Spezifikation der Performance-Eigenschaften von Softwarekomponenten

Andreas Schmietendorf[*], André Scholz[#]

* *Otto-von-Guericke-Universität Magdeburg, Fakultät Informatik, Institut für verteilte Systeme, Arbeitsgruppe Software-Technik, Universitätsplatz 2, D-39016 Magdeburg, Email: schmiete@ivs.cs.uni-magdeburg.de*

\# *Otto-von-Guericke-Universität Magdeburg, Fakultät Informatik, Institut für technische und betriebliche Informationssysteme, Arbeitsgruppe Wirtschaftsinformatik, Universitätsplatz 2, D-39016 Magdeburg, Email: ascholz@iti.cs.uni-magdeburg.de*

Zusammenfassung: Die komponentenorientierte Entwicklung von Softwaresystemen mit restriktiven Performance-Anforderungen, im Sinne von Antwortzeit- und Durchsatz, erfordert eine besonderen Beachtung. Dieses betrifft sowohl die Seite der Komponentenentwickler als auch die Seite der Applikationsentwickler, die auf der Basis von Komponenten Anwendungen erstellen. Die derzeit verwendeten Softwarekomponenten bieten zumeist keine Aussagen hinsichtlich ihres Performanceverhaltens während der Programmausführung. Eine kostengünstige Abklärung potentieller Performance-Probleme ist in den frühen Phasen einer komponentenorientierten Entwicklung nicht möglich. Probleme mit der Performance des Systems werden so erst beim Abnahmetest oder während der Systemeinführung erkannt, wodurch aufwendige Nacharbeiten, die sich typischerweise über alle Phasen der Softwareentwicklung erstrecken, notwendig werden. Der vorliegende Beitrag stellt einen Ansatz zur Beschreibung und Analyse der Performance-Eigenschaften von Softwarekomponenten vor. Weiterhin wird auf die spezielle Problematik der Realisierung dieser Aufgabenstellungen über einen vertrauenswürdigen „Dritten“ im Sinne eines Komponentenvertriebs eingegangen. Am Beispiel von „Enterprise Java Beans“ wird die direkte Integration der Performancebeschreibung innerhalb einer Softwarekomponente konzeptionell aufgezeigt. Die sich daraus ergebenden Innovationspotentiale dieser Technologie in Hinsicht auf ein effizientes Performance-Management werden anhand ausgewählter Beispiele skizziert.

Schlüsselworte: Benchmark, Komponenten, Performance-Analyse, Performance-Beschreibung, Performance Engineering, Qualitätsklassen

1 Motivation

Die Entwicklung von Softwaresystemen erfolgt zunehmend auf der Basis vorgefertigter Komponenten und Services. Eine komponentenbasierte Entwicklung impliziert eine arbeitsteilige Softwareentwicklung. Zum einen können spezialisierte Anbieter benötigte Komponenten zur Verfügung stellen. Zum anderen können Anwendungsentwickler auf der Basis derartiger Komponenten kundenspezifische Softwareprodukte zusammenstellen bzw. konfigurieren. Da Komponenten, wie z.B. JavaBeans, so einsetzbar sein sollten, dass keine Kenntnisse des internen Aufbaus und der Struktur notwendig sind, ergibt sich der Bedarf einer Beschreibung des nach außen sichtbaren Verhaltens hinsichtlich ihrer funktionalen und qualitativen Eigenschaften. Während die Funktionalität einer Komponente bei den heute am Markt verfügbaren Komponenten beschrieben wird, finden sich Qualitätsaussagen nur äußerst selten. Mit der zunehmenden Akzeptanz der komponentenbasierten Entwicklung wächst die

Forderung, auch nicht-funktionale Eigenschaften zu determinieren, da die Auswahl qualitativ unzureichender Komponenten zu vielfältigen Problemen führen kann.

