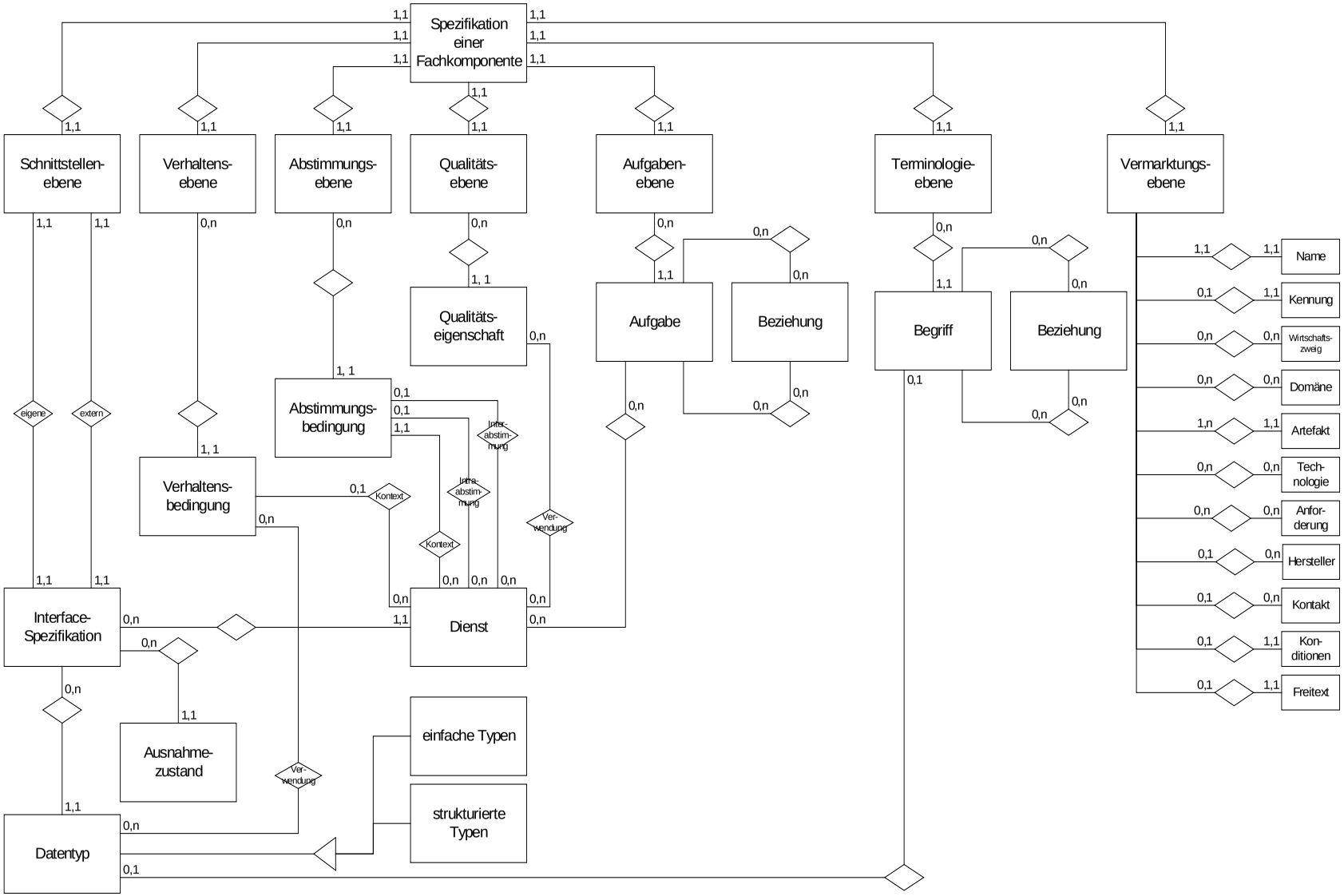


Bild 4: Metamodell für die Notationen des Memorandums

Die zur Zeit im Memorandum beschriebene Verfahrensweise zur Spezifikation der Qualitätsebene enthält nur wenige Hinweise auf erlaubte Spezifikationskonstrukte. Wird einer relativ allgemeinen Auffassung gefolgt, kann die Qualitätsebene als eine Menge von Qualitätseigenschaften verstanden werden. Einzelne Qualitätseigenschaften können hierbei speziellen Diensten einer Komponente zugeordnet werden, die sowohl durch die eigene als auch die externe Schnittstelle definiert werden. Auf diese Weise ist es auch möglich zu spezifizieren, dass die Dienste, die von einer Komponente nachgefragt werden, bestimmten Qualitätseigenschaften zu genügen haben.

Die Metamodellteilbereiche für die Aufgaben- und Terminologieebene sind analog aufgebaut. Es wird jeweils auf eine explizite Darstellung von Satzbauplänen etc. verzichtet, um das entwickelte Metamodell überschaubar zu halten. Während auf der Aufgabenebene eine Reihe von Aufgaben zuzüglich Beziehungen zwischen den Aufgaben definiert werden, sind auf der Terminologieebene Begriffe und Begriffsbeziehungen zu beschreiben. Mögliche Beziehungen zwischen Aufgaben sind bspw. „ist alternative Aufgabe“ oder „ist vorgelagert“. Beziehungen zwischen Begriffen können bspw. sein „ist synonym“, „ist teil-synonym“ und „ist Beispiel für“.

Die einzelnen Spezifikationsattribute der Vermarktungsebene können intuitiv sowohl als ERM-Attribute einer Komponente (bspw. Name, Kennung) als auch als eigenständige Entitätstypen (bspw. Hersteller, Ansprechpartner) verstanden werden. Um eine einheitliche Übersetzung des Memorandums in das Metamodell vorzunehmen und eine – aus Sicht der Autoren – wenig zielführende Diskussion zu vermeiden, ob bestimmte Informationen als „Entitäten“ oder als „Attribute“ von Entitäten zu verstehen sind, wurden sämtliche Spezifikationsattribute dieser Ebene als Entitätstypen im Metamodell repräsentiert. Durch die Festlegung unterschiedlicher Kardinalitäten wird ausgedrückt, ob es sich um optionale oder verpflichtende Angaben handelt bzw. ob einzelne Informationen für verschiedene Komponenten mehrfach verwendet werden können.

4 Schlussfolgerungen und Ausblick

In diesem Beitrag wurde ein relativ grobes Metamodell des Memorandums vorgestellt, das trotzdem mehr als 50 Konstrukte (Entitäts- und Beziehungstypen) enthält. Aus dem Metamodell wird ersichtlich, dass einzelne Konstrukte einer Komponentenspezifikation eine zentrale Bedeutung besitzen. Beispielsweise werden Dienste der Schnittstellenebene auf der Verhaltens-, Abstimmungs-, Aufgaben- und evtl. der Qualitätsebene – zum Teil sogar mehrfach – referenziert. Anderen Konstrukten kommt dagegen nur eine geringere Bedeutung zu. Im Memorandum wird bspw. ausgeführt, dass auf der Terminologieebene die Bedeutung von Begriffen zu klären ist, die auf sämtlichen Ebenen verwendet werden. Allerdings ist aus dem Metamodell ersichtlich, dass nur explizite Beziehungen zwischen Datentypen und Begriffen bestehen. Folglich gibt das Memorandum nur wenige Gestaltungsrestriktionen bzw. -hinweise, welche Begriffe auf der Terminologieebene zwingend zu erläutern sind.

Im Rahmen der Untersuchung sind verschiedene Grenzen deutlich geworden, die im Folgenden angesprochen werden:

- Subjektivität der Metamodellbildung: Das Bilden von Metamodellkonstrukten ist nicht immer eindeutig. Bestimmte Sachverhalte können aus unterschiedlichen Perspektiven unterschiedlich dargestellt werden. Beispielsweise ist aus dem Memorandum nicht ersichtlich, ob zu einer Aufgabe auf der Aufgabenebene zwingend ein

Dienst auf der Spezifikationsebene angegeben werden muss oder ob dieser optional ist. Dieser Sachverhalte sollte in einer künftigen Memorandumsversion geklärt werden.

- Detaillierung der Abbildung: Es ist nicht immer offensichtlich, wie detailliert das Metamodell zu erstellen ist. Beispielweise wurde im vorgestellten Metamodell ein Dienst auf der Schnittstellenebene durch jeweils ein Entitätstyp repräsentiert. Ebenso wäre es denkbar, das Metamodell weiter zu detaillieren, indem darüber hinaus die Signaturen eines Dienstes im Metamodell repräsentiert werden.
- Verwendete Metamodellierungssprachen: Als Metamodellierungssprache wird das ERM verwendet. Es stellt sich die Frage, ob die Verwendung anderer Metamodellierungssprachen zu Vorteilen führt, da diese evtl. eine höhere Ausdrucksmächtigkeit besitzen. Beispielsweise konnten in der verwendeten Modellierungssprache einzelne Randbedingungen nicht im Metamodell, sondern nur natürlichsprachlich formuliert.
- Im vorgestellten Metamodell wurde ausschließlich die primäre Notation berücksichtigt, ohne auf Eigenarten der sekundären Notation einzugehen. Es ist zu überlegen, inwieweit die sekundäre Notation Erweiterungen und Ergänzungen am vorgestellten Metamodell notwendig werden lässt. Beispielsweise wäre es denkbar, für sämtliche Wörter einer natürlichsprachlichen Spezifikation – wie sie bspw. für die Abstimmungsebene vorgeschlagen wird – zu fordern, dass sie ebenso auf der Terminologieebene definiert werden. Andererseits kann der Standpunkt vertreten werden, dass bei Verwendung der sekundären Notation generell nicht auf eine hohe Konsistenz zwischen den Abstimmungsebenen zu achten ist und daher sekundäre Notationen prinzipiell nicht im Metamodell zu berücksichtigen sind.

