Ein Softwarequalitätsmodell für SOA

Ghervin Diduch

Abstract—Diskussionen im industriellen Umfeld haben gezeigt, dass durch die Verwendung von Service Oriented Architecture(SOA), positive sowie negative Effekte in der allgemeinen Softwarequalität festzustellen sind. Dies ist ein eindeutiger Indikator dafür, dass SOA ein eigenes Modell zur Qualitätssicherung benötigt.
Aktuelle Vorgehen zur Qualitätssicherung decken nur Teile der in einem SOA Umfeld wichtigen Aspekte ab. So wie sich das traditionelle Testen von Software an SOA anpassen musste, muss sich nun auch das Modell zur Qualitätssicherung anpassen.
Um die Qualität von Dienst orientierten Systemen zu beschreiben, wurde ein vereinheitlichtes Meta-Modell erstellt. Auf diese Weise werden die verschiedenen SOA Qualitätssicherungsmodelle transparenter und die Unterschiede zwischen ihnen sichtbar. Nachfolgend werden die Qualitätssicherungsmodelle "S-Cube", "OASIS" und "Quamoco" sowie das in SOA integrierte Qualitätssicherungsmodell vorgestellt.

Index Terms—Software Quality, Quality Model, SOA, SBA

1 Einleitung

In diesem Paper wird ein SOA-Qualitätsmodell dargelegt. Des Weiteren werden bestehende SOA-Qualitätsmodelle und Software-Qualitätsmodelle betrachtet und analysiert. SOA ist eine wohl etablierte Architektur, vor allem in Enterprise Anwendungen[1]. Es wird als Rückgrad für die Business to Business (*B2B*) Kommunikation angesehen,da mit der *Service oriented Architecture* schnell auf neue Anforderungen reagiert werden kann.[2]

Sofware wird immer komplexer und stellt damit immer neue Anforderungen an die Qualitätssicherung. Aufgrund dieser Entwicklung werden aktuelle Techniken der Software Qualitätssicherung in Zukunft nicht mehr ausreichen.[3]

Umso wichtiger ist es, ein Verständnis der Softwarequalitätssicherung in Verbindung mit der *Service oriented Architecture* zu erlangen, damit die Softwarequalität auch in Zukunft im Umfeld einer *Service oriented Architecture* sichergestellt werden kann.

An SOA und Softwarequalität wird individuell sehr stark geforscht. Dies bietet eine gute Grundlage für die Verbindung dieser beiden Teilbereiche. Die aktuelle Forschung ist sich allerdings noch nicht einig, wie sich der SOA Ansatz auf die Qualität eines Systems auswirkt. Auch in der Industrie wurde der Einfluss von SOA auf ein bestehendes System kontrovers diskutiert. Es wird debattiert, ob sich der SOA Ansatz positiv oder negativ auf das System auswirkt. Dies ist ein klares Zeichen für eine fehlende Qualitätsmessung.

Existierende Software-Qualitätsmodelle behandeln nicht alle Aspekte eines SOA-Systems, was darauf verweist, dass ein SOA-Qualitätsmodell nötig ist, welches alle Eigenschaften eines SOA-Systems vereint. Dieses SOA-Qualitätsmodell muss sich in bestehende Testverfahren für SOA-Systeme sowie in aktuelle Analyse Tools einbinden lassen. Aus diesem Grund wird ein Metamodell zur Messung der Qualität eines *Service-Orientierten Systems* erarbeitet. Mit der Erstellung dieses Metamodells wird der Forschung sowie den Anwendern der SOA-Architektur ermöglicht, ihre Ergebnisse einheitlich zu messen und zu vergleichen.

In Kapitel 2 werden ähnliche Arbeiten im Bereich der Qualitätssicherung für SOA-Systeme einbezogen und analysiert, aber auch auf Beiträge im Bereich der Software-Qualitätssicherung eingegangen. Nachfolgend wird in Kapitel 3 eine vereinheitlichte Notation für die Darstellung von Qualität in einem SOA-System gegeben. In Kapitel 4 wird dann das SOA-Qualitätsmodell in Teilen vorgestellt sowie einige Beispiele zur Ausprägung angeführt. Abschließend wird in Kapitel 5 eine Zusammenfassung des Papers und ein Ausblick auf die nächsten Themen des SOA-Qualitätsmodells gegeben.

• *Ghervin Diduch. E-mail: ghervin.diduch@gmail.com*

Manuscript received 12 September 2013; accepted 10 January 2014; posted online 29 March 2014; mailed on 1 May 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Vergleichbare Arbeiten

In den Bereichen der allgemeinen Qualitätssicherung, der Softwareentwicklung, aber auch der SOA wird viel geforscht. In den folgenden Kapiteln werden einige dieser Arbeiten aufgegriffen und untersucht. Hierbei handelt es sich um zwei Qualitätsmodelle für SOA sowie ein leicht operationalisierbares Qualitätsmodell für die Softwareentwicklung.