Eine der wenigen Aussagen zum Umgang mit qualitativen Eigenschaften von Komponenten findet sich bei der Schweizer Bank. Entsprechend (Szyperski 1998) wurden dort auch Qualitätsanforderungen für Softwarekomponenten vertraglich fixiert Die Festlegungen beziehen sich sowohl auf funktionale Aspekte, wie die Syntax und Semantik, als auch auf sogenannte Quality Service Level. Diese Service Level decken Komponenteneigenschaften, wie z.B. Verfügbarkeitsangaben, Durchsatz oder Latenzzeiten, ab. Diese zugesicherten Eigenschaften werden während der Entwicklung und des Wirkbetriebs überwacht. Im Falle der Nichteinhaltung werden Vertragsstrafen für den Komponentenanbieter fällig (Scholz/Turowski 2000).

Die Determinierung der vorgenannten Qualitätsanforderungen bleibt jedoch auch in diesem Praxisfall unstrukturiert. Ebenso finden sich keine Aussagen, wie beim Wirkbetrieb unter Berücksichtigung der komplexen Interaktion zwischen Hard- und Softwareschichten sowie des Zusammenspiels zwischen verschiedenen Komponenten die Einhaltung der Service Level überprüft werden kann. Der vorliegende Beitrag schlägt einen Rahmenkonzept für die Qualitätsspezifizierung und den Vertrieb von Softwarekomponenten vor. Er fokussiert dazu im Schwerpunkt auf die Betrachtung des Qualitätsfaktors Performance.

2 Analyse der Ausgangssituation

Ein allgemeiner Vorschlag für eine Komponentenbeschreibung, wie er innerhalb der Deutschen Telekom AG eingeführt wurde, findet sich unter (Schmietendorf/Dumke 2000). Zur Beschreibung werden verschiedene Aspekte, wie z.B. administrative Informationen, domainspezifische Angaben oder auch qualitative Eigenschaften, entsprechend der ISO 9126 herangezogen:

- Übertragbarkeit (Anpassbarkeit, Installierbarkeit),
- Anwendbarkeit (Erlernbarkeit, Beherrschbarkeit, Verständlichkeit),
- **Effizienz (raum-, zeit- und ressourcenbezogen),**
- Funktionalität (Angemessenheit, Interoperabilität, Genauigkeit),
- Zuverlässigkeit (Fehlertoleranz, geringe Fehlerhäufigkeit, Fehlerverfolgbarkeit),
- Wartbarkeit (Analysierbarkeit, Änderbarkeit, Stabilität, Testbarkeit).

Für eine Beschreibung der Performance- bzw. Effizienz-Eigenschaften von Komponenten ist eine strukturierte Vorgehensweise notwendig. Innerhalb eines technischen Reports des Software Engineering Institute (SEI) findet sich dazu die folgende Feststellung (Klein 1996):

> *„No standard of practice exist for how to evaluate the performance of a software component. As reliance on COTS increases, the risk of discovering performance problems after it is too late may increase for systems for which performance predictability is critical."*

Folglich steht derzeit keine standardisierte Vorgehensweise zur Performance-Analyse einer kommerziellen Softwarekomponente zur Verfügung. In einem ersten Ansatz wird innerhalb des Reports die Entwicklung eines Rahmenwerks vorgeschlagen, das sowohl den Softwareentwicklern als auch den Komponentenanbietern zur Verfügung steht. Dem Softwareentwickler sollen dadurch potentielle Risiken bei der Verwendung der Komponenten verdeutlicht

werden und dem Komponentenanbieter soll eine Standardprozedur bei der Absicherung von Komponenteneigenschaften zur Verfügung gestellt werden.

3 Einführung von Qualitätsklassen

In Erweiterung des allgemeinen, 4-stufigen Zertifizierungsvorschlages von Komponenten (Meritt 1994) schlagen die Autoren die Einführung von Qualitätsklassen vor, die sich an den Kriterien der ISO 9126 orientieren (vgl. **Tabelle 1**).