Die Autoren schlagen einerseits vor, die bei der Untersuchung aufgedeckten Probleme bei der Überarbeitung und Weiterentwicklung des Memorandums zu berücksichtigen. Anderseits kann das hier eingeführte Metamodell weiter gepflegt und Bestandteil einer künftigen Memorandumsversion werden, sodass die einleitend angeführten Nutzenpotenziale weitgehend realisiert werden können.

Literatur

[Acke+02] *Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Kotlar, O.; Loos, P.; Mrech, H.; Raape, U.; Ortner, E.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T.; Turowski, K.* Vereinheitlichte Spezifikation von Fachkomponenten - Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme. http://wi2.wiwi.uni-augsburg.de/gi-memorandum.php.htm, Abruf 2003-03-01.

[BJPW99] *Beugnard, A.; Jézéquel, J.-M.; Plouzeau, N.; Watkins, D.*: Making Components Contract Aware. In: IEEE Computer 32 (1999) 7, S. 38-45.

[CoTu00] *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 - Beiträge des Workshops "Modellierung 2000", St. Goar, 5.-7. April 2000. Koblenz 2000, S. 179-194.

[DSWi98] *D'Souza, D. F.; Wills, A. C.*: Objects, Components, and Frameworks with UML - The Catalysis Approach. Reading, MA, et al. 1998.

[DGR02] *Davies, I.; Green, P.; Rosemann, M.*: Facilitating an Ontological Foundation of Information Systems with Meta Models. In: *A. Wenn; M. McGrath; F. Burstein (Hrsg.)*: Proceedings of the 13th Australasian Conference on Information Systems (ACIS 2002), 3-6 December. Melbourne 2002, S. 937-947.

[FeLo01] *Fettke, P.; Loos, P.*: Ein Vorschlag zur Spezifikation von Fachkomponenten auf der Administrations-Ebene. In: *K. Turowski (Hrsg.)*: Modellierung und Spezifikation von Fachkomponenten: 2. Workshop im Rahmen der vertIS (verteilte Informationssysteme auf der Grundlage von Objekten, Komponenten und Agenten) 2001, Bamberg, Deutschland, 05. Oktober 2001. Bamberg 2001, S. 95-104.

[Fisc00] *Fischer, B.*: Specification-Based Browsing of Software Component Libraries. In: Journal of Automated Software Engineering 7 (2000) 2, S. 179-200.

[Grif98] *Griffel, F.*: Componentware - Konzepte und Techniken eines Softwareparadigmas. Heidelberg 1998.

[Han99] *Han, J.*: An Approach to Software Component Specification. Proceedings of 1999 International Workshop on Component Based Software Engineering. Los Angeles, USA 1999.

[KlWB03] *Kleppe, A.; Warmer, J.; Bast, W.*: MDA Explained: The Model Driven Architecture: Practice and Promise. Boston et al. 2003.

[Mühl99] *Mühlen zur, M.*: Evaluation of Workflow Management Systems Using Meta Models. Proceedings of the 32th Hawaii International Conference on Systems Science (HICSS '99). Hawaii 1999.

[OMG01a] *OMG:* The Common Object Request Broker: Architecture and Specification: Version 2.5. Framingham 2001.

[OMG01b] *OMG:* Unified Modeling Language Specification: Version 1.4. Needham 2001.

[Ortn97] *Ortner, E.*: Methodenneutraler Fachentwurf - Zu den Grundlagen einer anwendungsorientierten Informatik. Stuttgart, Leipzig 1997.

[Same97] *Sametinger, J.*: Software Engineering with Reusable Components. Berlin et al. 1997.

[Stra96] *Strahringer, S.*: Metamodellierung als Instrument des Methodenvergleichs - Eine Evaluierung am Beispiel objektorientierter Analysemethoden. Aachen 1996.

[SüEb97] *Süttenbach, R.; Ebert, J.:* A Booch Metamodel. Fachberichte Informatik 5/97. Koblenz 1997.

[Szyp02] *Szyperski, C.*: Component Software - Beyond Object-Oriented Programming. 2. Aufl., London et al. 2002.

[Turo01] *Turowski, K.*: Fachkomponenten - Komponentenbasierte betriebliche Anwendungssysteme. Habil.-Schr. Magdeburg 2001.

Spezifikation von Fachkomponenten mit der UML 2.0

Jörg Ackermann

Universität Augsburg, Universitätsstraße 16, 86135 Augsburg, Deutschland, E-Mail: joerg.ackermann.hd@t-online.de

Zusammenfassung. Die UML 2.0 bietet (im Vergleich zur UML 1.4) bessere Möglichkeiten zur Modellierung von Komponenten. Dieser Beitrag stellt zunächst die relevanten UML 2.0 Konzepte vor. Danach wird diskutiert, welche Auswirkungen sich daraus auf das Memorandum „Vereinheitlichte Spezifikation von Fachkomponenten“ ergeben und wie das Memorandum entsprechend weiterentwickelt werden könnte.

Schlüsselworte: Fachkomponente; (Software-)Komponente; Formale Spezifikation; UML 2.0

Die präzise und geeignete Spezifikation von Fachkomponenten ist eine wesentliche Voraussetzung, damit sich der Bau von Anwendungssystemen aus am Markt gehandelten Softwarekomponenten etablieren kann. Wichtig ist dabei vor allem die Standardisierung der Komponentenspezifikationen, damit diese von allen beteiligten Parteien richtig verstanden werden. Die Arbeitsgruppe 5.10.3. der Gesellschaft für Informatik hat einen Vorschlag erstellt, wie Aufbau und Inhalt einer Spezifikation aussehen sollten. Für weitere Details siehe das Memorandum „Vereinheitlichte Spezifikation von Fachkomponenten“ [Turo2002].

Das Memorandum verwendet auf der Verhaltens- und der Abstimmungsebene Elemente der Unified Modeling Language (UML), insbesondere die Object Constraint Language (OCL). Grundlage für das Memorandum war die damals verfügbare Version UML 1.4 [OMG2001]. Mittlerweile wurde die Version UML 2.0 [OMG2003] entwickelt, die kurz vor der Verabschiedung steht. UML 2.0 bietet deutlich bessere Möglichkeiten zur Modellierung und Spezifikation von Komponenten. Diese werden in diesem Beitrag vorgestellt und es wird diskutiert, welche Möglichkeiten sich daraus für das Memorandum ergeben.

Im Kapitel 1 werden die Komponentenbegriffe in der UML und im Memorandum vorgestellt und miteinander verglichen. Im Kapitel 2 werden die für Komponenten relevanten Konzepte von UML 2.0 vorgestellt. Im Kapitel 3 wird diskutiert, wie die UML 2.0 in einer weiterentwickelten Fassung des Memorandums berücksichtigt werden könnte. Abschließend stellt das Kapitel 4 noch zwei Neuerungen in der OCL vor und zeigt, wie diese die Spezifikation von Fachkomponenten vereinfachen.

1 Komponentenbegriffe in der UML und im Memorandum

In diesem Abschnitt wollen wir zunächst die Komponentenbegriffe in der UML und im Memorandum vorstellen und diskutieren.

In der UML 2.0 wird eine Komponente folgendermaßen definiert [OMG2003, S. G-574]:

> A component is a “modular part of a system that encapsulates its contents and whose

> manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. …"

Im Vergleich dazu betrachten wir die Definition aus dem Memorandum [Turo2002, S. 1]:

> „Eine *Komponente* besteht aus verschiedenartigen (Software-) Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist."

Wir stellen dabei fest, dass beide Definitionen in wesentlichen Punkten übereinstimmen:

- Abgeschlossenheit (in UML bezeichnet als „modular part"),
- Kapselung (im Memorandum bezeichnet als „verbirgt ihre Realisierung"),
- Verwendung wohldefinierter Schnittstellen.

Die Unterschiede in den Definitionen ergeben sich (nach Meinung des Autors) vor allem aus der unterschiedlichen Sichtweise:

- UML betrachtet Komponenten als Teile von Systemen und definiert daher Komponenten in Bezug auf Systeme (Teil eines Systems, austauschbar).
- Das Memorandum fokussiert mehr auf die Komponenten an sich (wiederverwendbar, kombinierbar) und bringt zusätzlich betriebswirtschaftliche Aspekte (vermarktbar) ein.

Da beide Definitionen auf technischer Ebene weitgehend übereinstimmen, können nach Meinung des Autors UML-Komponenten zur Modellierung und Spezifikation von Komponenten gemäß des Memorandums eingesetzt werden.

Bei der Modellierung von Komponenten ist es sinnvoll, semantisch zwischen drei verschiedenen Arten von Modellen zu unterscheiden: (software-unabhängige) konzeptionelle Modelle, Spezifikationsmodelle und Implementierungsmodelle (vergleiche [Fowl1999]). Diese Modellarten fokussieren auf unterschiedlichen Aspekten bei der Beschreibung einer Komponente und ihres Umfelds.

