Spezifikation von Softwarekomponenten auf Qualitätsebene

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

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

\# *T-Systems/T-Nova, Entwicklungszentrum Berlin, Kompetenzgruppe für System- und Technoogieentwicklung, Wittestr. 30H, D-13509 Berlin, Email: a.schmietendorf@telekom.de*

Zusammenfassung: Die Idee, Softwaresysteme auf der Basis von Komponenten zu entwickeln, wurde bereits im Jahr 1969 durch Mc Ilroy formuliert. Ein komponentenbasiertes Vorgehen im Rahmen der Softwareentwicklung soll die eigentliche Entwicklungszeit drastisch verkürzen, eine höhere Produktivität gewährleisten und durch den Einsatz normierter und getesteter „Bauteile" zur Verbesserung der Produktqualität beitragen. Derzeitige industriell verwendete Komponentenansätze negieren weitgehend den Aspekt der Qualitätsbeschreibung einer konkreten Komponente, weshalb insbesondere auf diesem Gebiet umfangreicher Forschungsbedarf besteht. Im Rahmen dieses Beitrags wird die Themenstellung einer granularen Qualitätsspezifikation von Softwarekomponenten aufgegriffen. Dafür wird ausgehend vom generischen Qualitätsmodell der ISO 9126 die Ableitung eines komponentenspezifischen Qualitätsmodells unter Verwendung der GQM-Methode vorgeschlagen. Zur praktischen Umsetzung der Spezifikation wird zum einem die Verwendung einer modellbasierten Notation auf Basis der UML/OCL vorgeschlagen, zum anderen für die Operationalisierung konkreter Qualitätskriterien Möglichkeiten der Verwendung von Software-Metriken aufgezeigt.

Schlüsselworte: Komponenten, UML/OCL, Qualität, Qualitätsklassen, Metriken

1 Einführung

Die industrielle Produktion zeichnet sich durch einen hohen Grad an Wiederverwendung bereits erzeugter Zwischenprodukte (Halbfabrikate) aus. Insbesondere wird dadurch eine Verbesserung der Qualität, der Produktbereitstellungszeiten und der Produktivität erreicht. Grundlage eines derartigen Vorgehens sind ein arbeitsteiliger Prozeß zwischen „Zulieferer" (Fertigung von Komponenten) und „Endmontage" (Zusammensetzen der Komponenten) sowie Standards hinsichtlich der funktionalen und qualitativen Eigenschaften der eingesetzten Zwischenprodukte, um die kundenseitigen Anforderungen an das endgültige Produkt sowohl funktional als auch qualitativ erfüllen zu können.

Auch die Softwareentwicklung möchte dieses in allen ingenieurwissenschaftlich begründeten Industriezweigen erfolgreich angewandte Vorgehen durch den Einsatz der komponentenorientierten Softwareentwicklung adaptieren. Der Einsatz von Softwarekomponenten impliziert die Möglichkeit eines arbeitsteiligen Prozesses, so dass zumindest zwischen Herstellern und Verbrauchern von Komponenten unterschieden werden kann. Diese Producer-Consumer-Sicht erfordert die Beschreibung der von der Komponente erbrachten funktionalen und qualitativen Eigenschaften durch den Komponentenentwickler, die den Verbraucher in die Lage

versetzt, Software-Komponenten so einzusetzen, dass das Gesamtsystem den Anforderungen der späteren Benutzer gerecht wird.

Waren es zu Beginn der komponentenorientierten Softwareentwicklung überwiegend informationstechnische Komponenten (z.B. für die grafische Präsentation), so kann derzeit die Konzentration auf die eigentliche Fachlogik (z.B. Komponenten der Finanzwirtschaft) beobachtet werden. Dafür industriell einsetzbare Komponentenmodelle, wie z.B. die Enterprise Java Beans (EJB), das Microsoft Component Object Model (COM) oder auch die mit dem CORBA-3-Standard avisierten CORBA-Komponenten, konzentrieren sich insbesondere auf funktionale Aspekte, lassen aber nicht funktionale Eigenschaften (in Zukunft wollen wir diese allgemein als qualitative Eigenschaften bezeichnen) weitgehend außer Acht.

Durch den im Rahmen des GI-Arbeitskreises 5.10.3 bearbeiteten Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten wird das Ziel verfolgt, Fachkomponenten widerspruchsfrei und eindeutig hinsichtlich ihrer funktionalen und qualitativen Eigenschaften aus einer „Black Box“-Perspektive zu beschreiben. Da nicht davon ausgegangen werden kann, dass beim Konsumenten einer solchen fachlichen Komponente (häufig wird in diesem Zusammenhang von sogenannten COTS[1]-Komponenten gesprochen) der entsprechende Quellcode zu Verfügung steht, führt dies zur speziellen Problematik der Komponenten-Akquirierung und dem dabei vorhandenen Risiko, qualitativ unzureichende Komponenten innerhalb des eigentlichen Produktes einzusetzen. Ein Qualitätssicherung hat deshalb sowohl beim Komponentenproduzenten als auch beim Komponentenkonsumenten im Sinne einer Wareneingangskontrolle zu erfolgen.