2.1 S-Cube Referenzmodell

Es existieren viele Publikationen, welche Qualitätsmodelle explizit für SOA vorschlagen. Eines davon ist das *Quality Reference Model for Service Based Applications*[4].

Für die Erstellung dieses Qualitätsmodels wurden verschiedene Qualitätsmodelle aus unterschiedlichen Bereichen verwendet:

Unter anderem die ISO 9126 aus dem *Software Engineering* (SE), aber auch aus Bereichen *Service-oriented computing* (SOC), *Business Process Modeling* (BPM), *Grid Computing* und einige *UML Profile*.

Um die Brücke zwischen dem *Software Engineering* und dem *Service-oriented computing* zu schlagen, wurden statische Analysen zur Ermittlung der *Quality-of-service* Attribute, aber auch *design-by-contract* Modelle berücksichtigt.

Grundsätzlich versteht man ein Qualitätsmodell als eine strukturierte Sammlung von Attributen, welche die Qualität anzeigen. Diese Definition eines Qualitätsmodells umfasst allerdings nur den beschreibenden Aspekt. Das S-Cube Reference Model ist eine Auswahl von ca. 90 hierarchisch strukturierten Qualitätsattributen, aufgeteilt auf insgesamt 4 Ebenen mit jeweils einer Beschreibung sowie einer Definition des Qualitätsattributs. Elemente der obersten Ebene sind z.B. *Dependability, Security, Data-Related, Network, Infrastructure Related, Quality of Use Context, Usability, Configuration, Management, Cost, Performance* und noch viele weitere.

Von den Autoren wird das *S-Cube Reference Model* beschrieben als: "*...first common understanding of quality attributes that are important for servicebased applications.*". In einem ihrer letzten Werke *Integrated principles, techniques and methodologies for assuring quality*[1] haben sie das hier beschriebene Qualitätsmodell noch erweitert, allerdings bieten sie immer noch keine Tools zur automatischen Auswertung der Qualitätsattribute an. Dies sehen sie noch als zukünftige Arbeiten an.

Im Allgemeinen besteht das *S-Cube Refenrece Model* aus einer Sammlung von 89 Qualitätsattributen, das sehr viele Qualitätskonzepte abdeckt. Leider überschneiden sich diese Qualitätsattribute sehr oft. So existieren z.B. die beiden Attribute *availability* und *coninuous*

[1] `http://www.s-cube-network.eu/results/deliverables/wp-jra-1.3`

availability, welche intuitiv dasselbe ausdrücken. Denn eine laufende Verfügbarkeit(*coninuous availability*) des Systems ist nur dann zu erreichen, wenn die Verfügbarkeit (*availability*) 100 % betragen würde. Diese Art der Überschneidungen treten mehrfach und ohne erkennbaren Grund auf. Die Autoren führen an, dass ihre Qualitätsattribute der ISO 9126 folgen. Dies entspricht aber nicht immer der Wahrheit. Aufgrund der großen Menge an Qualitätsattributen und der häufigen Überschneidungen ist das S-Cube Reference Model zu komplex, um intuitiv angewendet werden zu können.

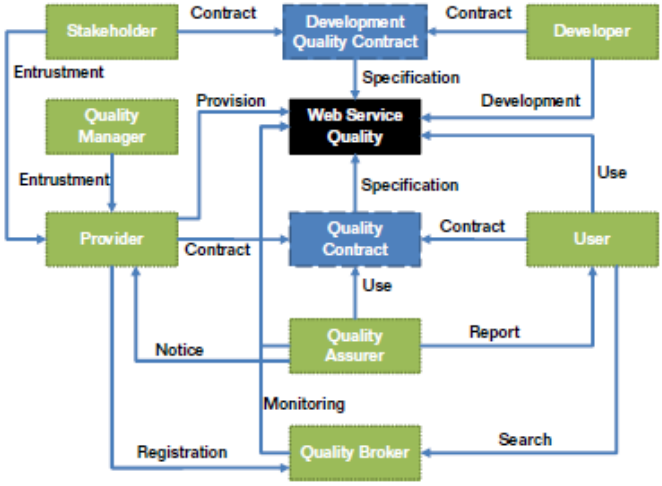

Abbildung 1. Elemente des OASIS WSQM

2.2 OASIS Qualitätsmodell für Webservices

Das Quality Model for Web Services (WSQM) 2.0 [5] von OASIS verfolgt einen ähnlichen Ansatz wie das in 2.1 gezeigte *S-Cube Referenz Model*. Auch hier werden Qualitätsattribute, die für die Qualitätsermittlung von Web Services wichtig sind, gesammelt.

