Enterprise Java Beans
Software- oder/und Fachkomponenten?

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

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

\# *T-Systems Nova, Entwicklungszentrum Berlin, Kompetenzgruppe für System- und Technologieentwicklung, Wittestr. 30H, D-13509 Berlin, Email: andreas.schmietendorf@t-systems.com*

Zusammenfassung: Der vorliegende Artikel analysiert die Technologie der Enterprise-Java-Bean-Komponenten (kurz EJB's) auf der Basis zweier unterschiedlicher Bewertungsansätze. In einem ersten Schritt werden die im Rahmen des GI-Arbeitskreises 5.10.3 identifizierten Spezifikationsebenen einer Fachkomponente auf ihre Anwendbarkeit im Rahmen EJB-basierter Komponenten untersucht. In einem zweiten Schritt erfolgt die metrikenbasierte Analyse einer auf der Basis mehrerer EJB-Komponenten implementierten Komponente. Als Grundlage für deren Architektur wurde das Session-Facade-Muster verwendet. Die Zielstellung dieses Beitrags besteht primär darin, eine Diskussion zur Themenstellung EJB-basierter Fachkomponenten anzuregen, potentielle Bewertungskriterien aufzuzeigen und für weitere Arbeiten auf diesen Themengebiet zu motivieren.

Schlüsselworte: Enterprise Java Beans, Fachkomponenten, Softwarekomponenten, Spezifikation, Spezifikationsebenen, Metriken

1 Einführung

Mit den Enterprise Java Beans (kurz EJB) bietet sich erstmals ein serverseitiges Komponentenmodell, das den Anspruch erhebt, dem Entwickler eine primär fachliche Sicht bei der Implementierung von Komponenten zu ermöglichen. Auf dieser Grundlage sollte theoretisch entsprechend der folgenden Definition aus [Fachkomponenten 2002] die Entwicklung von sogenannten Fachkomponenten möglich sein:

> *Eine Fachkomponente ist eine Komponente, die eine bestimmte Menge von Diensten einer betrieblichen Anwendungsdomaine anbietet.*

Nach einem kurzen Überblick zur EJB-Technologie im folgenden Abschnitt soll innerhalb dieses Artikels der Frage nachgegangen werden, inwieweit die EJB-basierten Softwarekomponenten die Möglichkeit für die Entwicklung von Fachkomponenten bieten bzw. welche potentiellen Probleme bei der Verwendung entstehen können. Anhand des im Rahmen der Gesellschaft für Informatik (Arbeitskreis 5.10.3) erarbeiteten Vorschlags zur Vereinheitlichung der Spezifikation von Fachkomponenten sollen potentielle Schwächen des EJB-Komponentenmodells herausgearbeitet, aber auch die Anwendbarkeit der Spezifikationsebenen in Bezug auf diese Komponententechnologie verifiziert werden. Im weiteren Verlauf soll anhand einer Integrationsapplikation aus dem Bereich der Telekommunikation, die auf der Basis von EJB's implementiert wurde, der Frage nachgegangen werden, welche Granularität Komponenten-Interfaces im Kontext mit der den Komponenten inhärenten Funktionalitäten

besitzen. Dafür werden entsprechende Software-Metriken vorgeschlagen, um ein grundlegendes Gefühl für den Umfang und die an der Schnittstelle angebotene Funktionalität zu vermitteln. Die Zielstellung besteht zu einem darin, eine erste Positionierung zum Thema „Wann sind Enterprise Java Beans Fachkomponenten?“ zu erarbeiten, zum anderen die Verwendbarkeit der Spezifikationsebenen im Kontext einer industriellen Komponententechnologie zu analysieren.

2 Bestandteile einer EJB-Komponente

Im Folgenden soll ein Überblick zum EJB-Komponentenmodell entsprechend der Spezifikation in der Version 1.1 bzw. zum Teil Version 2.0 gegeben werden. Abbildung 1 verdeutlicht zum einem den Prozess der Entwicklung, zum anderen die Bestandteile einer EJB. Bis zur EJB-Spezifikation 1.1 wurden die sogenannten SessionBeans (realisieren funktionales Verhalten) und EntityBeans (realisieren Persistenzeigenschaften) unterschieden. Beiden Komponententypen ist die Verwendung eines sogenannten Home-Interfaces zur Erzeugung und Steuerung der Komponente und eines Remote-Interfaces für die Zugriffe eines Clients auf die Dienste der Komponente gemeinsam. Ab der Spezifikation 2.0 stehen darüber hinaus lokale Ausprägungen der oben genannten Interfaces (Home und Remote) zur Verfügung, welche die Möglichkeit einer kostengünstigeren Kommunikation zwischen EJB-Komponenten innerhalb einer virtuellen Java-Maschine bieten. Wir wollen im weiteren vom Component-Interface sprechen, das sowohl die Funktionen des Remote-Interfaces als auch dessen lokale Ausprägung berücksichtigt.