Die UML unterscheidet zwischen logischen Komponenten (z.B. Businesskomponenten, Prozesskomponenten) und physischen Komponenten (z.B. EJB-Komponente, CORBA-Komponente, COM+-Komponente, .NET-Komponente). Leider wird diese Unterscheidung nicht näher erläutert. Aber anhand der angegebenen Beispiele kann man davon ausgehen, dass in konzeptionellen Modellen und in Spezifikationsmodellen logische Komponenten verwendet werden, während physische Komponenten in Implementierungsmodellen zu finden sind.

Im Memorandum erfolgt explizit keine weitere Unterscheidung von Komponenten wie in UML. Bei der Spezifikation von Fachkomponenten steht allerdings die Außensicht der Komponente und nicht ihre Implementierung im Vordergrund. Beim Einsatz von Modellen im Memorandum sollte es sich daher um Spezifikationsmodelle handeln und es sollten logische Komponenten verwendet werden.

2 Komponenten in der UML 2.0

Schon die UML 1.4 enthielt das Modellierungskonstrukt *Komponente*. Dieses war jedoch nur

für die Modellierung von physischen Komponenten vorgesehen. Für die Modellierung von logischen Komponenten enthielt die UML 1.4 keine explizite Unterstützung, weshalb verschiedene Autoren unterschiedliche Modellierungskonstrukte zur Beschreibung logischer Komponenten heranzogen. So wurden z.B. Subsysteme [Andr2003, S. 27] oder Klassen mit dem Stereotyp «comp spec» [ChDa2001] verwendet.

Die bedeutendste Änderung in der UML 2.0 besteht darin, dass auch logische Komponenten mit dem Konstrukt *Komponente* modelliert werden können. Dadurch ergeben sich deutlich bessere Möglichkeiten zur verständlichen und konsistenten Spezifikation von Komponenten.

Die wichtigsten Konstrukte und deren Verwendung werden nachfolgend dargestellt:

- Eine Komponente wird als ein Rechteck mit dem Schlüsselwort «component» dargestellt.
- Eine Komponente ist im UML Metamodell ein Subtyp von Klasse. Eine Komponente kann daher Attribute und Methoden haben. Außerdem kann eine Komponente optional eine interne Struktur haben und eine Menge von Ports zur Verfügung stellen.

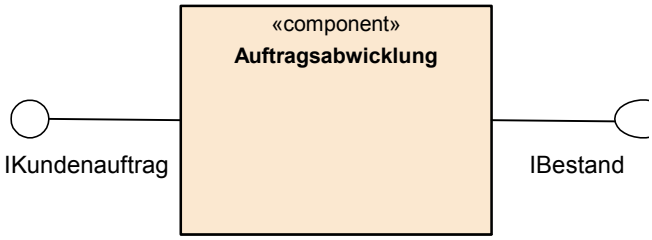


Bild 1: Komponente mit Schnittstellen

- Die externe Sicht einer Komponente (Black-Box-Sicht) besteht aus der Komponente und ihren Schnittstellen. Im Bild 1 sieht man die Darstellung einer Komponente *Auftragsabwicklung* mit den Schnittstellen *IKundenauftrag* und *IBestand*.
- Eine Komponente kann mehrere Schnittstellen haben. Dabei wird zwischen bereitgestellten und benötigten Schnittstellen unterschieden. Im Bild 1 werden exemplarisch eine bereitgestellte (*IKundenauftrag*) und eine benötigte Schnittstelle (*IBestand*) dargestellt.

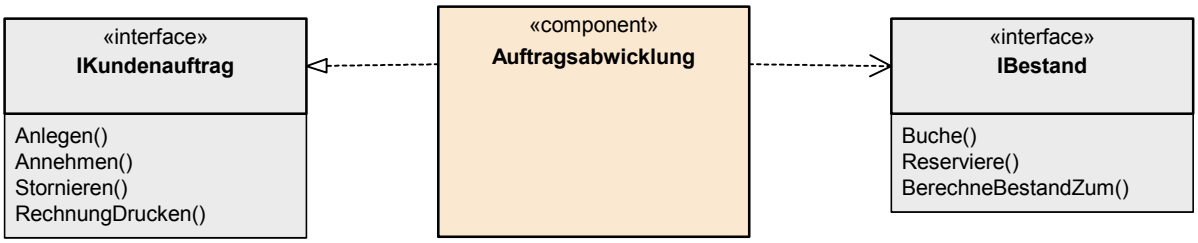


Bild 2: Detaillierte Darstellung der Komponenten-Schnittstellen

- Sollen die Schnittstellen detaillierter dargestellt werden, kann man dafür das Konstrukt

Schnittstelle («interface») verwenden. Die Schnittstelle kann inklusive ihrer Operationen (mit Parametern) und eventuell vorhandener Attribute modelliert werden. So sieht man im Bild 2, dass die Schnittstelle *IKundenauftrag* die Operationen *Anlegen*, *Annehmen*, *Stornieren* und *RechnungDrucken* zur Verfügung stellt (auf die Angabe der Parameter wurde an dieser Stelle verzichtet).

- Optional kann eine Komponente mit Ports versehen werden. Ein Port ist eine Schnittstelle mit eigenem Bezeichner. Mit Ports kann man z.B. verschiedene Zugangspunkte zu einer Komponente unterscheiden, die technisch dieselbe Schnittstelle verwenden.
- Optional kann ein Zustandsdiagramm mit einer Schnittstelle (oder einem Port) verbunden werden, welches dynamische Bedingungen (Aufrufreihenfolge von Operationen) expliziert.
- Von der internen Sicht einer Komponente (White-Box-Sicht) wird gesprochen, wenn die interne Struktur und die Realisierung einer Komponente dargestellt werden.

3 Erweiterungsvorschläge für das Memorandum

Um die Spezifikation von Fachkomponenten anhand des Memorandums einem möglichst breiten Personenkreis zugänglich zu machen, werden im Memorandum bewusst allgemein bekannte Notationstechniken vorgeschlagen. Deshalb werden im Memorandum auf der Verhaltens- und der Abstimmungsebene Elemente der UML verwendet.

Die Spezifikation und Modellierung von Komponenten ist derzeit ein wichtiges Forschungsthema und wird daher von vielen Autoren untersucht. Entsprechend gab es in den letzten 2-3 Jahren einiges an neuen Entwicklungen und Erkenntnissen. Eines der Standardwerke zur Modellierung von Komponenten mit UML ist [ChDa2001]. Viele Ansätze daraus sind in die Version UML 2.0 eingeflossen.

Das Memorandum sollte die UML-Version 2.0 berücksichtigen und Gebrauch von ihren Erweiterungen machen. Der Hauptgrund für diesen Vorschlag liegt darin, dass das Memorandum nur so weiterhin seinem Anspruch gerecht werden kann, weit verbreitete und bekannte Notationstechniken zu verwenden. Daneben ergeben sich auch einige Verbesserungen und Vereinfachungen bei der Spezifikation von Fachkomponenten.

Auf dieser Zielsetzung aufbauend unterbreiten wir die folgenden Vorschläge zur Weiterentwicklung des Memorandum. Alle beziehen sich jeweils auf die primäre Notationstechnik.

1. Allgemein:
   - Eine Komponenten-Spezifikation sollte ein Diagramm enthalten, welches die externe Sicht der Komponente darstellt. Dieses zeigt neben der Komponente auch die bereitgestellten und benötigten Schnittstellen. Vergleiche dazu Bild 1. Dieses Diagramm gilt für die gesamte Spezifikation und wird keiner Ebene zugeordnet.
   - Dienste, welche die Komponente von anderen Komponenten benötigt, werden als benötigte Schnittstellen modelliert. Die Bezeichnung und die Spezifikation solcher benötigter Schnittstellen erfolgt aus Sicht der Komponente selbst. (Man beachte, dass verschiedene andere Komponenten die benötigten Dienste zur Verfügung stellen können. Außerdem wird zum Zeitpunkt der Spezifikation im allgemeinen nicht bekannt

sein, welche konkrete Komponente dies überhaupt sein wird. Damit ist es nicht sinnvoll, bei den benötigten Diensten auf konkrete Schnittstellen konkreter Komponenten zu verweisen.) Anhand des Diagramms kann man erkennen, welche Interfaces eine Komponente zur Verfügung stellt und welche von ihr benötigt werden. Daher kann auf die Namenskonvention *Extern* verzichtet werden, die im Memorandum bisher als Workaround verwendet wurde.

- Eine Komponente kann in UML mehrere Interfaces haben. Daher sollte sich nicht, wie im Memorandum als Beispiel zur Schnittstellenebene angegeben, der Name der Fachkomponente aus dem Namen eines Interfaces ergeben.