Tabelle 1: Qualitätsklassen für Komponenten

Übertragbarkeit Q1	**Anwendbarkeit Q2**	**Effizienz Q3**	**Funktionalität Q4**	**Zuverlässigkeit Q5**	**Wartbarkeit Q6**
Anpassbarkeit	Erlernbarkeit	Raumbezogen	Angemessenheit	Fehlertoleranz	Analysierbarkeit
Installierbarkeit	Beherrschbarkeit	Zeitbezogen	Interoperabilität	Fehlerhäufigkeit	Änderbarkeit
	Verständlichkeit	Ressourcenbezogen	Genauigkeit	Fehlerverfolgbarkeit	Stabilität
					Testbarkeit

Die Qualitätseigenschaften einer Komponente können damit allgemein als 6-Tupel beschrieben werden:

Komponente_1 (q1, q2, q3, q4, q5, q6);

Nicht für jede Komponentenspezifizierung ist eine vollständige Angabe aller Qualitätsklassen erforderlich und damit vom verbundenen Aufwand her zu vertreten. Darüber hinaus kann eine formalisierte Beschreibung der Qualitätsklassen zumeist nur von nominal- oder ordinalskalierten Metriken ausgehen, d.h. qualitative Eigenschaften können nur in einer identifizierenden oder platzierenden Art und Weise beschrieben werden. Im speziellen Fall des hier untersuchten Qualitätskriteriums der Effizienz kann hinsichtlich der verwendbaren Metriken, wie z.B. Antwortzeiten oder auch Auslastungen, zumindest von intervallskalierten Metriken ausgegangen werden. Allgemein kann festgestellt werden, dass ein höheres Skalenniveau[1] verwendeter Metriken zu einem größeren Informationsgehalt und der Möglichkeit „höherwertiger meßtheoretischer Analysen“ führt. Weitere Informationen zum spezielle Themengebiet der Software-Metriken und Skaleneigenschaften finden sich z.B. unter (Dumke 2000).

Für die Abbildung der Informationen zu den ermittelten Qualitätseigenschaften einer konkreten Komponente bieten sich unterschiedliche Möglichkeiten an:

Hinterlegung in Komponenten-Repositories:

> Werden Qualitätsaussagen innerhalb entsprechender Komponenten-Repositories abgelegt, z.B. als HTML-Datei mit einem Link auf die einzusetzende Komponente selbst, kann die Suche effektiv unterstützt werden. Eine „maschinelle“ Verwendung der Angaben während der Entwicklung/Wirkbetrieb ist aber nicht möglich.

Integration innerhalb der Komponenten selbst:

> Die Abspeicherung der Informationen innerhalb der Komponenten selbst bietet mehrere Vorteile:
>
> - Zum einen kann unmittelbar bei Komponentenanwendung im Rahmen entsprechender Komponenten-Entwicklungsumgebungen auf die Qualitätseigenschaften zugegriffen werden.

[1] Skalen: Nominal, Ordinal, Intervall, Rational, Absolut (Zunahme des Skalenniveaus von links nach rechts)

- Zum anderen lassen sie sich während der Applikations-Ausführung für eine Qualitätsüberwachung direkt verwenden.

4 Spezifizierung und Vertrieb von Softwarekomponenten

Die Qualitätsspezifizierung einer Softwarekomponente sollte dem Softwareentwickler alle spezifischen Leistungscharakteristiken zur Verfügung stellen. Im Folgenden wird der vollständige Marktmechanismus schematisch beschrieben (vgl. Scholz 2000). Dem Entwickler einer Softwarekomponente wird dabei die Rolle des Komponenten-Providers zugeordnet (vgl. **Bild 1**). Demgegenüber fungiert die Person als Komponentenkäufer, die eine bestimmte Komponente in eine Umgebung integrieren möchte.