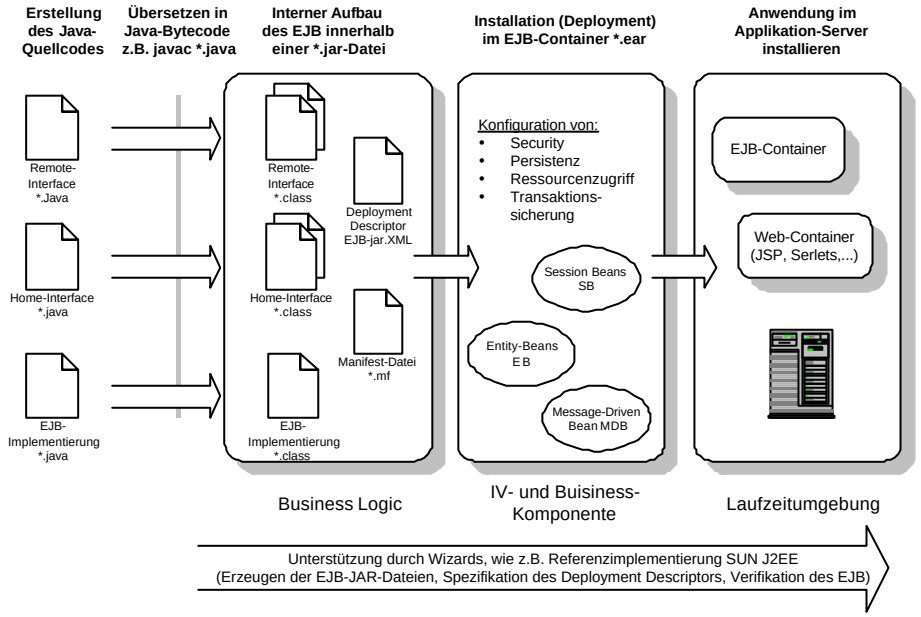

Abbildung 1: Vorgehensweise der Entwicklung/Verteilung mit EJB‘s

Zur Entwicklung eines EJB werden mindestens 3 Java-Quellcodedateien benötigt: eine für das Home-Interface, eine für das Component-Interface und eine für die eigentliche Bean-Klasse. Im Falle einer Entity Bean mit der Eigenschaft CMP[1] kommt hier ggf. noch eine Klasse für den Primary-Key zur eindeutigen Identifizierung hinzu. Die Implementierung der

[1] Container Managed Persistence

Java-Quellcodedateien (*.java) erfolgt innerhalb eines Editors bzw. einer Entwicklungsumgebung, wie z.B. dem JBuilder von Borland. Nach dem Übersetzen (Compilerlauf) der Java-Quellen in entsprechenden Java-Bytecode (*.class) erfolgt der eigentliche Vorgang zur EJB-Erzeugung (Paketierung), wobei eine entsprechende *.jar-Datei (Java-Archiv) erzeugt wird. Dieses Archiv enthält zusätzlich zu den Java-Klassen einen XML-Deployment-Descriptor und eine sogenannte Manifest-Datei (*.mf). Der Deployment-Descriptor erlaubt die Konfiguration vielfältiger Eigenschaften der EJB-Komponente, wie z.B. die Festlegung des Persistenzverhaltens, die Durchführung der Transaktionssteuerung oder aber die Identifizierung von durch die EJB referenzierten anderen EJB-Komponenten.

Auf der Grundlage derartiger EJB-Komponenten können dann komplette Applikationen zusammengestellt werden, die darüber hinaus Komponenten für z.B. einen webbasierten Zugriff (Servlets) bieten. Auf Message Driven Beans (ab EJB-Spezifikation 2.0) wird hier nicht weiter eingegangen, da diese ausschließlich als Listener für MOM[2]-Systeme eingesetzt werden und keine expliziten Interfaces für einen Client-Zugriff bieten.

3 Verwendbarkeit der Spezifikationsebenen im EJB-Kontext

Die ordinale bzw. platzierende Bewertung des EJB-Komponentenansatzes erfolgt hinsichtlich der Zweckmäßigkeit des Einsatzes der vorgeschlagenen Spezifikationsebenen, der expliziten Berücksichtigung innerhalb der EJB-Komponenten bzw. potentieller Widersprüche zu diesen:

- *Schnittstellenebene* – Auf dieser Ebene gilt es, die durch die Komponente öffentlich angebotenen Dienste hinsichtlich der verwendeten Signatur, übertragender Parameter inklusive der dabei verwendeten Datentypen und die potentiellen Fehlermeldungen zu spezifizieren. Darüber hinaus sind potentielle Dienste anzugeben, welche die Komponente zum Erbringen ihrer Leistung benötigt.