2 Qualitätseigenschaften von Softwarekomponenten

2.1 Ziel der Qualitätsebene

Durch [Turowski 1999] wurden 6 Spezifikationsebenen (Administrationsebene, Syntaxebene, Verhaltensebene, Abstimmungsebene, Qualitätsebene, Domänenebene) zur Beschreibung der Dienste einer Fachkomponente vorgeschlagen, in deren Rahmen die Qualitätsebene die nichtfunktionalen Eigenschaften einer Komponente ausgehend von einer Außensicht erfassen soll. Typische Beispiele von durch diese Spezifikationsebene zu erfassenden Eigenschaften beziehen sich auf die Verfügbarkeit, die Effizienz, die Portabilität oder auch die Wartbarkeit.

Diese Eigenschaften sollten zum einem bei Bedarf durch geeignete Notationen in entsprechenden Modellen des späteren Informationssystems erfaßt werden können. Zum anderen sollten am Markt erhältliche Komponenten derartige Informationen im Rahmen von selbstbeschreibenden Schnittstellen vorhalten. Nicht-funktionale Eigenschaften können zumeist nicht als der betrachteten Komponente inhärente Eigenschaft betrachtet werden. Erst im Rahmen der Interaktion von in die Komponente eingehenden Aufträgen sowie der Nutzung externer Hard- und Softwareressourcen durch die Komponente selbst können diese Eigenschaften sinnvoll determiniert werden, was zum Bedarf einer entsprechenden Referenzumgebung führt.

[1] commercial of the shelf components

Die letztlich verwendete Spezifikation muß neben den Qualitätseigenschaften der betrachteten Funktionen ebenfalls die an die Fachkomponente gestellte Anforderung und die durch die Komponente verbrauchten Ressourcen berücksichtigen. In diesem Zusammenhang sind entsprechend einzusetzende Methoden zur Ermittlung statistischer als auch dynamischer Qualitätseigenschaften festzulegen. Diese sollten sowohl beim Produzenten einer Fachkomponente als auch im Rahmen einer speziellen Qualitätskontrolle beim Anwender der Komponente durchgeführt werden. Für speziell die anwenderseitige Verifizierung von Qualitätseigenschaften hat sich die Aufgabenstellung des CURE (COTS usage risk evaluation) herausgebildet. Weitere Informationen dazu finden sich z.B. unter [SEI 2001].

Folgende Abbildung zeigt zusammenfassend die für die Spezifikation der Qualitätseigenschaften einer Komponente notwendigen Schritte, auf die in den folgenden Abschnitten vertiefend eingegangen werden soll:

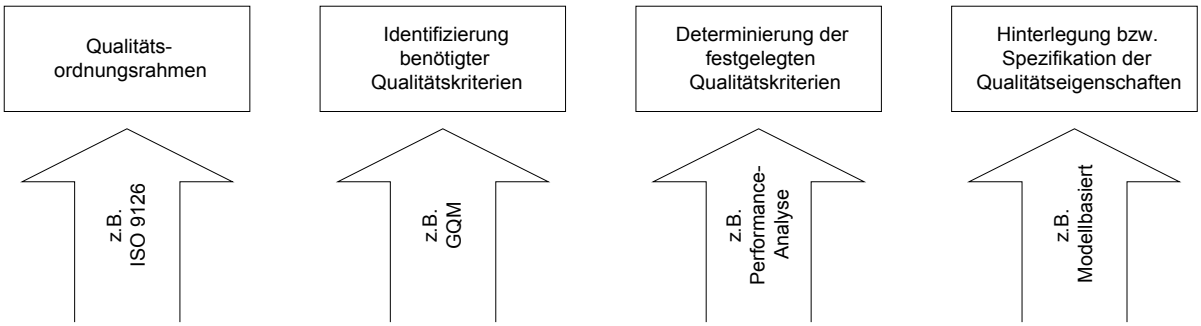

Abbildung 1: Spezifikationsschritte einer Fachkomponente

2.2 Einführung von Qualitätsklassen als Ordnungsrahmen

Der Begriff der Qualität eines Softwareproduktes im allgemein als auch im besonderen in Bezug auf Komponenten führt zu vielfältigen Interpretationsmöglichkeiten. Dementsprechend ist es notwendig, ein konkretes Qualitätsmodell festzulegen, um den Qualitätsbegriff zu operationalisieren. Für diese Aufgabenstellung haben sich sogenannte FCM[2]-Modelle (siehe dazu auch Darstellungen in [Balzert 1998]) herausgebildet, welche ausgehend von Qualitätsfaktoren entsprechende Qualitätskriterien festlegen und für deren Quantifizierung entsprechende Metriken vorschlagen.