Sie werden *Quality Factors* genannt und verbinden sich mit den *Quality Activities*. Des Weiteren gibt es *Quality associates*, die bestimmte Aufgaben oder Rollen in einem Unternehmen sowie die Rollen, welche den Service benutzen, wie z.b. Entwickler, Provider, oder Consumer, abbilden. Diese *Quality associates* interagieren mit anderen Elementen bzw. sich selbst über die *Quality Activities*. In Abbildung 1 wird dieses Zusammenspiel verdeutlicht. Die *Quality associates* werden als grüne Rechtecke mit gepunkteter Außenlinie und die *Quality Activities* als Pfeile dargestellt.

Im WSQM enthaltene Qualitätsattribute sind *Business Value, Service Level Measurement, Interoperability, Business Processing, Manageability und Security* . Jedes dieser Qualitätsattribute hat wiederum mehrere *Sub-Factors*, die einer Ebene zugeordnet werden. Im WSQM werden insgesamt drei Ebenen definiert.

Die erste Ebene bildet das *Business*, das den Mehrwert für das Unternehmen beschreibt, welcher mittels der Nutzung des Web Services durch den Consumer entsteht.

Die zweite Ebene ist der *Service*, der die messbaren Werte darstellt, welche bei der Nutzung des Services ermittelt werden können, z. B. die Verfügbarkeit oder die Performance.

Zuletzt wird die Ebene *System* genannt. Sie charakterisiert die Kompatibilität zwischen den einzelnen Web Services, sowohl auf Daten-, als auch auf Funktionsebene im System. Sie beschreibt aber auch die Sicherheits- sowie Handhabungs-Aspekte des Systems.

Über das WSQM kann man sagen, dass es sich momentan noch in einem unfertigen Zustand befindet. Es gibt viele Definitionen, die nicht präzise genug sind [6]. Außerdem fehlt eine Auskunft darüber, wie die definierten Qualitätsattribute gemessen werden sollen. Auch werden keinerlei Metriken zur Verfügung gestellt. Dies könnte dem Umstand geschuldet sein, dass die Beschreibung seit dem Jahr 2005 ohne Aktualisierung im *Commitee Draft*-Status festhängt.

2.3 Quamoco Modell für Softwarequalität

Das *Quamoco Model for Software Quality* ist kein direktes Qualitätsmodell für SOA, es wird stattdessen als Forschungsprojekt und umfangreiches Qualitätsmodell für Software benannt. Um es jedoch auch für SOA verwenden zu können, muss es operationalisierbar und anpassbar sein. Zur Erfüllung dieser Anforderungen wurde ein Metamodell entwickelt [7, 8]. Im Unterschied zu den zuvor genannten Qualitätsmodellen enthält dieses ein Modell der zu prüfenden Software selbst. Es beschreibt Entitäten, z. B. *Identifier* oder *Expression* aus denen die Software besteht. Diese Entitäten werden mit sogenannten "*Is-a*" oder "*part-of*" Verbindungen organisiert.

Das Quamoco Qualitätsmodell funktioniert nach dem *Faktor Konzept*. Ein Faktor ist eine Eigenschaft der zu prüfenden Software, welche einen direkten Einfluss auf die Qualität der Software hat und wird in dem Modell als Attribut einer Entität dargestellt. Faktoren sind z. B. *Correctness of Expression* oder *Self-Descriptive of Identifier*.

Um die Anpassbarkeit des Qualitätsmodells zu gewährleisten, ist dieses in Module unterteilt, was eine Trennung des gesamten Qualitätsmodells in viele kleine Module erlaubt. Diese einzelnen Module weisen untereinander wiederum Abhängigkeiten auf. Das *Quamaco Mode for Software Quality* ermöglicht die Erweiterung oder Überschreibung bestehenden Module, des Weiteren bietet das erstellte Metamodell eine Struktur, um Tools zur Messung anzubinden. Es wird von Partnern benutzt, welche eigene Richtlinien und unternehmensspezifische Qualitätsstandards etc. hinzufügen. Dieses individuelle Qualitätsmodell wird dann wieder integriert, wodurch ein sehr großes Qualitätsmodell mit mehr als 1073 Elementen entsteht. Zur automatisierten Messung der Faktoren bietet *Quamoco* eine eigene Toolkette[9] an. Diese Tools können sich in unterschiedliche Quellcode Analysetools integrieren. Unter anderem in FindBug und PMD für Java sowie in Gandarme für C# oder PCLint für C/C++. Eine Studie[8] zu diesen Tools zeigte, dass die Ergebnisse der automatischen Auswertung den Auswertungen eines Experten in nichts nachstehen.