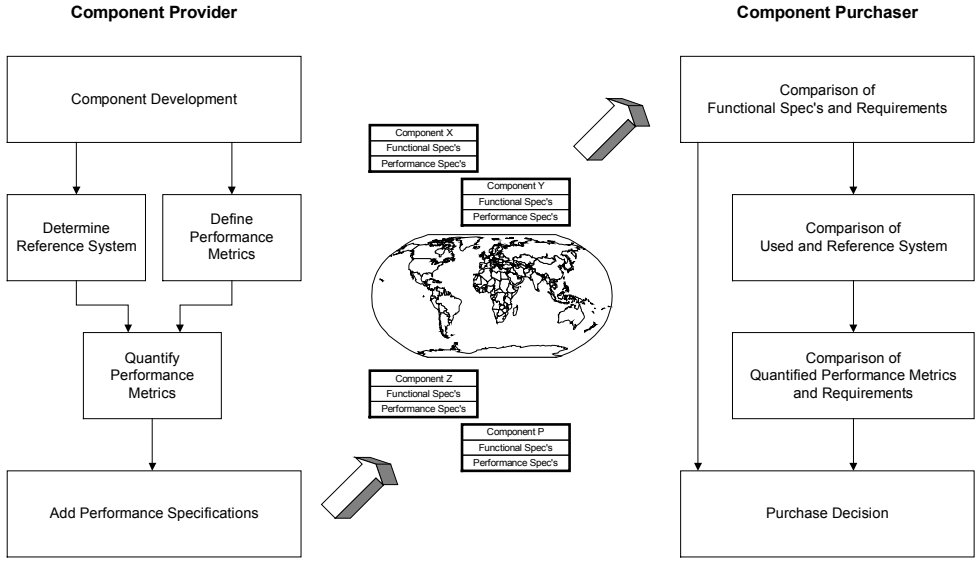

Bild 1: Konzeption und Umgang mit den Performance-Spezifikationen

Der Komponenten-Provider muss nach der Entwicklung des Softwarebausteins ein bestimmtes Referenzsystem, auf dem alle weiteren Spezifikationen basieren, festlegen. Das Referenzsystem setzt sich aus einem Hardware- und einem Betriebssystem sowie weiteren zu Grunde liegenden Services zusammen. Zumeist verfügt das Referenzsystem über die für die Ausführung der Komponente empfohlene Konfiguration.

Daneben müssen Leistungsgrößen entsprechend der Qualitätsklasse Q3 definiert werden. Je mehr Metriken zur Verfügung stehen, desto größer ist die marktbezogene Transparenz der Komponente, aber auch der Aufwand, diese zu ermitteln. Zum einen werden Metriken aus Sicht des Betreibers (z.B. CPU-Auslastung) verwendet, die die Auslastung interner Systemressourcen charakterisieren. Zum anderen werden Metriken aus Sicht des späteren Systemanwenders, wie beispielsweise Antwort- oder Ausführungszeit, berücksichtigt.

In einem nächsten Schritt sind die festgelegten Leistungsgrößen auf dem definierten Referenzsystem zu quantifizieren. Zu diesem Zweck sollte sich der Komponenten-Provider der Ausführung von Benchmarks bedienen, in deren Rahmen externe und interne Performancemetriken ermittelt werden. Dabei kann unter anderem auf bestehende Hardware- und Softwaremonitore sowie zum Zwecke der Verallgemeinerung der Ergebnisse auf standardisierte Komponenten-Benchmarks zurückgegriffen werden. Die Ergebnisse einer metrikenba-

sierten Quantifizierung der Performanceeigenschaften sowie die Spezifikation des Referenzsystems werden in die Qualitätsbeschreibung der Softwarekomponente eingefügt. Darüber hinaus werden dort ebenfalls alle Daten hinterlegt, mit deren Hilfe die Transformation der Ergebnisse auf andere Referenzsysteme vereinfacht wird. So tragen beispielsweise funktionelle Beziehungen zwischen den Ergebnissen der verschiedenen Referenzsysteme entscheidend zu einer Umwandlung bei. Nach diesen vorbereitenden Maßnahmen kann die Softwarekomponente auf dem globalen Markt angeboten werden.