Mit der ISO 9126 findet sich ein Standard für ein Qualitätsmodell entsprechend dem FCM-Ansatz. Im folgenden soll dieser als Orientierungshilfe herangezogen werden, um Qualitätsklassen zur Bewertung von Komponenten zu bilden. Mit der Einführung der folgenden Qualitätsklassen (siehe auch [Schmietendorf/Scholz 2000]) im Rahmen der Spezifikation von Softwarekomponenten ergibt sich die Möglichkeit einer granularen Berücksichtigung von Qualitätseigenschaften konkreter Komponenten.

- *Übertragbarkeit Q1:* Anpassbarkeit, Installierbarkeit;
- *Anwendbarkeit Q2:* Erlernbarkeit, Beherrschbarkeit, Verständlichkeit;

[2] factor criteria metrics model

- *Effizienz Q3:* Raumbezogen, Zeitbezogen, Ressourcenbezogen;
- *Funktionalität Q4:* Angemessenheit, Interoperabilität, Genauigkeit;
- *Zuverlässigkeit Q5:* Fehlertoleranz, Fehlerhäufigkeit, Fehlerverfolgbarkeit;
- *Wartbarkeit Q6:* Analysierbarkeit, Änderbarkeit, Stabilität, Testbarkeit.

Eine konkrete Komponenten kann so z.B. die Qualitätseigenschaften Q1 und Q3 aufweisen, d.h. innerhalb der Spezifikation befinden sich Aussagen zu genau diesen Qualitätseigenschaften, nicht aber zu allen weiteren. Wie letztendlich die konkrete Ausprägung der Spezifikation in Bezug auf genau eine Qualitätsklasse erfolgt, soll in einem weiteren Schritt mit Hilfe der GQM-Methode geklärt werden.

2.3 Komponentenspezifische Qualitätsmodelle

Für die eigentliche Identifizierung der für eine konkrete Komponente zu spezifizierenden Qualitätseigenschaften kann z.B. das GQM-Pardigma (Goal Question Metric) verwendet werden [Solingen/Berghout 1999]. Dieses bietet eine methodische Vorgehensweise zur Erstellung eines entwicklungsspezifischen Qualitätsmodells und kann so auch für die Komponententechnologie herangezogen werden. Dafür führt die GQM-Methode, ausgehend von zu formulierenden Qualitätszielen (im speziellen Fall der Qualität können diese der ISO 9126 entnommen werden) und den Fragen, wie diese erreicht werden können, zu den für die Quantifizierung bzw. Beantwortung der Fragen notwendigen Meßgrößen. Für die Beantwortung der identifizierten Fragestellungen werden typischerweise die folgenden Erfolgskriterien in Anlehnung an [Dumke 2001] herangezogen:

- Sichtweise: Komponentenentwickler, Komponentenanwender, Auftraggeber,...
- Anwendungsbereich: spezielles Projekt bzw. ausgewählte Produktklasse
- Zweck: Analyse, Verständnis,...
- Kontext: z.B. im Rahmen eines ausgewählten Entwicklungsteams.

Darüber hinaus sind im Rahmen des GQM-Ansatzes Aufgabenstellungen der effizienten Meßwerterfassung, der Ergebnisinterpretation und der Validation durchzuführen (siehe dazu z.B. [Schmietendorf 2000]). Auf diese Themenstellung soll im Rahmen dieses Beitrags allerdings nicht weiter eingegangen werden.

Für speziell die Qualitätsklasse Q3 (Effizienz) ergibt sich so die folgende Ausprägung:

Goal (Ziel)

> Das determinierte Performanceverhalten der durch eine Komponente angebotenen Funktionen ist für deren erfolgreichen Einsatz von entscheidender Bedeutung. (z.B. bei Komponenten im Bereich von Telekommunikationsanwendungen).

Question (Frage)

> Welche Kenngrößen werden zur Erfassung des zeit- und raumbezogenen Performanceverhaltens benötigt?

Metric (Meßgröße)

Für die Erfassung des nach außen sichtbaren Performanceverhaltens werden die Größen Antwortzeit und Durchsatz einer konkreten Funktion der Komponente herangezogen.

Darüber hinaus besteht die Notwendigkeit, den für die Erbringung dieser Größen benötigten Ressourcenverbrauch (CPU, I/O, Softwareservices) festzulegen, das entsprechende Lastprofil (Auftragsarten sind im zeitlichen Verlauf zu berücksichtigen), die verwendete Hard- und Softwarearchitektur und die Leistungseigenschaften der betrachteten Gesamtarchitektur (Netzwerke, Rechner, Services,...) zu erfassen, was der bereits angesprochenen Referenzumgebung entspricht.