  Der Zugriff auf die fachlich begründeten Funktionen einer EJB-Komponente erfolgt, wie bereits dargestellt, über das Component-Interface (entsprechend der EJB-Spezifikation 2.0 das Remote-Interface bzw. dessen lokale Ausprägung). Aus fachlicher Sicht kommt dem Component-Interface damit die ausschließliche Bedeutung zu. Im Folgenden wird die Implementierung eines Remote-Interfaces beispielhaft wiedergegeben.

```
public interface EuroCalcRemote extends javax.ejb.EJBObject {

  // Umrechnung Euro-Betrag in DM
  public double euro_to_dm(double amount) throws RemoteException;

  // Umrechnung DM-Betrag in Euro
  public double dm_to_euro(double amount) throws RemoteException;
}
```

  Die Verwendung der OMG IDL entsprechend [Fachkomponenten 2002] zur Spezifikation auf Schnittstellenebene bringt bei der Verwendung von EJB's aus unserer Sicht keine signifikanten Vorteile, da die Generierung von Stubs bzw. Skeletons bereits eine EJB-inhärente Funktionalität auf der Basis des vorgestellten Component-Interface ist. Nicht

[2] MOM Message oriented Middleware

abgedeckt wird auf dieser Grundlage die explizite Kennzeichnung von Diensten (interface extern), welche durch die Komponente benötigt werden.

- *Verhaltensebene* – Die Verhaltensebene dient primär der Beschreibung von Vor- und Nachbedingungen konkreter Dienste der Komponente. Eine Vorbedingung kann dabei zum Beispiel die Belegung konkreter Parameter der Schnittstellendefinition sein. Darüber hinaus können Invarianten für interne Zustände der Komponente vergeben werden.

  Die Formulierung von Vor- und Nachbedingungen bzw. Invarianten kann mittels der vorgeschlagenen primären Notation der Object Constraint Language außerhalb der EJB durchgeführt werden. Die Sicherstellung dieser Festlegungen kann sowohl programmtechnisch innerhalb des Remote-Interfaces als auch dynamisch durch Verwendung von Umgebungsvariablen zur wertmäßigen Belegung spezifizierter Invarianten der EJB-Komponenten erfolgen. Eine durchgängige, d.h. weitgehend automatisierte Verwendung der OCL-basierten Spezifikation im Rahmen des Komponentendesigns ist zwar wünschenswert, wird aber derzeit aus Sicht der Autoren von keinem CASE-Werkzeug (z.B. Rational Rose, Telelogic) explizit unterstützt.

- *Abstimmungsebene* – Die Abstimmungsebene dient der Festlegung von Reihenfolgebeziehungen zwischen den Diensten verschiedener Komponenten als auch den Diensten innerhalb einer Komponente. Entsprechende Informationen zu dieser Ebene zeigen Szenarien der Interaktion mit anderen Komponenten.

  Die Beziehungen zwischen den Diensten verschiedener EJB-Komponenten werden innerhalb des Deployment-Descriptors beschrieben. Festlegungen zur Reihenfolge, wie dieses die Spezifikation vorsieht, erfolgen dabei jedoch nicht. Dementsprechend sind, sofern diese Angaben zur fehlerfreien Erledigung der Aufgabenstellung einer Fachkomponente benötigt werden, diese in externen Dokumenten entsprechend dem Vorschlag der Spezifikation niederzulegen.

- *Qualitätsebene* – Ziel dieser Ebene ist es, die qualitativen Eigenschaften (z.B. Antwortzeit und Durchsatz) der Komponente und ihrer angebotenen Dienste zu erfassen. Häufig wird in diesem Zusammenhang auch von den nichtfunktionalen Eigenschaften gesprochen.

  Die Spezifikation der Qualitätseigenschaften einer EJB-Komponente ist ein derzeit weitgehend ungelöstes Problem. Die Spezifizierung dieser Eigenschaften erfordert, wie unter [Schmietendorf 2001] dargestellt, die Analyse der Komponenten im Rahmen einer entsprechenden Referenzumgebung. Dabei gilt es weitgehend generische Qualitätsaussagen zu gewinnen, die im Rahmen der komponentenorientierten Erstellung von Applikationen auch sinnvoll verwendet werden können. Auf dieser Grundlage gewonnene Aussagen zu qualitativen Eigenschaften einer EJB-Komponenten sind in externen Dokumenten, oder aber als inhärente Beschreibung innerhalb des Komponentenarchivs (*.jar) zu pflegen. Eine weitere Möglichkeit, qualitative Aussagen über die betroffene Komponente zu gewinnen, besteht in der Aufnahme statischer Quellcodemetriken. Auf diese Vorgehensweise wird im nächsten Abschnitt ausführlich eingegangen.