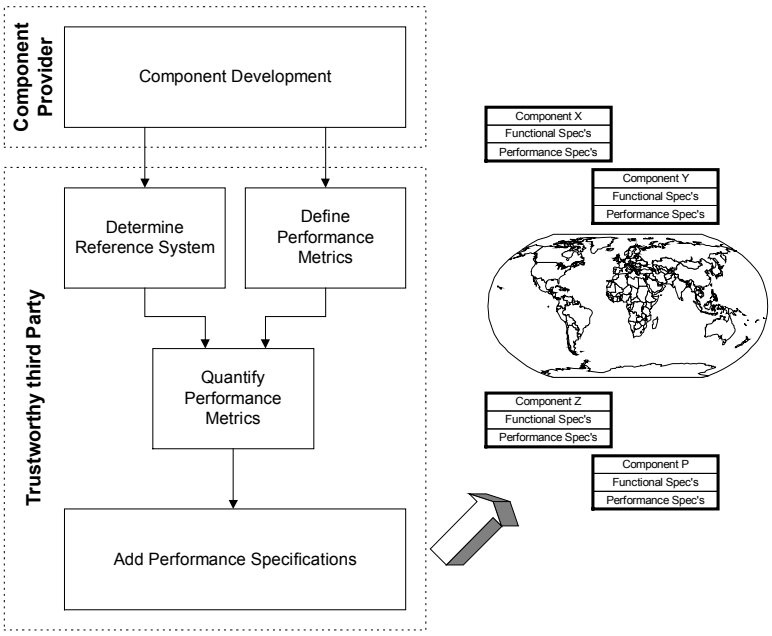

Bild 2: Performance-Spezifizierung durch eine vertrauenswürdige Institution

Auf der anderen Seite untersucht der potentielle Käufer die funktionellen Spezifikationen der Komponente. Entsprechen die funktionellen Spezifikationen den Anforderungen, werden in einem nächsten Schritt die Leistungsgrößen geprüft. Zu diesem Zweck wird das Referenzsystem mit dem System in Beziehung gesetzt, dessen Einsatz beim Anwender beabsichtigt wird. Sollten sich die Systemspezifikationen signifikant unterscheiden, muss der Komponentenkäufer alle weiteren Daten untersuchen, die in der Qualitätsbeschreibung zur Verfügung gestellt wurden. Funktionelle Beziehungen können benutzt werden, um hypothetische Werte für eine bestimmte Umgebung zu kalkulieren. In einem nächsten Schritt muss der Komponentenkäufer seine eigenen Leistungsanforderungen mit den quantifizierten Metriken der Qualitätsbeschreibung vergleichen, die nunmehr an das einzusetzende System adaptiert wurden.

Eines der größten Probleme innerhalb dieses Marktmechanismus ist die korrekte und umfassende Determinierung und Quantifizierung der Leistungsgrößen. Der Komponenten-Provider könnte die Ergebnisse zu seinem Vorteil anpassen. Deswegen sollte die Determinierung und Quantifizierung einer vertrauenswürdigen „dritten Instanz“ (Trustworthy third Party) übertragen werden, wie es bereits bei Benchmark-Ergebnissen mit Hilfe internationaler Organisationen erfolgreich praktiziert wurde (vgl. Bild 2).

5 Strukturierte Performance-Analyse

Im Folgenden werden die Aufgabenstellungen für die im letzten Absatz des vorigen Abschnitts eingeführte „dritte Instanz“ für die Performance-Spezifizierung konkretisiert. Für die Beschreibung der Performance-Charakteristik einer Softwarekomponente werden aus unserer Sicht folgende Informationen benötigt (vgl. Bild 3):

- *Standardlastprofil:* Die durch die Komponente angebotenen Kernfunktionen sind hinsichtlich ihrer Aufrufhäufigkeit und der dabei übergebenen Datenmengen zu charakterisieren. Hierbei sollte primär die typische Verwendung einer Komponente berücksichtigt werden.
- *Ressourcenverbrauch:* Unter dem gegebenen Lastprofil ist der Ressourcenverbrauch sowohl von anderen softwaretechnischen Komponenten bzw. Services als auch hinsichtlich der in Anspruch genommenen Hardwareressourcen (z.B. CPU-, I/O-, Speicher) festzulegen.
- *Performance-Verhalten:* Das zeitliche Verhalten der im Lastprofil festgelegten Kernfunktionen der Komponente (Antwortzeit und Durchsatz) ist zu dokumentieren. Darüber hinaus sind Angaben über das interoperable Verhalten der Komponente zu anderen Softwarekomponenten und Services darzustellen.