Für speziell die Qualitätsklasse Q3 wurde die Vorgehensweise zur Ermittlung der vorgenannten Meßgrößen unter [Schmietendorf/Scholz 2000] bereits dargestellt.

3 Spezifikation von Qualitätsmerkmalen

3.1 Determinierung von Qualitätskriterien

Die Determinierung eines konkreten Qualitätskriteriums erfordert den Einsatz einer Methode zur Ermittlung der entsprechenden Eigenschaften. Unter [Schmietendorf/Scholz 2000] wurde für die Qualitätsklasse Q3 beispielhaft die Durchführung von Benchmarktests vorgeschlagen. Im Rahmen eines solchen Tests wird die Komponente innerhalb einer Referenzumgebung, bestehend aus Hard- und Software, zur Ausführung gebracht und die Funktionen der Komponente durch Auftragseingänge belastet. Dafür ist ein entsprechendes Auftragsprofil festzulegen, welches die Auftragsarten, das zeitliche Auftreten von Aufträgen, die Verhältnisse der Auftragsarten zueinander und die mit jedem Auftrag übergebene Datenmenge berücksichtigt. Darüber hinaus sind notwendige Datenbestände der Gesamtapplikation zu erzeugen und mögliche Konfigurationen der Komponenten festzulegen.

Für die konkrete Abbildung der zu spezifizierenden Qualitätseigenschaften einer Fachkomponente sollten sowohl modellbasierte Notationen, wie die UML[3] bzw. die darin enthaltene OCL[4], als auch Softwaremetriken verwendet werden.

3.2 Modellbasierte Notationen

Sollen Qualitätseigenschaften einer Komponente im Rahmen der UML-Notation festgehalten werden, bieten sich dafür insbesondere die Interaktions-, Zustands-, Paket- und Verteilungsdiagramme an. Um diese Diagramme für die Qualitätsebene zu verwenden sind die nativen UML-Erweiterungsmöglichkeiten (Stereotypes, Annotationen, Constraints, Tagged Values) heranzuziehen. Entsprechende Vorschläge zur Verwendung der UML-Notation im Kontext mit dem Effizienzverhalten (Qualitätsklasse Q3) von Software-Artefakten finden sich unter [Schmietendorf/Dimitrov 2001].

---

[3] UML Unified Modeling Language
[4] OCL Object Constraint Language

Im folgenden sollen die Möglichkeiten dieser Notation in Bezug auf die Spezifikation von Komponenten skizziert werden, im Anschluß zeigen ein Interaktions- und Verteilungsdiagramm beispielhaft die konkrete Ausprägung.

- *Interaktionsdiagramm*: erlaubt die Darstellung von Interaktionen zwischen Komponenten-Instanzen. In deren Rahmen kann zum einem das Auftragsprofil, das zeitliche Verhalten von Komponenten-Methoden und der Ressourcenbedarf von Komponenten-Instanzen mittels Constraints, Tagged Values und Annotationen festgehalten werden.

- *Zustandsdiagramm:* erlaubt z.B. die Erfassung des Lebenszyklus einer Komponente, wofür Zustände (Knoten) und Zustandsübergänge (Kanten) mit entsprechenden Übergangs-Bedingungen verwendet werden. Dementsprechend bietet sich mit diesem Modell eine direkte Beziehung zur Markov'schen Modellierung, welche für die modellbasierte Performanceanalyse einer Komponente herangezogen werden kann.

- *Paketdiagramm:* erlaubt die atomare Darstellung der Komponente selbst. Durch sowohl Annotationen als auch Constraints lassen sich das abstrakte Ressourcenverhalten sowie statische Eigenschaften spezifizieren, wie z.B. Granularität und Kopplungsgrade einer Komponente.

- *Verteilungsdiagramm:* bietet den direkten Bezug zu Hard- und Softwareressourcen der Laufzeitumgebung, sofern explizite Angaben zu den benötigten Ressourcen mittels Annotationen und Sterotypen möglich sind.

Abbildung 2 zeigt die Spezifikation von zeitlichen Vorgaben (Ausführungszeiten und Latenzzeiten im Rahmen eines UML-Interaktionsdiagramms.