- *Terminologieebene* – Diese Ebene regelt die semantischen Eigenschaften der Komponente in Bezug auf die verwendeten Begrifflichkeiten in Bezug auf die unterstützte Anwen-

dungsdomain. Insbesondere für Fachkomponenten ist diese Festlegung wichtig, regelt sie doch, was z.B. unter einem Datenobjekt „Kunden“ aus fachlicher Sicht zu verstehen ist.

- *Aufgabenebene* - Ziel dieser Ebene ist die Darstellung der durch die Komponente realisierbaren Funktionalitäten aus fachlicher Sicht. Dabei können sowohl die Komposition als auch Dekomposition von Aufgabenstellungen im Rahmen eines komplexen Gesamtsystems berücksichtigt werden. Entsprechend der Spezifikation wird hier die Verwendung einer Fachnormsprache vorgeschlagen. Ziel ist es, die verwendeten Begrifflichkeiten weitgehend eindeutig zu gestalten und die Bildung von Aussagen durch eine entsprechende Grammatik zu unterstützen. Aus Sicht der Autoren stellt sich hier die Frage, inwieweit unterschiedliche Branchen und/oder Anwendungsdomänen zum Bedarf spezifischer Fachnormsprachen führen.

- *Vermarktungsebene* - Für den Handel von Komponenten (COTS) besteht der Bedarf, allgemeine technische, betriebswirtschaftliche und organisatorische Informationen (z.B. Ansprechpartner für den Fehlerfall und Reaktionszeiten) vorzuhalten. Ziel dieser Ebene ist es, entsprechende Kataloge von Komponenten zu unterstützen.

  Da für die EJB-Komponenten keine Spezifikation verfügbar ist, welche die Inhalte der *Terminologie-, Aufgaben- und Vermarktungsebene* aufgreift, kann die vorgeschlagene Notation theoretisch ohne Änderungen zum Einsatz kommen.

Die derzeitige EJB-Spezifikation 2.0 unterstützt nur ausgewählte Merkmale der ersten drei Spezifikationsebenen explizit, die weiteren können verwendet werden, schlagen sich jedoch nur implizit innerhalb der EJB-Komponenten nieder.

Insgesamt kann festgestellt werden, dass die Spezifizierung entsprechend [Fachkomponenten 2002] nur für ausreichend „grob granulare“ Komponenten überhaupt sinnvoll ist. Für einzelne EJB-Komponenten erscheint dieses keineswegs zielführend, da es vom Aufwand her nicht zu vertreten ist. Wir wollen daher die Eigenschaft der Granularität EJB-basierter Komponenten im folgenden Abschnitt durch eine metrikenbasierte Analyse einzelner EJB bzw. mehrerer EJB’s, die über eine sogenannte Session Facade (J2EE-Pattern) gekapselt werden, operationalisieren.

4 Metriken-basierte Bewertung

4.1 Metriken-basierte Analyse elementarer EJB‘s

Eine wesentliche Rolle für die erfolgreiche Wiederverwendung von EJB-Komponenten spielt deren Granularität, also die Größe einer solchen Komponente. In diesem Zusammenhang liegt es nahe, Kriterien für die „richtige“ Größe von EJB-Komponenten über empirische Untersuchungen vorhandener Komponenten zu finden. Für die metrikenbasierte Analyse wurden 24 EJB (20 Entity-Beans und 4 Session-Beans) herangezogen. Im Folgenden sollen dementsprechend 24 EJB-Komponenten, die eine EAI[3]-Integrationsapplikation zur Steuerung und Überwachung des Bereitstellungsprozesses von Netzwerkprodukten der Telekommunikation

[3] Enterprise Application Integration

realisieren, untersucht werden. In einem ersten Ansatz wurden die Bestandteile dieser Komponenten hinsichtlich ihres Umfangs (d.h. effektiver Lines of Code - eLoC) untersucht.

In Abbildung 2 kann der Umfang von 20 untersuchten EJB-Komponenten (hier vom Typ Entity-Beans) nachvollzogen werden. Auffällig ist, dass der größte Teil der Komponenten bei der Bean-Klasse einen Umfang von ca. 100 bis 200 eLoC aufweist.