2. Schnittstellenebene:
   - Hier sollte ebenfalls die UML zum Einsatz kommen. In einem speziellen Diagramm (vergleiche Bild 2 für eine verkürzte Darstellung) werden die Schnittstellen der Komponente detailliert beschrieben. Das Diagramm enthält für alle Schnittstellen die zugehörigen Dienste (Operationen) mit ihren Parametern und Ausnahmen.
   - Bisher gab es durch die Verwendung der OMG IDL auf der Schnittstellenebene und der UML OCL auf der Verhaltensebene einen Methodenbruch [Acke2001], der u.a. dazu führte, dass die Spezifikationsobjekte der verschiedenen Ebenen nicht eindeutig einander zugeordnet werden konnten. Dieses Problem wurde bisher durch Namenskonventionen umgangen. Durch die Verwendung der UML auf der Schnittstellenebene würde dieses Problem behoben. Außerdem könnte auf eine Notationstechnik verzichtet werden.
   - Falls weiterhin eine Schnittstellenbeschreibung mit der OMG IDL benötigt wird, sollte untersucht werden, ob diese aus der UML-Schnittstellenbeschreibung abgeleitet werden kann. Dann könnte man, wenn benötigt, die Beschreibung in OMG IDL automatisch generieren.
   - Durch diesen Vorschlag würde auf der Schnittstellen-, der Verhaltens- und der Abstimmungsebene einheitlich die UML (zusammen mit der OCL) verwendet. Dies passt auch sehr gut zum Vorschlag von Overhage, einen thematischen Bereich *Technik* mit diesen drei Ebenen zu bilden. [Over2002]

3. Verhaltensebene
   - Manche Komponenten verwalten Geschäftsdaten, die im Zusammenhang mit der Funktionalität und den Diensten der Komponente stehen. So könnte z.B. eine Komponente *Auftragsabwicklung* alle bisher erfassten Kundenaufträge speichern und verwalten. Die Gesamtheit der verwalteten Daten zu einem Zeitpunkt wird oft auch als interner State der Komponente bezeichnet. Siehe dazu z.B. [ChDa2001, S. 123ff.]. Verwaltet die Komponente Geschäftsdaten, so muss auf der Verhaltensebene die Abhängigkeit der Dienste vom internen State (und umgekehrt) spezifiziert werden. So kann ein Dienst *IKundenauftrag.Stornieren* nur Aufträge stornieren, die der Komponente bekannt sind. Oder ein Dienst *IKundenauftrag.Anlegen* wird, wenn erfolgreich ausgeführt, einen neuen Auftrag in der Komponente anlegen.
   - In der Fallstudie [Acke2001] wurde vorgeschlagen, ein UML-Klassendiagramm zur konzeptionellen Beschreibung des internen States zu verwenden. Dies wurde im Memorandum aufgegriffen, in dem es optional ein UML-Klassendiagramm als konzeptionelles Schema vorsieht, welches u.a. zur Abbildung des internen States der

Komponente verwendet werden kann.

- Mit der UML 2.0 kann das konzeptionelle Schema passender als interne Sicht der Komponente modelliert werden. Die zur Beschreibung des States notwendigen Entitäten werden als Klassen mit dem Standard-Stereotyp «type» dargestellt. Dieses Modell dient als eine Art Interface Information Model (vergleiche z.B. [ChDa2001]) und dient als Grundlage, um Zusicherungen mit der OCL zu formulieren. Im Bild 3 ist ein konzeptionelles Schema für die Komponente *Auftragsabwicklung* abgebildet. Dieses enthält die Entitäten *Kundenauftrag* und *Kunde*, deren Eigenschaften (Attribute und Assoziationen) in OCL-Ausdrücken verwendet werden können.

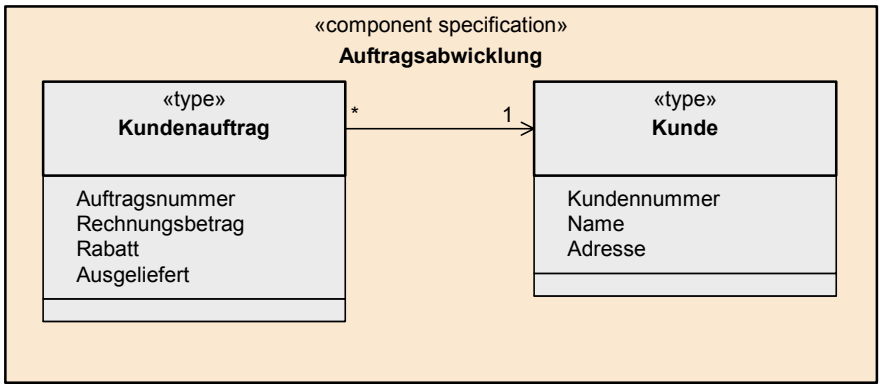


Bild 3: Komponente mit spezifikationsrelevanten internen Typen

4. Abstimmungsebene

- Die UML schlägt vor, Zustandsmaschinen als Standard zu verwenden, um Sachverhalte auf der Abstimmungsebene zu beschreiben. Demgegenüber stehen die vom Memorandum vorgeschlagenen temporalen Operatoren.
- Es wird vorgeschlagen, beide Varianten als Alternativen in das Memorandum aufzunehmen, da beide ihre spezifischen Vorteile haben. Dadurch wird die Konformität zur UML gewahrt, ohne die Ausdrucksmächtigkeit zu verringern.

Durch die unterbreiteten Vorschläge ergeben sich die folgenden Vorteile:

- Schnittstellen-, Verhaltens- und Abstimmungsebene verwenden mit der UML eine einheitliche Notationstechnik.
- Die Spezifikation auf diesen Ebenen profitiert von den erweiterten Ausdrucksmöglichkeiten von UML 2.0.
- Die Spezifikation bleibt konform zum Modellierungsstandard UML.
- Auf einige der bisher notwendigen Workarounds (gedachte Komponente Extern, Namenskonventionen zur Verbindung von Schnittstellen und Verhaltensebene) kann verzichtet werden.

Mit diesen Vorschlägen wird also dafür plädiert, die UML konsequent auf den Ebenen einzusetzen, wo sie die Spezifikation verbessern kann und wo sie international als Modellierungsstandard anerkannt ist. Das sind derzeit die Schnittstellen-, Verhaltens- und

(mit Einschränkungen) die Abstimmungsebene. Das Memorandum leistet über diese Ebenen hinaus einen wichtigen Beitrag, in dem es alle spezifikationsrelevanten Objekte identifiziert und den Spezifikationsebenen zuordnet. Das Memorandum betrachtet dabei auch Objekte (z.B. Terminologie und betriebliche Aufgaben), die von der UML nicht abgedeckt werden.

4 Verbesserungen bei der OCL 2.0

In den Fallstudien [Acke2001] und [Acke2003] sind in einigen Details Probleme bei der Verwendung der Object Constraint Language (OCL) aufgetreten. Die Version UML 2.0 enthält einige Erweiterungen, die für die Spezifikation von Fachkomponenten relevant sind und zwei der identifizierten Probleme beheben.

Es soll spezifiziert werden, dass während der Abarbeitung eines Dienstes ein anderer Dienst aufgerufen wird, bei dem es sich um eine Nicht-Query-Methode handelt. Dies war bisher mit OCL nicht möglich. Im Memorandum wurden solche Bedingungen auf der Abstimmungsebene formuliert und für die Abstimmungsebene wurde (genau wegen dieses Problems) postuliert, dass (im Gegensatz zum OCL-Standard) auch Nicht-Query-Methoden in OCL-Ausdrücken verwendet werden können. Vergleiche dazu [Acke2001, S. 55]. Die OCL 2.0 bietet nun einen Operator *hasSent* ('^'), mit welchem genau dieser Sachverhalt ausgedrückt werden kann.

Außerdem ist es mit OCL 2.0 möglich, auf die Exportparameter eines Dienstes auch im Kontext eines anderen Dienstes zuzugreifen. Dazu wurde der übliche Zugriff auf Eigenschaften (durch Verwendung von '.') auch auf die Exportparameter einer Methode erweitert. Da ein solcher Zugriff in den Fallstudien notwendig war, wurde diese Möglichkeit dort einfach vorweggenommen (allerdings durch Verwendung von '::'). Vergleiche dazu [Acke2001, S. 39].

Die neuen Möglichkeiten mit der OCL sollen nun an einem Beispiel demonstriert werden. Betrachten wir eine Beispielkomponente *Auftragsverwaltung*. Diese hat zwei Interfaces: das bereitgestellte Interface *IKundenauftrag* (u.a. mit der Methode *Annehmen*) und das benötigte Interface *IBestand* (u.a. mit der Methode *Buche*). Vergleiche dazu auch Bild 2. Der Dienst *Annehmen* soll folgende Aufgaben ausführen:

- Überprüfen, ob die Auftragsdaten konsistent sind und ob der Auftrag ausgeführt werden kann,
- Ermitteln, ob die erforderlichen Materialien verfügbar sind (bei einer anderen Komponente über die Methode *IBestand.BerechneBestandZum*),
- Buchen der erforderlichen Materialien (bei einer anderen Komponente über die Methode *IBestand.Buche*),
- Zurückmelden, ob der Auftrag angenommen werden kann.