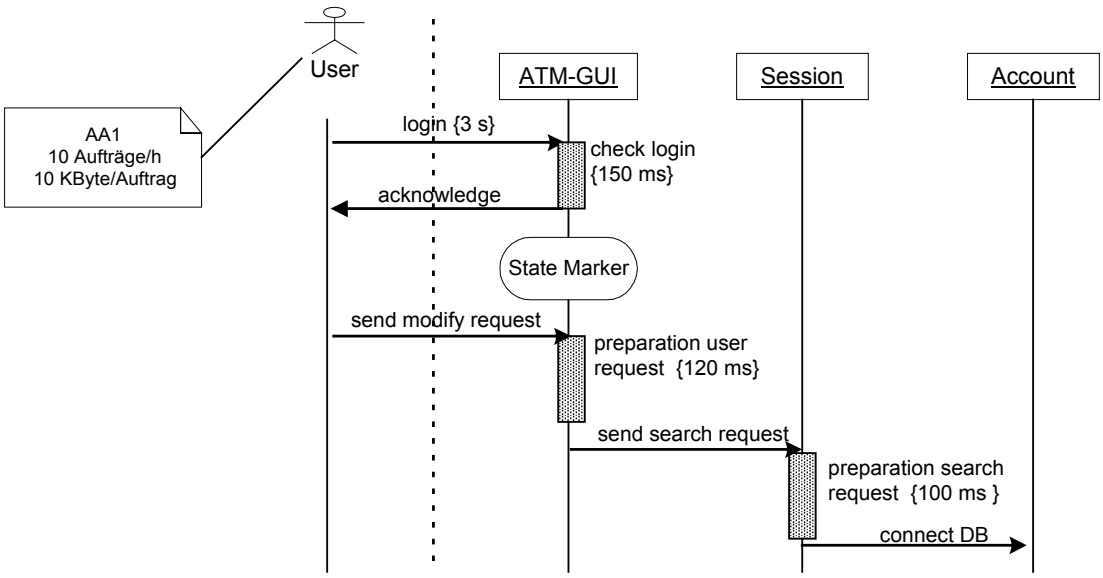

Abbildung 2: Spezifikation im Rahmen von Sequenzdiagrammen

Abbildung 3 zeigt die vorgeschlagene Vorgehensweise unter Zuhilfenahme restriktiver Stereotypen und Annotationen im Rahmen von Verteilungsdiagrammen:

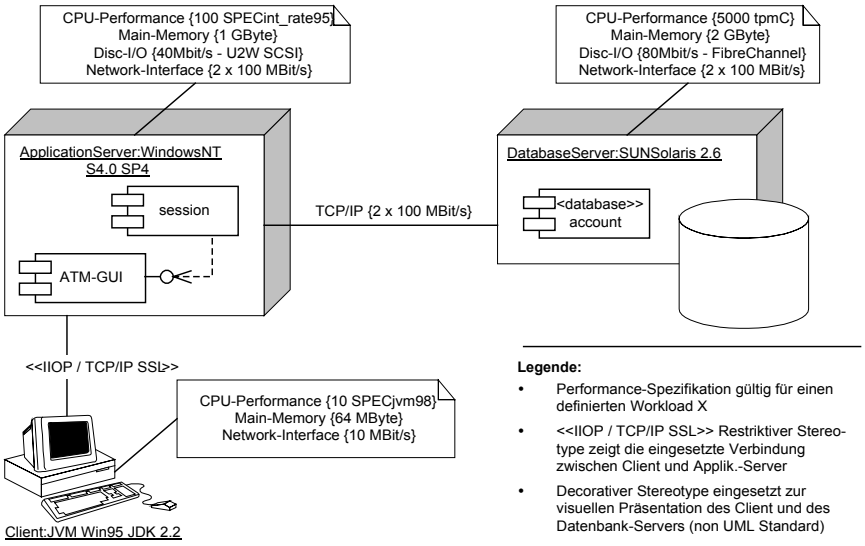

Abbildung 3: erweiterte UML-Verteilungsdiagramme

Die Object Constraint Language OCL bietet die Möglichkeit, Rahmenbedingungen bzw. Voraussetzungen, die eine Komponente zur Erbringung ihrer geforderten funktionalen und qualitativen Eigenschaften benötigt, festzulegen.

Im folgenden sei beispielhaft der Ressourcenbedarf einer konkreten Methode an I/O-Operationen mittels der OCL-Notation dargestellt:

```
context Komponente inv I/O-Bedarf
      self.show() < 100 NW-I/O
```

Die OCL bietet darüber hinaus die Möglichkeiten, Performanceanforderungen im Sinne von Antwortzeit- und Durchsatz-Schranken zu formulieren. Sinnvoll ist es, diese aus dem Modell direkt in die betroffene Komponenten zu überführen, wobei im speziellen Fall der EJB-Komponenten eine Abbildung im Rahmen des sogenannten XML-Deployment-Deskriptor vorgenommen werden kann. Diese Informationen können so insbesondere zur Festlegung von SLA‘s (Service Level Agreement) bzw. SLO‘s (Service Level Objectives) im Rahmen des Wirkbetriebs herangezogen werden können. Inwieweit die Vereinbarung komponentenbezogener SLA’s sinnvoll ist, hängt zum einem von der Granularität der betrachteten Komponente, dem mit einem umperformanten Funktionsverhalten einhergehenden Risiko und den ggf. erforderlichen dynamischen Reaktionen (z.B. Loadbalancing) auf Performanceengpässe ab.