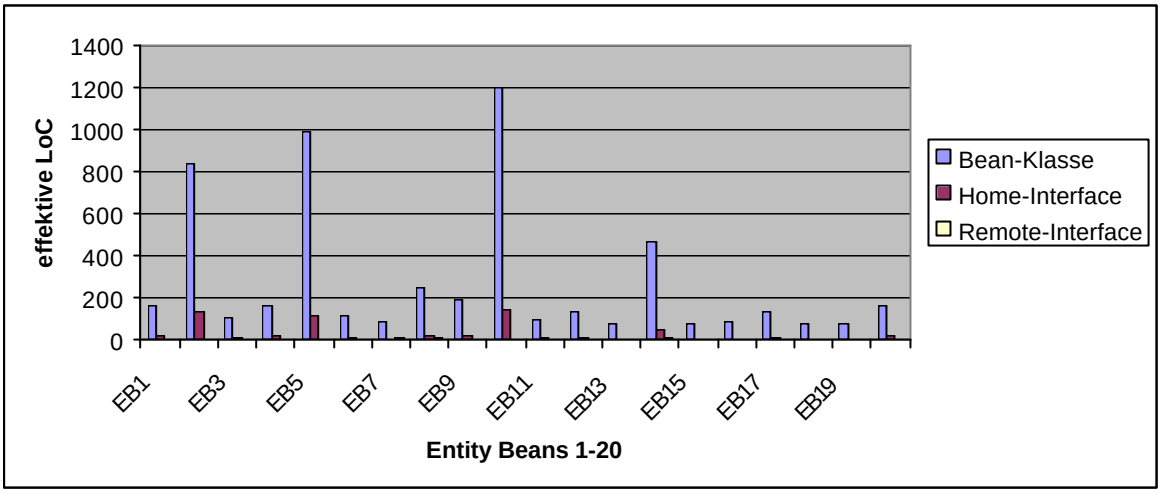

Abbildung 2: Umfangsmetriken von EntityBean-Komponenten (kurz EB)

Die Größe des Home-Interfaces korreliert in der untersuchten Version sehr stark mit der Größe der Bean-Klasse. Dabei stellt sich ein Verhältnis von 1:8 bei einem Korrelationskoeffizienten von $r_{XY} = 0.985$, wie in Abbildung 3 ersichtlich ist, ein. Das Remote-Interface weist über alle Entity-Bean-Komponenten eine Größe von durchschnittlich 7 eLoC auf.

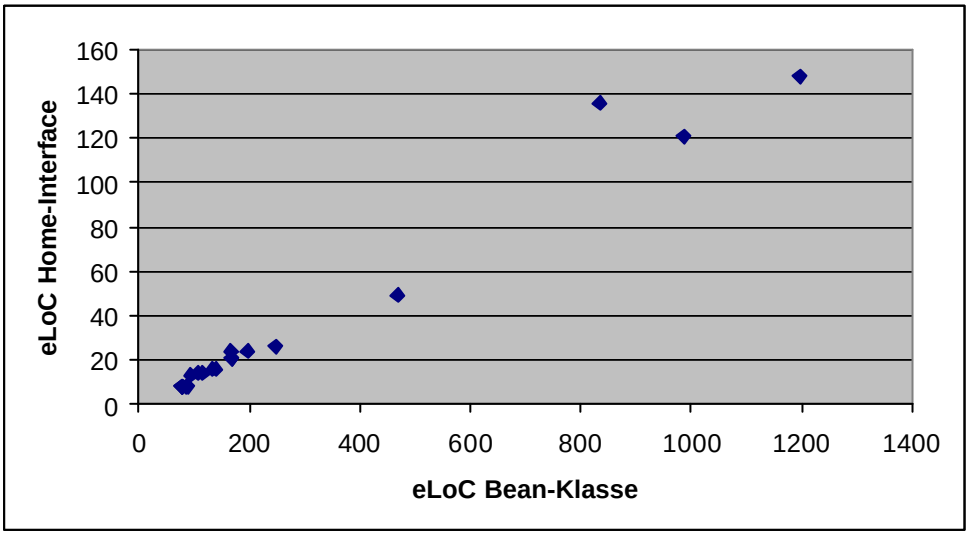

Abbildung 3: Zusammenhang zwischen Bean-Klasse und Home-Interface

Im Folgenden sollen 4 SessionBeans untersucht werden. Während die Größe der Bean-Klasse, welche die fachliche Funktionalität realisiert, zwischen minimal 164 eLoC und maximal 756 eLoC schwankt, bleibt der Umfang der Home- bzw. Remote-Interfaces mit ca. 6 eLoC bzw. 10 eLoC weitgehend konstant.