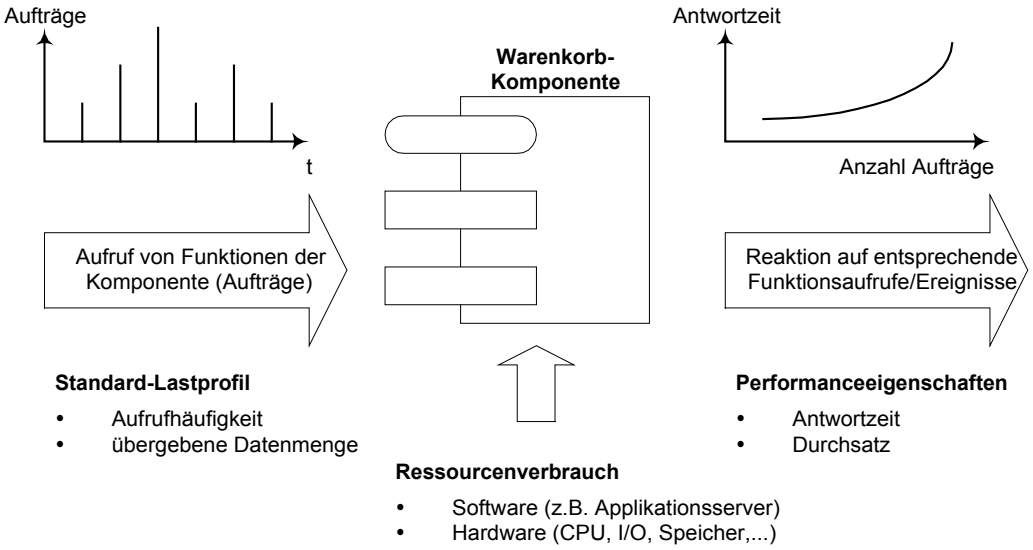

Bild 3: Elemente der Performance-Beschreibung

Für die Ermittlung dieser Angaben ist die Ausführung von Benchmarks erforderlich, wie unter (Sametinger 1997) kurz angeregt. Ähnlich der Beschreibung elektronischer Baugruppen werden dabei für eine konkrete Softwarekomponente umfangreiche Kennlinienfelder ermittelt und innerhalb entsprechender Datenblätter zur Verfügung gestellt. Für eine effiziente Benchmark-Ausführung sollten Standardtestumgebungen und Lasttreiber verwendet werden, die wiederholende Funktionsaufrufe eines zu definierenden bzw. zu konfigurierenden Lastmixes erlauben. Methodisch können die folgenden Schritte unterschieden werden:

1. Entwicklung eines Lastprofils, d.h. Identifizierung der wesentlichen durch die Komponenten angebotenen Funktionen (Methoden) und deren Aufrufverhalten.

2. Bereitstellung einer Laufzeitumgebung für die Komponenten, u.U. Installation weiterer notwendiger Komponenten und Services im Rahmen der Testumgebung.
3. Wenn die Komponenten auf einen Datenbestand zugreifen, sind entsprechende Testdaten bereitzustellen.
4. Atomare Ausführung der zu untersuchenden Funktionen der Komponenten und Aufzeichnung der Aufrufe durch einen Monitor
5. Sukzessive Modifikation der aufgezeichneten Funktionsaufrufe, so dass unterschiedliche Datenbereiche bei der Ausführung genutzt werden.
6. Konfiguration des Lasttreibers entsprechend dem für die Komponente ermittelten relevanten Lastprofil.
7. Ausführung des Benchmarks und Aufnahme von internen und externen Performance-Metriken.
8. Festhalten der Ergebnisse in entsprechenden Datenblättern zur Beschreibung der Performance-Eigenschaften der vermessenen Komponente.