Der Kundenauftrag kann nicht angenommen werden, wenn die erforderlichen Materialien nicht verfügbar sind. Dies wird in der Nachbedingung in Bild 4 ausgedrückt. Der verfügbare Bestand wird über die Methode *IBestand.BerechneBestandZum()* ermittelt und von dieser Methode im Exportparameter *Bestand* zurückgegeben. In der dritten Zeile von Bild 4 sieht man, wie mit Hilfe des OCL-Ausdrucks `BerechneBestandZum().Bestand` auf den Exportparameter *Bestand* zugegriffen werden kann, um die Werte danach mit dem benötigten Bestand vergleichen zu können.

```
Auftragsabwicklung::IKundenauftrag::Annehmen(in at: Auftrag): boolean
 post: let benötigterBestand = ... in
       let verfügbarerBestand = IBestand.BerechneBestandZum(...).Bestand in
       if benötigtesMaterial > verfügbaresMaterial
         then result = false endif
```

Bild 4: OCL-Bedingung mit Zugriff auf einen Exportparameter

Ist das benötigte Material vorhanden, wird von der Methode *Annehmen* die Methode *Buche* von *IBestand* gerufen. Dieser Sachverhalt wird in der Nachbedingung im Bild 5 mit Hilfe des OCL-Ausdrucks `IBestand^Buche(...)` ausgedrückt.

```
Auftragsabwicklung::IKundenauftrag::Annehmen(in at: Auftrag): boolean
 post: if benötigtesMaterial <= verfügbaresMaterial
         then IBestand^Buche(...) endif
```

Bild 5: OCL-Bedingung mit dem Operator hasSent (‘^’)

Literatur

[Acke2001] *Ackermann, J.*: Fallstudie zur Spezifikation von Fachkomponenten. In: *K. Turowski (Hrsg.)*: 2. Workshop Modellierung und Spezifikation von Fachkomponenten., Bamberg 2001, S. 1 – 66.

[Acke2003] *Ackermann, J.*: Zur Spezifikation der Parameter von Fachkomponenten. In: *K. Turowski (Hrsg.)*: 5. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 5). Augsburg 2003, S. 47 – 154.

[Andr2003] *Andresen, A.:* Komponentenbasierte Softwareentwicklung mit MDA, UML und XML. Hanser Verlag, München 2003.

[ChDa2001] *Cheesman, J.; Daniels, J.:* UML Components. Addison-Wesley, Boston 2001.

[Fowl2000] *Fowler, M.:* UML Distilled – A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2000.

[OMG2001] *OMG (Hrsg.):* OMG Unified Modeling Language Specification, Version 1.4, September 2001. Needham 2001.

[OMG2003] *OMG (Hrsg.):* Unified Modeling Language: Superstructure, Version 2.0, Stand 10.04.2003. URL: http://www.omg.org/uml, Abruf am 2003-06-10.

[Over2002] *Overhage, S.:* Die Spezifikation – kritischer Erfolgsfaktor der Komponentenorientierung. In: *K. Turowski (Hrsg.)*: 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 4). Augsburg 2002, S. 1 – 17.

[Turo2002] *Turowski, K. (Hrsg.).:* Vereinheitlichte Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 der Gesellschaft für Informatik. Augsburg 2002.

Erfahrungen im Umgang mit der Spezifikation von Web Services

Andreas Schmietendorf[*#], Reiner Dumke[*], Daniel Reitz[*#]

* *Otto-von-Guericke-Universität Magdeburg, Fakultät Informatik, IVS, Universitätsplatz 2, D-39016 Magdeburg, Email: reitz|schmiete|dumke@ivs.cs.uni-magdeburg.de*

\# *T-Systems Nova, Entwicklungszentrum Berlin, E21/STE, Wittestr. 30H, D-13509 Berlin, Email: andreas.schmietendorf|daniel.reitz@t-systems.com*

Zusammenfassung: Der vorliegende Artikel analysiert die derzeitige Vorgehensweise bei der Spezifikation von Web Services und vergleicht diese mit dem Vorschlag des GI-Arbeitskreises 5.10.3 in Bezug auf Fachkomponenten. Nach einer einführenden Darstellung der Web Service Technologie wird zuerst einmal auf potentielle Unterschiede bzw. Gemeinsamkeiten von Fachkomponenten und Web Services eingegangen. Unter Berücksichtigung dieses Vergleichs und des Memorandum „Vereinheitlichte Spezifikation von Fachkomponenten“ werden mehr als 120 im Internet verfügbare Web Services einer Bewertung hinsichtlich ihrer Spezifikation unterzogen. Ziel dieser empirischen Analyse ist es die derzeitige Reife der Spezifikationen von Web Services zu bewerten, zumanderen soll ein Diskussion angeregt werden, inwieweit das Memorandum auch auf den Bereich der Web Services übertragen werden kann.

Schlüsselworte: Web Services, Fachkomponenten, Empirische Bewertung, Spezifikation, Spezifikationsebenen, Metriken

1 Einführung

Noch existiert keine einheitliche Auffassung, was genau unter einem Web Service zu verstehen ist. Sehr allgemein kann von einem softwarebasierten Service gesprochen werden, der über das weltweit verfügbare Internet angeboten wird. Web Services kombinieren dabei die Vorteile der komponentenbasierten Entwicklung, der Internet-Technologie und im zunehmenden Maße auch der Agententechnologie, dabei werden folgenden Vorteile erwartet:

- Hoher Grad an Standardisierung und breite Akzeptanz innerhalb der Industrie
- Synchrones und asynchrones Kommunikationsmodell
- Unterstützung des Komponenten- und Integrationsparadigma
- Einfache Vorgehensweise zur Kommunikation durch Firewalls hindurch
- Brücke zwischen heterogenen Technologieansätzen

Bei Web Services handelt es sich um netzwerkbasierte Applikationen, die ihre im Internet angebotenen Funktionen mittels des WSDL-Protokolls (Web-Service Description Language) beschreiben, für den Informationsaustausch XML-Dokumente (eXtensible Markup Language) verwenden und für den Aufruf entfernter Methoden bzw. die Datenübertragung das SOAP-Protokoll (Simple Object Access Protocol) nutzen. Typischerweise erfolgt die Übertragung der in XML-Dokumente verpackten Daten und Funktionsaufrufe auf der Basis des http-Protokolls, so dass auch durch Firewalls hindurch kommuniziert werden kann. Diese Eigen-

schaft bietet die Möglichkeit echte B2B[1]-Anwendungen zu entwickeln. Zur Lokalisierung im Internet verfügbarer Web Services werden UDDI-Verzeichnisdienste (Universal Description, Discovery and Integration) verwendet.

2 Vergleich von Fachkomponenten und Web Services

Innerhalb dieses Abschnitts wollen wir Fachkomponenten und Web Services miteinander vergleichen. Im Gegensatz zu IV-technischen Komponenten bei deren Implementierung vielfältige technische Problemstellungen durch die Entwicklung zu lösen sind, handelt es sich bei Fachkomponenten um anwendungsnahe Komponenten die eine Konzentration der Entwicklung auf die fachliche Problemstellung erlauben sollen. In Anlehnung an [Memorandum 2002] und unter Berücksichtigung der in [Frank 2001] getätigten Aussagen, kann eine FK folgendermaßen charakterisiert werden:

> Eine Fachkomponente ist eine anwendungsnahe Komponente, die eine bestimmte Menge von Diensten einer betrieblichen Anwendungsdomäne über eine wohl definierte Schnittstellen anbietet und dabei das Kriterium der Abgeschlossenheit (d.h. Kapselung der internen Struktur) erfüllt.

Demgegenüber kann ein Web Service in Anlehnung an [Gemeni 2002] folgendermaßen charakterisiert werden:

> Unter Verwendung der Extensible Markup Language (XML) bieten Web-Services eine standardisierte Vorgehensweise zur synchronen, als auch asynchronen Kommunikation zwischen im Internet verteilten Anwendungen. Während die bisherigen Middlewareansätze immer eine technologieabhängige Vorgehensweise implizierten, bieten Web-Serives eine technologieunabhängige und damit lose Kopplung von Anwendungssystemen. Ein Web-Service kann dabei fachlich begründete Funktionen im Sinn einer Fachkomponente beschreiben, anbieten und zentral verfügbare Verzeichnisse registrieren.

Der im Rahmen des GI-Arbeitskreises 5.10.3 entwickelte Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten kann aus Sicht der Autoren auch zur Spezifikation von Web Services herangezogen werden. Ziel dieses Vorschlags ist zum einem die Darstellung der benötigten Informationen um Fachkomponenten im Rahmen der eigenen Anwendung einsetzen zu können, zum anderen sollen mögliche Vorgehensweisen zur Beschreibung (Notationen und Modelle) des Komponentenverhalten aufgezeigt werden. An dieser Stelle wollen wir auf die detaillierte Darstellung der Spezifikationsebenen verzichten und verweisen auf das unter www.fachkomponenten.de erhältliche Memorandum.