Durch die EJB-Komponenten werden innerhalb des Remote-Interfaces die folgende Anzahl von Diensten angeboten:

- Session Bean 1: 4 angebotene Dienste
- Session Bean 2: 3 angebotene Dienste
- Session Bean 3: 5 angebotene Dienste
- Session Bean 4: 6 angebotene Dienste.

Der Anzahl der angebotenen Dienste kann durchaus als schmale und übersichtliche Schnittstelle angesehen werden, welche dem 7±2 Gesetz von G. A. Miller (1956) folgt.

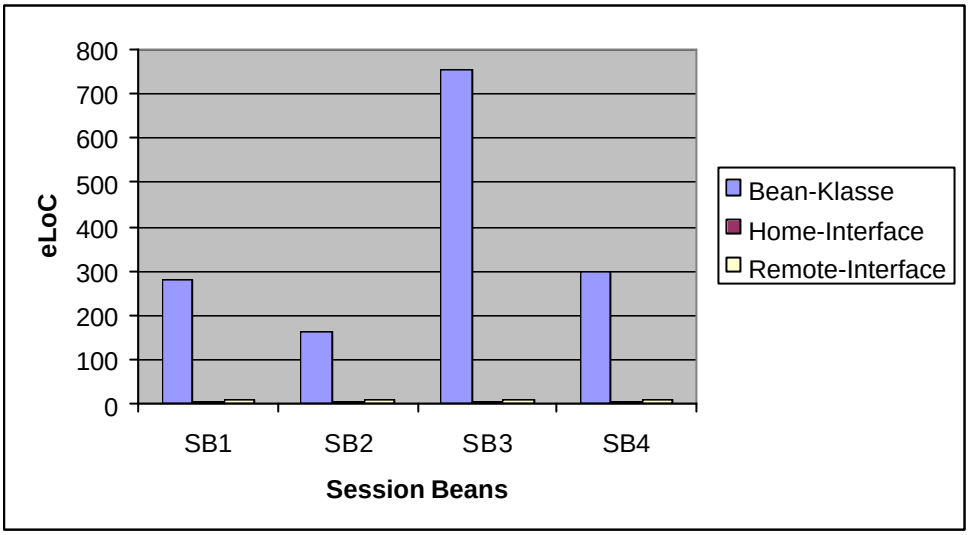

Abbildung 4: Umfangsanalyse von Session Beans (kurz SB)

4.2 EJB-basierte Fachkomponente aus der Basis eines Design-Pattern

Die vorhergehende Analyse elementarer EJB zeigte ausgewählte Eigenschaften dieser Komponenten auf. Es sei darauf verwiesen, dass hier selbstverständlich noch keine statistische Sicherheit aufgrund der geringen Anzahl analysierter Komponenten erzielt werden konnte. Dennoch zeigen sich erste Trends, die die Aussagen in [Schmietendorf 2000], [Lezius 2001], [Beneken 2002] weitgehend bestätigen. Insbesondere die geringe Größe und die hohen Abhängigkeiten von anderen Komponenten legen den Schluss nahe, dass nicht jede EJB automatisch eine Fachkomponente ist bzw. überhaupt sein kann. Erst der Einsatz entsprechender EJB-Muster (Design Pattern) erlaubt aus Sicht der Autoren tatsächlich die Entwicklung von Fachkomponenten. In diesem Zusammenhang hat sich insbesondere das Muster der Session Facade durchgesetzt. Weitere Informationen zu Design Pattern im Kontext EJB-basierter Komponenten finden sich z.B. unter [J2EEPattern 2001], [Schmietendorf 2002] und [Beneken 2002].