3.3 Metrikenbasierte Spezifikation von Qualitätsmerkmalen

Eine weitere Möglichkeit zur Erfassung qualitativer Eigenschaften einer Komponente besteht in der allgemeinen Verwendung von Softwaremetriken (über Performancemetriken hinausgehend), die qualitative Eigenschaften einer Komponente widerspiegeln.

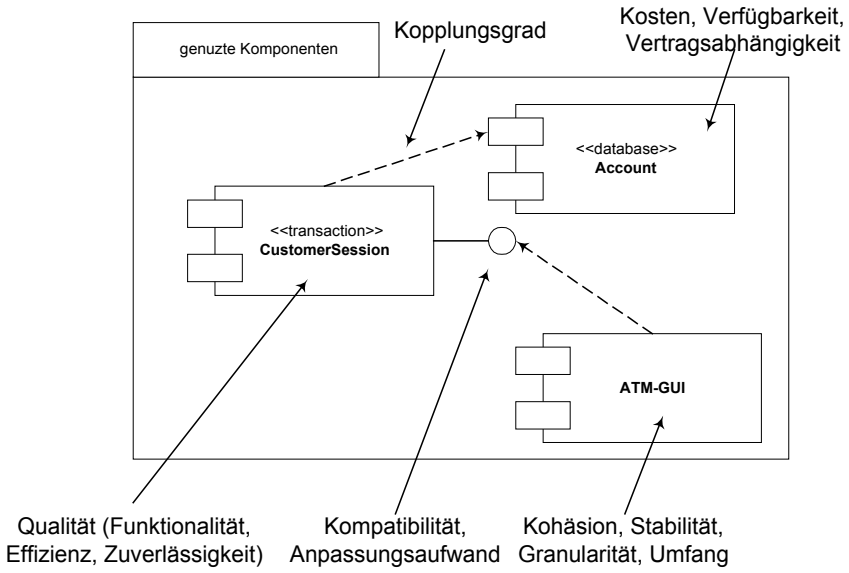

Abbildung 4: Messansätze im Rahmen einer Komponentenarchitektur [Dumke 2001]

Auf der Basis empirisch gewonnener Schwellwerte (Metriken) für „qualitativ hochwertige" Softwarekomponenten lassen sich von seiten des Komponentenentwicklers Komponenteneigenschaften durch den Einsatz von Programmierrichtlinien verifizieren bzw. im Rahmen der Beschreibung für den späteren Komponentenkonsumenten festhalten. Die dabei verwendeten Meßgrößen können ein unterschiedliches Skalenniveau (nominal-, ordinial-, intervall-, ratioskaliert) aufweisen. Ein höheres Skalenniveau impliziert sowohl einen größeren Informationsgehalt dieser Metriken als auch die Möglichkeit, höherwertige statistische Operationen über diesen anzuwenden.

4 Praktische Meßansätze von Komponentenarchitekturen

4.1 Zielstellungen der durchgeführten Studien

Im Rahmen von am Entwicklungszentrum Berlin der T-Nova durchgeführten Studien wurden zwei Zielstellungen der Analyse von Komponenten verfolgt, um entsprechende Meßgrößen zu Komponenten zu gewinnen:

1. Die Analyse statischer Eigenschaften von Komponenten unter Zuhilfenahme entsprechender Softwaremetriken (siehe dazu [Schmietendorf/Dumke 2000] bzw. [Lezius 2001]),
2. Die Analyse dynamischer Eigenschaften, wobei speziell das Performanceverhalten untersucht wurde [Nakonz 2001].

Im folgenden sollen einige Eckpunkte dieser Untersuchungen aufgezeigt werden, eine ausführliche Darstellung der Ergebnisse ist im Rahmen eines solchen Beitrages selbstverständlich nicht möglich.

4.2 Analyse statischer Komponenteneigenschaften

Die Analyse statischer Komponenteneigenschaften auf der Basis der Vermessung von ca. 50 Java-Komponenten (sowohl JavaBeans als auch EJB's) aus dem industriellen und akade-

mischen Umfeld führte zu einem ersten Entwurf einer entsprechenden Programmierrichtlinie, die sowohl durch Komponentenentwickler als auch Komponentenanwender verwendet werden kann. Der Aufbau dieser Richtlinie enthält eine Darstellung des Qualitätsmerkmals, das entsprechende Kriterium, einsetzbare Metriken und Schwellwerte zur Bewertung einer konkreten Größe. Als Beispiele seien in Bezug auf die Wiederverwendbarkeit folgende Metriken genannt:

- Komponentengröße (z.B. Klassen, eLoC),
- Kopplungsmaße (z.B. Fan In/Out),
- Abstraktion der Komponente (z.B. Vererbungstiefe),
- Dokumentation der Schnittstellen (z.B. Verhältnis eLoC zu Kommentaren),
- Komplexität der Komponenten (hier McCabe konkreter Methoden),
- Granularität (z.B. Anzahl zur Verfügung gestellter fachlicher Funktionen).