Die Spezifikationsebenen wurden primär zur Unterstützung einer auf der Basis von Komponenten durchzuführenden Entwicklung definiert. D.h. im Rahmen der Entwicklung einer neuen Anwendung soll die Verwendung am Markt gehandelter Komponenten (aus Black Box Sicht) unterstützt werden. Web Services implizieren ebenfalls diese Zielstellung, gehen aber noch einen deutlichen Schritt weiter. Während eingekaufte Fachkomponenten auch physisch im Rahmen der Entwicklung verwendet werden, residieren Web Services innerhalb des Internet und können bei Verwendung im Rahmen der eigenen Anwendung auch dort verbleiben. D.h. eine Anwendung kann auf der Basis im Netzwerk verteilter Fachkomponenten, diese

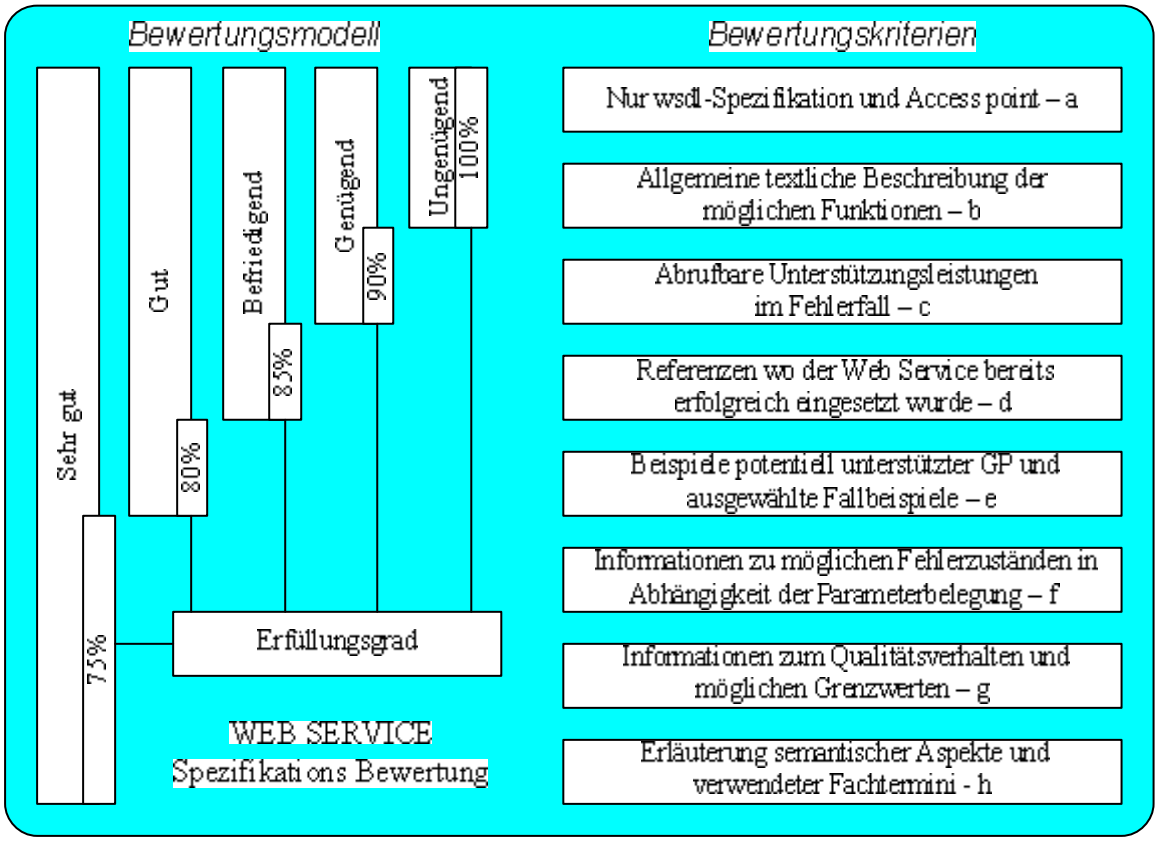
[1] B2B Business to Business

basieren jetzt auf der Web Service-Technologie, realisiert werden. In diesem Sinne verändert sich die Anwendungsentwicklung von der Komposition lokal verfügbarer Fachkomponenten hin zur Integration im Netz verteilter Web Services.

Im Bezug auf das Memorandum des GI-Arbeitskreises 5.10.3 berücksichtigen die aktuellen Kernstandards zu Web-Services im Rahmen der WSDL-Beschreibung lediglich die *Schnittstellen-* und zum Teil die *Verhaltensebene*. Die weiteren Ebenen, welche sich auf die Geschäftsprozessintergration (*Abstimmungsebene*), auf qualitative (*Qualitätsebene)*, semantische (*Terminologie- und Aufgabenebene*) und wirtschaftliche (*Vermarktungsebene)* Aspekte beziehen, werden durch derzeitige Standardisierungsansätze noch nicht explizit erfaßt.

3 Bewertung der Spezifikation von Web Services

Der erfolgreiche Einsatz eines konkreten Web Service innerhalb eines betrieblichen Anwendungssystems hängt maßgeblich von der Güte der angebotenen Spezifikation (Beschreibung) des Web Service ab. Diese Beschreibung muß zumindest Aussagen über die angebotene Funktionalität unter Berücksichtigung der wsdl-Spezifikation bereithalten und darüber hinaus den Ort (typischerweise die URL) wo der Web Service im Internet erreichbar ist wiedergeben. Um die derzeitige Vorgehensweise bei der Spezifikation von Web Service zu erfassen, wurden in Anlehnung an die Spezifikationsebenen des Memorandum, das folgende in der Abbildung 1 dargestellte Bewertungsmodell entwickelt.



Durch Anwendung der GQM-Methode (Goal – Question – Metric) entsprechend [Solingen 1999] wurden die folgenden „Top Level“ Bewertungskriterien identifiziert. Darüber hinaus erfolgte eine Zuordnung dieser Bewertungskriterien zu den Spezifikationsebenen des Komponentenmemorandum.

- Spezifikation des Web Service enthält Aussagen zur wsdl-Beschreibung und den entsprechenden Zugriffspunkt im Internet – **a**. (vgl. Schnittstellenebene)

- Spezifikation des Web Service enthält eine grundlegende Beschreibung der angebotenen Funktionen - **b**. (vgl. Aufgabenebene und Verhaltensebene)

- Spezifikation des Web Service enthält Informationen zu ggf. notwendigen Unterstützungsleistungen im Fehlerfall. - **c**. (vgl. Vermarktungsebene)

- Spezifikation des Web Service enthält Referenzen zu Anwendungen bei denen der Web Service bereits erfolgreich eingesetzt wird. - **d**. (vgl. Vermarktungsebene)

- Spezifikation des Web Service enthält Informationen zu potentiell unterstützen Geschäftsprozessen und ausgewählten Fallbeispielen - **e**. (vgl. Abstimmungsebene)

- Spezifikation des Web Service enthält Informationen zu möglichen Fehlerzuständen in Abhängigkeit der Parameterbelegung angebotener Methoden - **f**. (vgl. Verhaltensebene)

- Spezifikation des Web Service enthält Informationen zum Qualitätsverhalten und ggf. möglichen Grenzwerten - **g**. (vgl. Qualitätsebene)

- Spezifikation des Web Service enthält Informationen zu den genutzten Fachtermini - **h**. (vgl. Terminologieebene und Aufgabenebene)

- Spezifikation des Web Service enthält Aussagen zum Lizenzmodell und die Vorgehensweise zur Verrechnung der Nutzung (vgl. Vermarktungsebene) - **i**.

Zur Bewertung der Web Service Spezifikationen wurde ein einfaches, in Anlehnung an die Schulnoten entwickeltes Modell genutzt. Um eine möglichst genaue Ermittlung der Bewertung zu unterstützen, erfolgte eine Detaillierung der Bewertungskriterien durch Bilden von Subkriterien die erfüllt sein müssen um eine bestimmt Bewertung zu erreichen.

- *ungenügend* – Die Spezifikation berücksichtigt nur die Aspekte a und b. Die Verwendung eines derart beschriebenen Web Service innerhalb industrieller Anwendungen ist nicht zu empfehlen.

- *genügend* – Die Spezifikation berücksichtigt die Aspekte a, b und c. Die Verwendung eines derart beschriebenen Web Service innerhalb industrieller Anwendungen ist möglich, impliziert aber noch erhebliche Risiken.

- *befriedigend* – Die Spezifikation berücksichtigt die Aspekte a, b, c und d. Die Verwendung eines derart beschriebenen Web Service innerhalb industrieller Anwendungen ist möglich und kann umfangreichen Tests unterzogen werden.

– *gut* – Die Spezifikation berücksichtigt die Aspekte a, b, c, d, e und f. Die Verwendung eines derart beschriebenen Web Service innerhalb industrieller Anwendungen kann empfohlen werden, da bereits umfangreiche Erfahrungen zum Verhalten vorliegen.

– *sehr gut* – Die Spezifikation berücksichtigt alle Beschreibungsaspekte. Auf dieser Grundlage können genutzte Leistungen von Web Service verrechnet werden und sogenannte Service Level Agreements (SLA) abgeschlossen werden.

Darüber hinaus wurde ein Erfüllungsgrad eingeführt, der die folgende Berechnung bei der Ermittlung einer konkreten Bewertung zugrunde legt:

- genügend – 100% für a, b und 90% für c
- befriedigend - 100% für a, b und 90% für c sowie 85% für d
- gut - 100% für a, b, 90% für c und 85% für d sowie 80% für e
- sehr gut - 100% für a, b, 90% für c, 85% für d und 80% für e sowie 75% für f, g, h

4 Anwendung des Bewertungsmodells

Mittels des entwickelten Bewertungsmodells wurden bisher über 120 Web Services einer entsprechenden Analyse unterzogen. Die Auswahl der analysierten Web Services erfolgte nach dem Zufallsprinzip, wobei das Web Service Verzeichnis www.xmethods.com und die dort derzeit verzeichneten ca. 350 Web Services (Stand: Juli 2003) zugrunde gelegt wurden. Dabei ergab sich die in Abbildung 1 dargestellte prozentuale Verteilung über alle bewerteten Web Services hinweg.