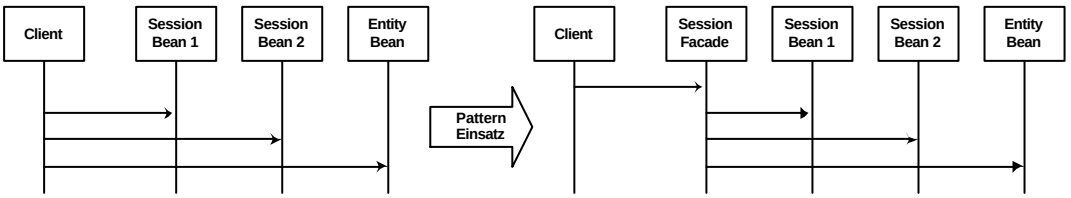

Abbildung 5: Einsatz des Session-Facade-Musters

Das Session-Facade-Muster (siehe auch Abbildung 5) unterstützt die Entwicklung wiederverwendbarer grobgranularer Fachkomponenten, reduziert potentiellen Netzwerkverkehr zwischen Client und Server und bietet darüber hinaus die Möglichkeit eines zentralisierten Transaktions- und Security-Managements. Innerhalb der im vorhergehenden Abschnitt analysierten Session Beans werden entsprechend dem Entwurfsmuster Session Facade mehrere Entity Beans referenziert, so dass der Zugriff auf die komplette Fachkomponente nur noch über die bereits dargestellten Session Beans erfolgt. Die folgende Tabelle gibt eine zusammenfassende Sicht auf die Anzahl der jeweils referenzierten Komponenten (hier Entity Beans) und deren Gesamtumfang in eLoC wieder:

Tabelle 1: Übersicht der Session Facade Fachkomponenten (SB und EB)

Fachkomponenten mit Session Facade	Anzahl angebotener Dienste	Umfang der Session Beans in eLoC	Anzahl referenzierter Entity Beans	eLoC aller Entity Beans
Fachkomp. 1 Session Bean 1	4	281	5	1020
Fachkomp. 2 Session Bean 2	3	164	4	772
Fachkomp. 3 Session Bean 3	5	756	16	4290
Fachkomp. 4 Session Bean 4	4	300	7	1301

Es zeigt sich, dass sich zwischen dem Umfang der Session Beans und der Anzahl, aber auch dem Umfang in eLoC der referenzierten Entity Beans ein Zusammenhang ergibt.

Abbildung 6 zeigt diesen Zusammenhang noch einmal deutlicher auf, wobei sich für die 4 Stichproben ein Korrelationskoeffizient von $r_{XY} = 0{,}99$, d.h. eine sehr hohe Korrelation ergibt.

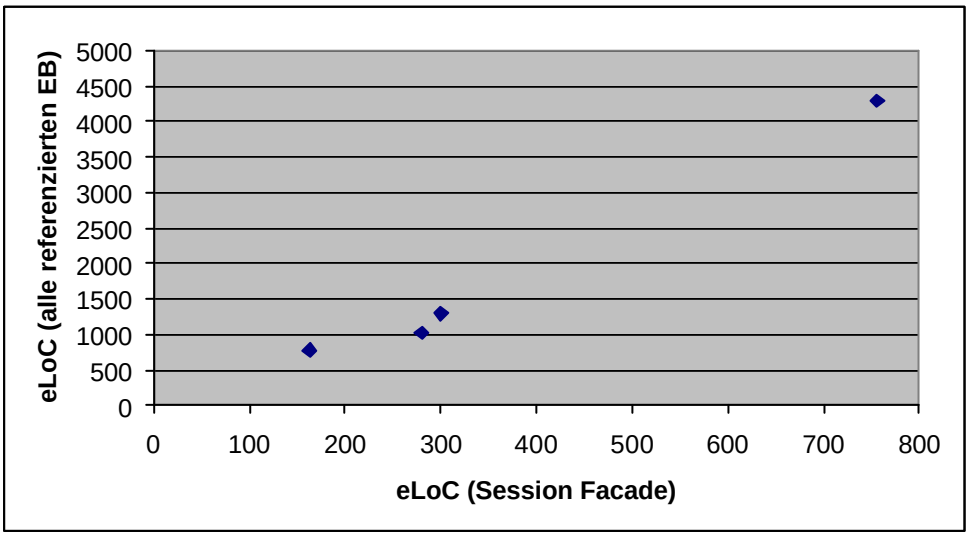

Abbildung 6: Session-Facade und Summe der Entity Beans

4.3 Potentielle Design-Empfehlungen

In Auswertung der durchgeführten Untersuchungen könnten erste Design-Empfehlungen[4] für Fachkomponenten, welche auf der Grundlage von EJB-Komponenten erstellt werden, folgendermaßen aussehen:

- Die Schnittstelle zu einer Fachkomponente wird immer auf der Grundlage einer Session Facade Bean und deren Home- bzw. Remote-Interface gebildet.
- Schnittstellen innerhalb der Fachkomponente zwischen verwendeten EJB sollten als lokale Interfaces realisiert werden und von außen nicht sichtbar sein.
- Das Remote-Interface sollte nicht mehr als 7±2 Dienste enthalten, die sowohl den Zugriff auf die Komponente als auch die Bündelung externer Abhängigkeiten regeln.
- Der Umfang einer Fachkomponente sollte mindesten 1.000 bis einige 10.000 eLoC groß sein und auf mehreren EJB-Komponenten basieren.
- Zwischen dem Umfang der Session Facade und den dahinter liegenden EJB-Komponenten sollte sich ein Verhältnis von 1:5 bis 1:10 einstellen.