Ein anderer Input für die Einschätzung der Qualität kommerziell angebotener Komponenten (COTS - commercial of the shelf) besteht in der Verwendung indirekter Faktoren, die sich auf das Prozeß- und Ressourcenniveau der jeweiligen Komponentenhersteller beziehen. Als Beispiel für ein dafür verwendbares Prozeßmodell sei das CMM (Capability Maturity Model) genannt. Inwiefern dieses auch die Aspekte einer komponentenorientierten Softwareentwicklung berücksichtigt, kann derzeit nicht abschließend geklärt werden. Erste Anhaltspunkt dazu finden sich im CMMI unter [Ahern 2001].

4.3 Analyse der Performanceeigenschaften von Komponenten

Eine umfangreiche Untersuchung der Performanceeigenschaften von EJB-Komponenten, die am Entwicklungszentrum Berlin durchgeführt wurde, zeigte bereits vielfältige Problemstellungen auf. Beispielhaft seien der damit verbundene Aufwand, die erreichbare Genauigkeit bei der Aufnahme der Performance-Metriken, die Verallgemeinerung der Ergebnisse, die Parallelisierung der Zugriffe auf die Bean-Instanz oder auch die Granularität der untersuchten Komponenten selbst genannt. Im Detail konnten z.B. die folgenden performancerelevanten Erfahrungen gewonnen werden:

- Die Zeitdauer für die Instanziierung eines Entity-Bean hängt maßgeblich von dessen Größe ab; bei einer Session-Bean spielt die Größe der Bean-Klasse dagegen keine Rolle.

- Session-Beans benötigen eine wesentlich kürzere Initialisierungszeit im Vergleich zu Entity-Beans.

- Einen geringen Einfluß auf die Performance besitzen die dem Bean übergebenen Parameter und die Tatsache, ob es sich um einen schreibenden oder lesenden Zugriff handelt.

- Die umfangreiche Verwendung von Entity-Beans impliziert bei den derzeit verfügbaren Container- bzw. Applicationserver-Implementierungen Performancerisiken.

Die folgende Abbildung zeigt den zur Vermessung entwickelten Benchmark:

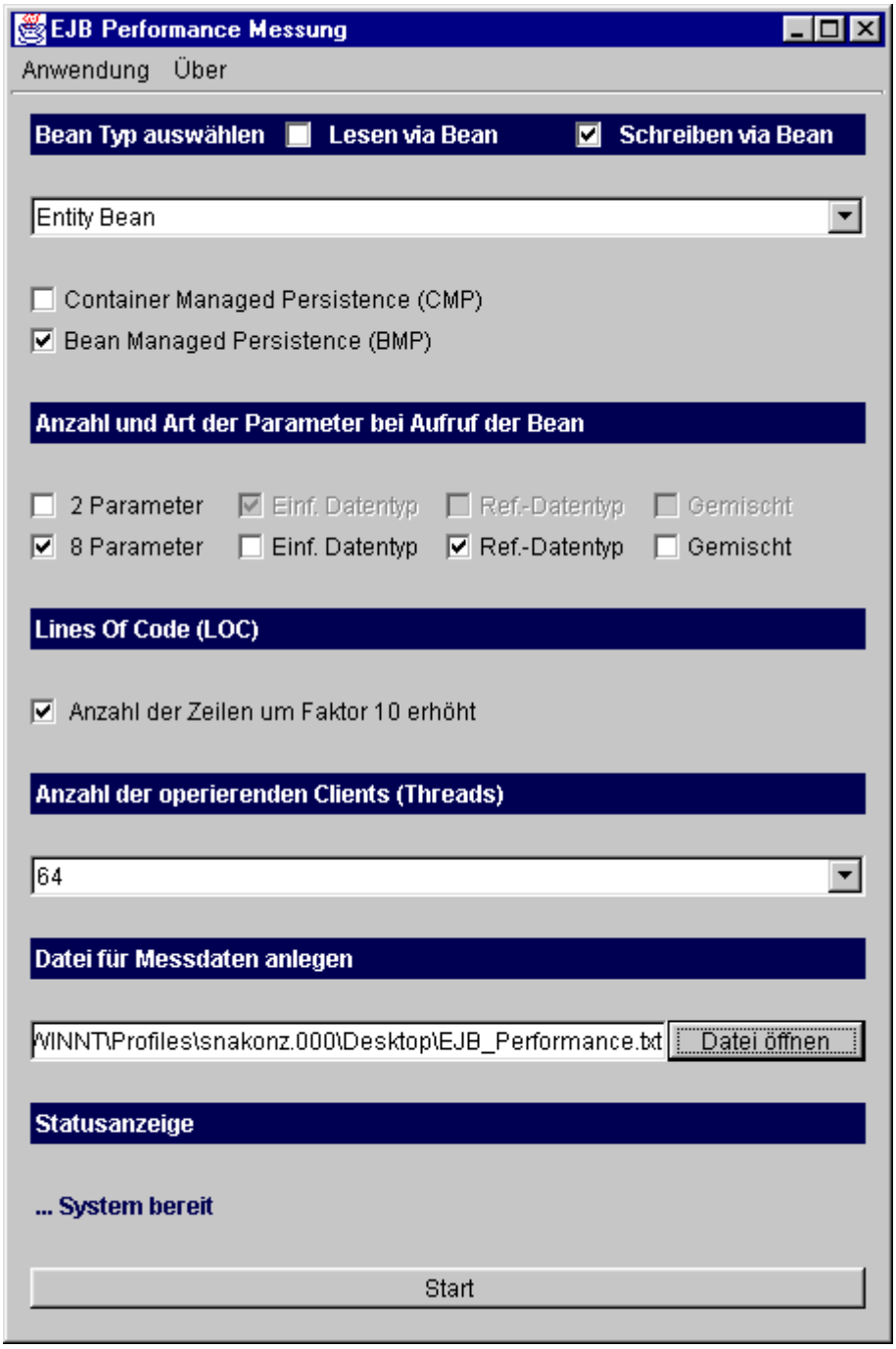

Abbildung 5: grafische Oberfläche des EJB-Benchmarks [Nakonz 2001]

5 Zusammenfassung

Es stellt sich die Frage, ob der mit der Determinierung und Spezifizierung von Qualitätseigenschaften einhergehende Aufwand für den kommerziellen Komponentenmarkt in der Praxis tatsächlich durchsetzbar ist. Der hier vorgestellte generische Ansatz der Qualitätsspezifikation bietet zwar eine praxistaugliche Vorgehensweise, welche insbesondere im Rahmen von fachspezifischen Komponentenframeworks Verwendung finden könnte, es bleibt jedoch abzuwarten ob die kommerziellen Randbedingngen dieses auch zulassen. Innerhalb des hier ange-

strebten Memorandums erscheint eine darüber hinaus gehende Standardisierung der Syntax und Semantik konkreter Qualitätsattribute aufgrund der potentiellen Vielfältigkeit nahezu unmöglich. Darüber hinaus bietet sich mit agentifizierten Komponenten die Möglichkeit, das reaktive Verhalten von Agenten für die Qualitätssicherung von Komponenten auszunutzen, wodurch der Mehrwert einer statischen Qualitätsspezifikation, wie hier vorgeschlagen, durchaus in Frage gestellt wird.

6 Quellenverzeichnis

[Ahern 2001] Ahern, D.M.; Clouse, A.; Turner, R.: CMMI Distilled. A Practical Introduction to Integrated Process Improvement. Addison-Wesley: Boston, San Francisco, New York, … 2001

[Balzert 1998] Balzert, H.: Lehrbuch der Software-Technik. Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum Akademischer Verlag GmbH: Heidelberg, Berlin, 1998

[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)

[Nakonz 2001] Nakonz, S.: Benchmarking verteilter Komponentensysteme. Diplomarbeit, Fachhochschule für Technik und Wirtschaft Berlin, 2001 (fachliche Betreuung am Entwicklungszentrum Berlin der T-Nova)

[Schmietendorf 2000] Schmietendorf, A.; Dimitrov, E.; Dumke, R.; Foltin, E.: Konzeption und erste Erfahrungen einer Metriken-basierten Software Wiederverwendung. In: R. Dumke, F. Lehner: Software-Metriken. Deutscher Universitäts-Verlag/Gabler Edition Wissenschaft: Wiesbaden, 2000, S. 95

[Schmietendorf/Dimitrov 2001] Schmietendorf, A; Dimitrov, E: Possibilities of performance modelling with UML. In R. Dumke et. al.: Performance Engineering. State of the art and current trends. LNCS 2047, Springer 2001

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

[Schmietendorf/Scholz 2000] Schmietendorf, A.; Scholz A.: Spezifikation der Performance - Eigenschaften von Softwarekomponenten. In: Proc. Modellierung und Spezifikation von Fachkomponenten, Workshop im Rahmen der MobIS 2000, Siegen, S. 41

[SEI 2001] COTS Usage Risk Evaluation, http://www.sei.cmu.edu/cbs/cure-one-pager.html

[Solingen/Berghout 1999] Solingen, v. R.; Berghout, E.: The Goal/Question/Metric Method. McGraw Hill Verlag 1999

[Turowski 1999] Turowski, K.: Ordnungsrahmen für komponentenorientierte betriebliche Anwendungssysteme, In: Turowski, K. (Hrsg.): Tagungsband 1. Workshop komponentenorientierte betriebliche Anwendungssysteme, Magdeburg, 1999