Bei dieser Vorgehensweise gestaltet sich die Determinierung des Ressourcenbedarfs problematisch. Erfolgt ein Bezug auf direkte Hardwareeigenschaften des gewählten Referenzsystems, wie z.B. den CPU- oder Speicherverbrauch, sind die Performance-Aussagen nur für die im Test verwendete Hardware gültig. Eine Übertragbarkeit auf andere Systeme ist damit nur schwer möglich. Eine bedingte Alternative stellt der Bezug auf Messgrößen dar, die beispielsweise durch Standard-Benchmarks (z.B. SPEC jbb2000 - Java Business Benchmark) ermittelt werden.

6 Abbildung von Qualitätseigenschaften am Beispiel von EJB‘s

Am Beispiel der Verwendung von Enterprise Java Beans (EJB) soll die direkte Integration von Qualitätseigenschaften konzeptionell verdeutlicht werden. Die Spezifikation der EJBs berücksichtigt durch das Rollenkonzept explizit die Möglichkeit einer arbeitsteiligen Softwareentwicklung. Derzeit bietet die EJB-Spezifikation aber keine Aussagen zur Abbildung/Gewährleistung von Qualitätseigenschaften von EJB-Komponenten.

Aus Sicht der Komponentenentwickler bestehen EJBs mindestens aus einer Java-Klasse für die Realisierung der fachlichen Funktionalität, einer Interface-Klasse für den Zugriff auf gebotene Funktionen und einem Home Interface zur Verwaltung des EJB-Lebenszyklusses. Die EJBs werden in Java Archive Files weitergegeben, die alle Klassen inklusive einer Manifest-Datei zusammenfassen und darüber hinaus einen sogenannten Deployment Descriptor (DD) enthalten.

Durch einen auf der Basis der Extended Markup Language (XML) beschriebenen DD soll eine Anpassung entsprechender Attribute der EJBs ermöglicht und ein „Blackbox Reuse" unterstützt werden. Der DD kann als Beilage einer EJB oder einer aus mehreren EJBs zusammengesetzten Applikation betrachtet werden.

Auf Grund der einfachen Erweiterbarkeit des DD kann ein `<quality>`-Tag eingeführt werden, der entsprechende Informationen zu den für die Komponente spezifizierten Qualitätsklassen enthält. Beispielhaft wurde die CPU des für die Performancespezifizierung verwendeten Referenzsystems innerhalb des `<RS>`-Tag angegeben. Da die Funktionalitäten einer EJB-Komponenten immer über den Aufruf konkreter Methoden angesprochen werden,

muß diese als erstes über den **<METHOD>**-Tag qualifiziert werden. Angaben zu den Performance-Eigenschaften der Methode „Kunde_hinzufügen“ finden sich jetzt innerhalb des **<Q3>**-Tags. Innerhalb des **<Q31>**-Tags findet sich so z.B. die maximale Antwortzeit der Methode (vgl. Bild 4).

```
<?xml version="1.0" encoding="Cp1252"?>

<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise
    JavaBeans 1.1//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd'>

<ejb-jar>
  <description>no description</description>
  <display-name>Ejb1</display-name>
  <enterprise-beans>
    <session>
            <display-name>CartEJB</display-name>
            <ejb-name>CartEJB</ejb-name>
            <home>CartHome</home>
            <remote>CartClient</remote>
            <ejb-class>CartEJB</ejb-class>
            <session-type>Stateful</session-type>
            <transaction-type>Container</transaction-type>
    </session>
    <quality>
         <RS>CPU: Risc P5C-233 SPEC:156</RS>
         <METHOD NAME=”Kunde_hinzufügen”>
            <Q1>...</Q1>
            <Q2>...</Q2>
            <Q3>
              <Q31>Response_Time: 1.5s</Q31>
              <Q32>CPU_%: 70%</Q32>
              <Q33>...</Q33>
            </Q3>
            <Q4>...</Q4>
            <Q5>...</Q5>
            <Q6>...</Q6>
         </METHOD>
    </quality>
  </enterprise-beans>
</ejb-jar>
```

Bild 4: Erweiterter XML Deployment Descriptor

Die innerhalb des **<Q3>**-Tag abgelegten Informationen, die im Rahmen des für EJBs typischen Deploy-Vorgangs (Einbringen der EJBs in entsprechende Container unter Verwendung von Wizards) spezifiziert werden, können so unter anderem für die Auslösung konkreter Aktivitäten z.B. bei Überschreitung der vorgegebenen Antwortzeiten herangezogen werden. Wenn der zur Ausführung genutzte Container sowie der Applikationsserver diese Information interpretieren kann, werden die folgenden Funktionen ermöglicht:

- Überwachung definierter Service Level Agreements,
- Accounting in Abhängigkeit der erbrachten Leistung,
- Unterstützung eines Load Balancing.