5 Zusammenfassung und Ausblick

Ziel dieses Artikels war es nicht, fertige Lösungen vorzustellen. Vielmehr sollte der industrielle Bedarf verdeutlicht werden, die Spezifikationsebenen von Fachkomponenten auch praktisch umsetzen zu können. In diesem Zusammenhang kommt den EJB-Komponenten derzeit eine herausragende Bedeutung zu. Durch die bei dieser Technik gegebenen Möglichkeit der Trennung von anwendungs- und technikspezifischem Quellcode können Anwendungssysteme auch bei einem Wechsel genutzter Services (z.B. Datenbankmanagementsystem) weiterhin stabil betrieben werden, ohne Änderungen am Quellcode der Applikation vorzunehmen, was einen effektiven Investitionsschutz darstellt.

Die durchgeführten empirischen Analysen zur Granularität von EJB-Komponenten stellen einen ersten Ansatz dar, um potentielle Bewertungskriterien von Fachkomponenten zu gewinnen. Zur effektiven Verwendung der Ergebnisse von Softwaremessungen werden empirisch untersetzte Bewertungsmodelle benötigt, die derzeit noch nicht zur Verfügung stehen. Das hier in Anlehnung an [Griffel 1998] verwendete Bewertungsmodell basiert noch auf sehr vagen Annahmen und muss selbstverständlich einer umfangreichen Validation unterzogen werden werden. In diesem Zusammenhang ist insbesondere die Verwendung weiterer Softwaremetriken zur statischen, aber auch dynamische Analyse von EJB-Komponenten geplant.

Über die Verwendung von Softwaremetriken kann das Design einer Fachkomponente analysiert werden, nicht jedoch die Feststellung, dass es sich um eine Fachkomponente handelt. Inwieweit es sich bei einer EJB bzw. der Aggregation mehrerer EJB‘s über eine Session Facade um eine Fachkomponente handelt, hängt selbstverständlich nicht vom Umfang oder der Anzahl verfügbarer Schnittstellen ab, sondern vielmehr von der fachlichen Zweckbestimmung. Da hier jedoch semantische Aspekte im Vordergrund stehen, die einer Formalisierung nur schwer zugänglich sind, kann die Festlegung, ob es sich um eine Fachkomponente han-

[4] Aufgrund der geringen statistischen Sicherheit der durchgeführten Untersuchungen sind diese zunächst als Ansätze für weitere Untersuchungen denn feststehende Erkenntnisse zu werten!!!

delt, nur durch den Komponentenentwickler bzw. den Anwender (Assembler) selbst durchgeführt werden.

Im Ergebnis der durchgeführten Untersuchungen wollen wir die folgende These aufstellen:

> *Über eine Session Facade (Session Bean) referenzierte EJB-Komponenten bieten die Möglichkeit zur Implementierung von Fachkomponenten, keinesfalls aber ein elementares EJB.*

6 Quellenverzeichnis

[Beneken 2002] Beneken, G.; Schamper, M.: Qualität ist das beste Rezept - Komponenten mit J2EE-Pattern. Java Spektrum, Ausgabe 2, Februar/März 2002

[Fachkomponenten 2001] 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.: Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme, Februar 2002

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

[Hoyt 2001] Hoyt, M.: A Framework for Software Component Interface Specification and Analysis. master thesis, University of Waterloo, Ontario, Canada, 2001

[J2EEPattern 2001] Sun Microsystems: Sun Java Center J2EE Patterns, J2EE Patterns Catalog; Version 1.0 Beta; 2001 (URL: developer.java.sun.com)

[Lezius 2001] Lezius, J.: Qualitätsbewertung von Softwarekomponenten auf der Basis von Metriken. Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, 2001 (fachliche Betreuung am Entwicklungszentrum Berlin der T-Nova)

[Schmietendorf 2000] Schmietendorf, A.; Dumke R.: Metriken-basierte Bewertung von Software-Komponenten. In: Proc. CONQUEST 2000, Nürnberg, 14./15.9.2000, S. 104

[Schmietendorf 2001] Schmietendorf, A.; Dumke, R.: Spezifikation von Softwarekomponenten auf Qualitätsebene. In Turowski, K. (Hrsg.): Tagungsband zum 2. Workshop Modellierung und Spezifikation von Fachkomponenten (im Rahmen der vertIS 2001), Universität Bamberg, Oktober 2001, S. 113-123

[Schmietendorf 2002] Schmietendorf, A.; Dimitrov, E.; Dumke, R.: Enterprise Java Beans – Komponentenbezogenes Software Engineering. MITP-Verlag: Bonn, August 2002