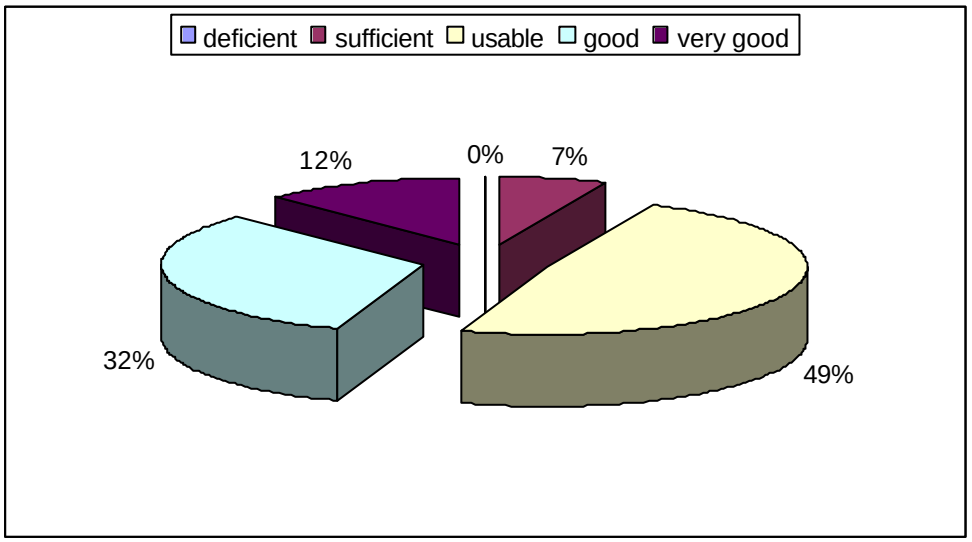


Abbildung 2: Bewertung der Spezifikation von Web Service

Das auch aus Sicht der Autoren überraschend positive Ergebnis hinsichtlich des Reifegrades heute verfügbarer Spezifikationen bzw. Beschreibungen zu Web Services bedarf noch einiger relativierender Aussagen.

- Zumeist werden bei der Spezifikation keine formalen Notationen verwendet.

- Derzeit angebotene Web Service besitzen häufig nur geringen Funktionsumfang.
- Explizite Darstellungen zur Terminologieebene finden sich nur selten.
- Vermarktungs- und Aufgabenebene verschmelzen zumeist.
- Abstimmungsebene wird zumeist nur implizit durch ein Anwendungsbeispiel erfaßt.
- Web Service wurden explizit für die Wiederverwendung entwickelt.
- Detailtiefe der Spezifikationsebenen wurde nur bedingt bewertet.
- Viele Informationen sind im Internet durch Hyperlinks referenziert
- Die angebotenen Web Services tragen nur teilweise einen kommerzielle Charakter
- Einige Web Services sind mit verschiedenen Versionen abgelegt

Darüber hinaus ist anzumerken, dass derzeit keine Aussage getroffen werden kann, inwieweit eine umfangreiche Spezifikation tatsächlich zum breiten Einsatz eines speziellen Web Service beiträgt. Hier sollte ein Web Service Verzeichnis auch eine Referenz zu Projekten enthalten wo der entsprechende Web Service bereits erfolgreich eingesetzt wird. Im Falle des hier verwendeten Web Service Verzeichnis findet sich ein solcher Hinweis nur zum Teil.

5 Schlußfolgerungen für das Memorandum

Aus den durchgeführten Untersuchungen lassen sich einige Vorschläge für potentielle Verbesserungen des Memorandum ableiten. So zeigt sich bei Web Services das Vermarktungs- und Aufgabenebene typischerweise nicht getrennt betrachtet werden. Ebenso wird die Abstimmungsebene durch konkrete Einsatzbeispiele unterstützt. Als aus Sicht der Autoren negativ zu bewerten ist die geringe Verwendung formaler Notationen, wie z.B. der Unified Modelling Language (kurz UML) im Rahmen der Spezifikation von Web Services. Ebenso kritisch ist die, je UDDI-Verzeichnisdienst, unterschiedliche Beschreibungsstruktur enthaltender Web Services anzumerken. Aus Sicht der Autoren kann zur Harmonisierung der unterschiedlichen Spezifikationsansätze das Memorandum (in einer überarbeiteten und auf den Bereich der Web Services erweiterten Version) sinnvoll eingesetzt werden.

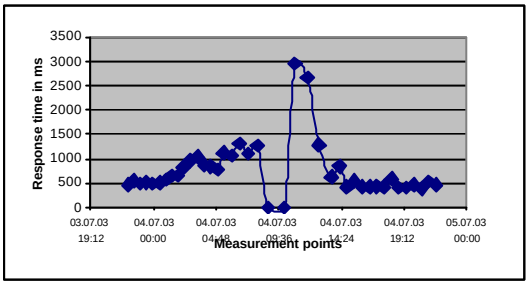
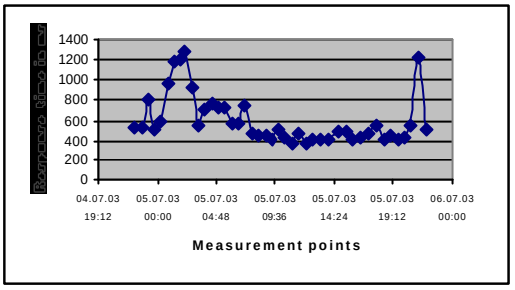
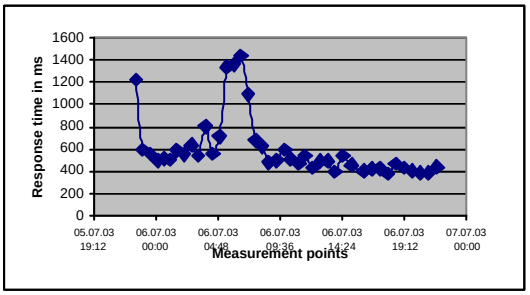
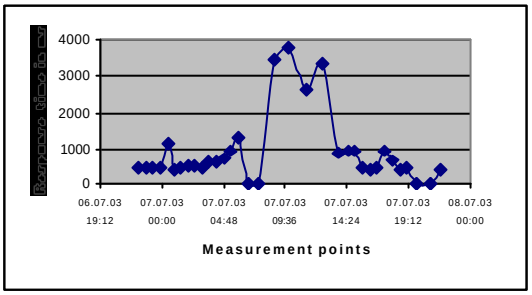
Die signifikanteste Veränderung aus Sicht der Autoren bezieht sich auf die Qualitätsebene. Bedingt durch die Ausführung des Web Service innerhalb des Internet bietet sich theoretisch die Möglichkeit einer Überwachung der Qualitätseigenschaften des Web Service durch einen unabhängigen Meßdienst an, wie unter konzeptionell [Schmietendorf 2003] dargestellt. Auf der Basis eines solchen unabhängigen Meßdienst (allg. wollen wir diesen als Measurement Service bezeichnen) lassen sich Aussagen/Erfahrungen zur Effizienz, Zuverlässigkeit und Wartbarkeit verhältnismäßig einfach gewinnen und im Rahmen der Spezifikation von Web Services ebenfalls webbasiert potentiellen Interessenten anbieten. Dabei können z.B. folgende Qualitätseigenschaften sukzessive erfaßt werden:

- Betrachtung des Leistungsverhalten (Antwortzeit und Durchsatz)
- Betrachtung der Verfügbarkeit der angebotenen Dienste
- Änderungshäufigkeit der angebotenen Dienste. (z.B. neue Versionen)
- Zeitliches Auslastungsverhalten (z.B. Wochenverlauf)

- Darstellung einer Prioritäten gesteuerten Performance-Verhalten

Exemplarisch zeigt Tabelle 1 die erreichten Antwortzeiten des Web Services „CarRentalQuotesService“. Dieser Web Service dient zur Ermittlung der besten Angebote weltweit agierender Autovermietungen, eine umfassende Beschreibung zu diesem Web Service findet sich in der Anlage zu diesem Artikel. Die Ermittlung der Performance und Verfügbarkeitscharakteristik im Tagesverlauf und gleichzeitige Bereitstellung im Rahmen der Spezifikation des Web Service ermöglicht einem potentiellen Interessenten die Bemessung der eigenen Integrationsarchitektur bei Verwendung dieses Web Service. Insbesondere entfällt durch den Einsatz eines unabhängigen „Measurement Service“ das Problem der Referenzumgebung (vgl. [Memorandum 2002] S.13), da hier Software- und Runtime-Umgebung erfasst werden bzw. der Anbieter eines Web Service die Leistungseigenschaften sicherstellen muss (sofern entsprechende SLA’s festgelegt wurden).

Tabelle 1: Performance und Verfügbarkeit „CarRentalQuotesService“

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Messungen am Freitag</td><td>Messungen am Sonnabend</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Messungen am Sonntag</td><td>Messungen am Montag</td></tr>
</table>

Im speziellen Fall des „CarRentalQuotesService“ konnten z.B. die folgenden Erfahrungen über einen Zeitraum von 1 Woche gewonnen werden.