Uns ist bewußt, dass derzeit kein Applikationsserver einen derartigen erweiterten Quality Descriptor unterstützt. Auch sind die letztendlich innerhalb des `<quality>`-Tag festzulegenden Einträge in Zusammenarbeit mit der Firma Sun weiter zu formalisieren, um eindeutige Auswertungen zu ermöglichen. Die hier beschriebene Strukturierung soll daher lediglich ein mögliches Modell für die Berücksichtigung der Qualitätseigenschaften von Komponenten vorschlagen.

7 Zusammenfassung und Ausblick

Der vorliegende Beitrag gibt einen Einblick in die komplexe Themenstellung der Performance-Spezifizierung von Softwarekomponenten. Das vorgeschlagene Konzept der Performance-Spezifikation und -Analyse auf der Basis spezifischer Benchmarks bietet eine methodische Vorgehensweise zur Determinierung und Distribution des Performance-Verhaltens einer Softwarekomponente. Mit Hilfe der Qualitätsklassen kann auf unterschiedliche Anforderungen der Komponentennutzer reagiert werden, da jede Komponente differenzierte Anforderungen an die Determinierung qualitätsrelevanter Eigenschaften aufweist.

Dem Entwickler steht mit der Darstellung der Performance-Eigenschaften ein weiteres Auswahlkriterium zur Verfügung, das eine performance-orientierte Softwareentwicklung auf der Basis von Komponenten erst ermöglicht.

In einem nächsten Schritt soll das in diesem Beitrag vorgestellte Konzept explizit auf die JavaBean- bzw. EJB-Technologie bezogen und prototypisch implementiert werden. Darüber hinaus soll eine dynamische Reaktion der Komponenten auf Performance-Probleme integriert werden, so dass die komplette Interaktionskette einer Anwendungsfunktion aus Endbenutzersicht erfaßt werden kann. Dafür ist es erforderlich, Komponenten während der Entwicklung zu instrumentieren. Das von der Computer Measurement Group standardisierte "Application Response Measurement APIs" kann dabei für C++- und Java-Komponenten verwendet werden. Über den hier vorgeschlagenen erweiterten Deployment Descriptor soll ein direktes „Hineingenerieren“ dieser Meßpunkte in den Quellcode der Komponenten erreicht werden.

Literaturverzeichnis

Dumke, R.: Software Engineering. Eine Einführung für Informatiker und Ingenieure. Vieweg-Verlag, Braunschweig/Wiesbaden, 2000

Klein, M.: State of the Practice Report. Problems in the Practice of Performance Engineering, Technical Report, No. SEI-95-TR-020, Software Engineering Institute, Pittsburgh, 1996

Meritt, S.: Reuse library. In: Marciniak: Encyclopedia of Software Engineering, p. 1069-1071, John Wiley & Sons, 1994

Sametinger, J.: Software Engineering with Reusable Components. Springer-Verlag, Berlin Heidelberg New York, 1997

Schmietendorf, A., Dumke, R.: Metrikenbasierte Bewertung von Software-Komponenten. In: Tagungsband zur Conquest 2000, Arbeitskreis Software-Qualität Franken e.V., Nürnberg, 2000

Scholz A.: Performance Specifications of Software Components. In: Conference Guide IRMA 2000, p. 879

Scholz, A., Turowski, K.: Service Level Agreements of Performance Requirements. World Class IT Service Management Guide. J. Bon. Amsterdam, ten Hagen & Stam: p. 249-256.

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