- Antwortzeitverhalten schwankt zwischen minimal 0.3 Sek. bzw. maximal 3.75 Sek.
- Die Verfügbarkeit im Verlauf der Woche lag bei 96,4%
- Am Wochenende verbessert sich das Antwortzeitverhalten (0,4 bis 1,4 Sekunden)
- Größte Last während der Zeit von 09:00 Uhr bis 14:00 Uhr entsprechend GMT
- Vermessen wurden die komplexeste Funktion des Web Service
- Web Service wird bisher eher im europäischen Raum genutzt

6 Zusammenfassung

Die hier dargestellten Untersuchungen wurden insbesondere durch Erfahrungen bei der Anwendung des Memorandums im industriellen als auch akademischen Umfeld motiviert. Immer wieder wurde sowohl im Rahmen industrieller Projekte der Telekommunikationsbranche, als auch innerhalb der Ausbildung die Kritik eines zu umfassenden Spezfikationsansatzes geäußert. Die Schwerpunkte der Kritik lagen bei folgenden Punkten:

- zu viele verwendete Notationen
- nicht umsetzbar in der Praxis, da vom Aufwand her nicht zu vertreten
- keine Entwicklungsumgebung die diesen Ansatz unterstützt
- der Mehrwert einer entsprechenden Spezifikation kommt dem Entwickler nicht zugute
- keine Möglichkeit die Güte der Spezifikation zu bewerten

Für die Autoren dieses Beitrags lag hier der Ansatzpunkt bereits erfolgreich eingesetzte Beschreibungs- und Spezfikationsansätze im Umfeld von Web Services einer empirischen Analyse zu unterziehen. Ziel war es, im Sinne einer „best practice" Empfehlung, aus diesen Ansätzen zu lernen und so zu einer Verbesserung des Spezifikationsansatzes für Fachkomponenten beizutragen. Im Rahmen dieser Untersuchungen können insbesondere die folgenden beiden Aspekte zur Optimierung des Momorandum vorgeschlagen werden:

– Etablieren eine zum Memorandum korrespondierenden Bewertungsmodelle – im Sinne eine „Buisiness Component Maturity Model – BCMM"

– Konzeptionelle Entwicklung einer unabhängigen Instanz zur Zertifizierung der angebotenen Fachkomponenten durch einen unabhängigen „dritten". (vgl. Trust Center)

Für die weitere Vervollkommnung des Memorandum bedarf es aus Sicht der Autoren unbedingt einer Validierung des Spezifikationsansatzes im Sinne einer korrekten Identifizierung der tatsächlich benötigten Informationen für die erfolgreiche Integration Fachkomponenten bzw. Web Services. Der Schluss, dass eine Fachkomponente bzw. ein Web Services umso besser spezifiziert seien, je umfangreicher die dabei zur Verfügung gestellt Informationen sind, ist derzeit nicht bewiesen.

7 Quellenverzeichnis

[Frank 2001] Frank, U.; Jung, J.: Prototypische Vorgehensweise für den Entwurf anwendungsnaher Komponenten S. 57-74, in Proc. zur Verbundtagung VertIS 2001, Universität Bamberg, 2001

[Gemeni 2002] Der Markt für Web-Services - Erwartungen, Treiber, Investitionsabsichten, Cap Gemeni Ernst & Young, Juni 2002

[Memorandum 2002] Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Klein, U.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T.; Turowski, T.: Vereinheitlichte Spezifikation

von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 - Komponentenorientierte betriebliche Anwendungssysteme, Februar 2002

[Schmietendorf 2001] Schmietendorf, A.; Dumke, R.: Spezifikation von Softwarekomponenten auf Qualitätsebene, in Turowski, K.: 2. Workshop Modellierung und Spezifikation von Fachkomponenten, Bamberg/Deutschland, Oktober 2001

[Schmietendorf 2003] Schmietendorf, A.; Dumke, R.; Wipprecht, M.: SLA Management - Herausforderungen im Rahmen von Web Service basierten Infrastrukturen, in Proc. 16. cecmg-Jahrestagung & 6. EuroCMG Conference (only CD-ROM), Hannover/Deutschland, April 2003

[Solingen 1999] Solingen, v. R.; Berghout, E.: The Goal Question Metric (GQM) Method, McGraw Hill, 1999

Anlage – Beispiel der Spezifikation “CarRentalQuotesService”

Im Folgenden soll die Orginal-Spezifikation des Web Service “CarRentalQuotesService” als ein ausgewähltes Beispiel wiedergegeben wurde. Diese wurde unverändert vom Web Service Verzeichnis www.xmethods.com übernommen. Bei diesem Web Service handelt es sich um einen Dienst der die Angebote verschiedener Autovermietungsfirmen miteinander vergleicht. Dafür werden die im Internet publizieren Angebote aktuell miteinander verglichen und so der aktuell günstigste Anbieter identifiziert. Leider bietet dieser Service nicht die Möglichkeit einer direkten Bestellung, dafür muss auf bewährte Systeme umgestiegen werden.

Bewertete wurde dieser Web Service entsprechend des unter Abschnitt 3 eingeführten Bewertungsmodells mit „gut“. Dabei wurden allerdings nicht nur die im Folgenden dargestellten Informationen bewertet, sondern auch die unter http://www.dsdata.co.uk/WS/ verfügbaren Informationen bzw. das dort abrufbare Beispiel zum Einsatz des Web Service. Die im Rahmen des Artikels dargestellten Messungen zur Verfügbarkeit bzw. Performance beziehen sich ebenfalls auf diesen Web Service.

Detailed Description

This service takes the pain out of comparing car rental quotes from the major rental companies. You submit your criteria, such as dates/times and locations and the service locates the best current deals on the Internet returning a list of quotes ordered by price. The service actually visits each car rental site, in parallel, automatically to find the best deal at that moment. Note that this is different to most car rental search engines out there that rely on a database for quotes which may respond quickly but are often misleading and have out-of-date results. The user is provided with completely independent quotes and left to book the car rentals as he/she sees fit.

Usage Notes

The main interface is "getQuotes()" which gets the actual car rental quotes. To feed in the appropriate codes for countries, locations, results-currencies and car types you can optionally call getCountries(), getLocations(), getCurrencies() and getCarTypes(). These are handy for building a fully-functional client.

More information, including a C# demonstration client is available for free download from http://www.dsdata.co.uk/WS/.

To use the service from the web you can try http://www.searchbookgo.com/ which uses these web services at the back end as well. The service is totally independent and free to use. Please feedback or comments to info@dsdata.co.uk.

RPC Profile for Service "Real Time Car Rental Quotes"

As a convenience for those who need to manually configure basic SOAP RPC calls, we provide this page which summarizes all the necessary parameters need to configure a SOAP RPC call. This information is derived automatically from the service WSDL file. It is a subset of what can be found in the more comprehensive WSDL Analyzer available from the service detail page.

Method Name	getQuotes	
Endpoint URL	http://wavendon.dsdata.co.uk/axis/services/CarRentalQuotes	
SOAPAction		
Method Namespace URI	urn:SBGCarRentalQuotes.sbg.travel.ws.dsdata.co.uk	
Input Parameters	**carType**	string
	country	string
	currency	string
	pickupDate	dateTime
	pickupLocation	string
	returnDate	dateTime
	returnLocation	string
	clientID	string
	userEmail	string
Output Parameters	**getQuotesReturn**	ArrayOf_tns2_CarRentalQuoteResponse

Method Name	getCountries	
Endpoint URL	http://wavendon.dsdata.co.uk/axis/services/CarRentalQuotes	
SOAPAction		
Method Namespace URI	urn:SBGCarRentalQuotes.sbg.travel.ws.dsdata.co.uk	
Input Parameters		
Output Parameters	**getCountriesReturn**	ArrayOf_xsd_string

Method Name	getLocations	
Endpoint URL	http://wavendon.dsdata.co.uk/axis/services/CarRentalQuotes	
SOAPAction		
Method Namespace URI	urn:SBGCarRentalQuotes.sbg.travel.ws.dsdata.co.uk	
Input Parameters	**country**	string
Output Parameters	**getLocationsReturn**	ArrayOf_xsd_string

Method Name	getCurrencies	
Endpoint URL	http://wavendon.dsdata.co.uk/axis /services/CarRentalQuotes	
SOAPAction		
Method Namespace URI	urn:SBGCarRentalQuotes.sbg.travel.ws.dsdata.co.uk	
Input Parameters		
Output Parameters	**getCurrenciesReturn**	ArrayOf_xsd_string

Method Name	getCarTypes	
Endpoint URL	http://wavendon.dsdata.co.uk/axis/services/CarRentalQuotes	
SOAPAction		
Method Namespace URI	urn:SBGCarRentalQuotes.sbg.travel.ws.dsdata.co.uk	
Input Parameters		
Output Parameters	**getCarTypesReturn**	ArrayOf_xsd_string

Herausgeber:
Prof. Dr. Klaus Turowski
Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II
Universität Augsburg
Universitätsstraße 16, 86135 Augsburg
Phone: +49(821)598-4431; Fax : -4432
E-Mail: klaus.turowski@wiwi.uni-augsburg.de
URL: http://wi2.wiwi.uni-augsburg.de

ISSN 1619-8980