Quamoco definiert ein generelles Metamodell, das alle Anforderungen der Kunden beachtet. Dies zeigte eine Untersuchung[10] aus dem Jahre 2010. Allerdings ist es zu allgemein und stellt keinerlei Leitfaden oder Anweisung dazu bereit, wie aus diesem Metamodell ein echtes Qualitätsmodell erstellt werden soll. Die Struktur des Metamodells limitiert es auf einzelne Softwareprodukte. Es könnte zwar ausreichen, um ein Qualitätsmodell für SOA zu erstellen, würde aber keine Antwort auf SOA spezifische Qualitätsfragen liefern können.

3 Modelierungs Konzepte

Viele Qualitätsmodelle sind nur beschreibend [z. B. [4]]. Diese werden zugunsten jener, die sich praktisch anwenden lassen, explizit nicht betrachtet. Im Folgenden wird geklärt, wie das Erreichen von definierten Qualitätszielen(*goals*) mithilfe von Systemeigenschaften (*facts*) und deren Maßnahmen(*measures*) bewertet werden können.

Definition 1 (Activity): *An activity is performed on or with the system and is characterized by an attribute describing a desired property of that activity, which is related to the quality of the system.* [11]

Definition 2 (Activity Refinement): *An activity that is part of another activity renes it.*[11]

Definition 3 (Entity): *An entity is an element of a system or a service, to which a property can be assigned.*[11]

Definition 4 (Attribute): *An attribute is an observable property of an entity.*[11]

Definition 5 (Fact): *The tuple consisting of entity and attribute is denoted fact. Feasible facts wrt. the quality model are those that are assessable using a degree of fulllment.*[11]

Fakten setzen sich immer aus einer *Entity* und einem *Attribute* zusammen und können *Activities* positiv ($\longrightarrow +$) oder negativ ($\longrightarrow -$) beeinflussen. So wird z.B. aus der *Entity* '*Interface Description*' und dem *Attribute* '*UP-TO-DATENESS*' ein *Fact*. Dieser *Fact* definiert die Aktualität eines Interfaces und wird folgendermaßen dargestellt:

[Interface Description | UP-TO-DATENESS]

Allgemein kann man sagen, je aktueller eine Interfacebeschreibung ist, desto besser kann ein Service wiederverwendet (*re-used*) werden.

Der gerade beschriebene *Fact* hat also einen positiven Einfluss ($\longrightarrow +$) auf die *Activity* '*Reuse*'.

[Interface Description | UP-TO-DATENESS] $\longrightarrow +[Reuse]$.

Dies ist ein ähnlicher Ansatz wie bei ABQM oder Quamoco.

Definition 6 (Fact Refinement): *A fact which describes a more specic property or a property of a more specic entity compared to another fact, renes this fact.*[11]

Definition 7 (Impact): *An impact describes an in uence a fact has on an activity, which is either positive or negative wrt. to a particular attribute of this activity.*[11]

Definition 8 (Measures): *Tools or manually conducted tasks to translate certain characteristics of an entity into a quantitative representation are called measures.*[11]

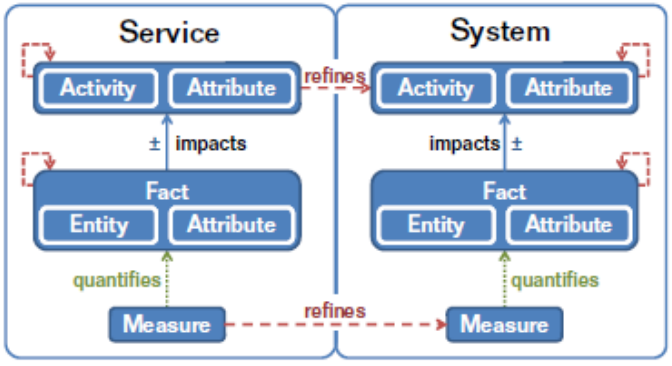

Abbildung 2. Elemente des SOA Qualitätsmodell

3.1 Modell Schichten

Viele der bekannten Qualitätsmodelle sind nicht direkt auf SOA ausgelegt, sondern eher auf normale Software oder Systeme und deren Sub-Systeme, ähnlich wie Quamoco. Deswegen muss eine Erweiterung des Qualitätsmodells geschaffen werden, die zwischen einem Qualitätsmerkmal eines Services und einem Qualitätsmerkmal eines Systems unterscheidet. Dafür wurden sogenannte *Scopes* angelegt.

Definition 9 (Scope): *The scope of an activity, a fact or a measure denes whether the respective element refers to a single service or the whole system:[Activity | ATTRIBUTE]* $_{Scope}$*, [Entity | ATTRIBUTE]* $_{Scope}$*, Measure* $_{Scope}$.[11]

Für bestimmte Systeme kann es nötig sein, mehrere zusätzliche *Scopes* zu definieren, wie z. B. *composite service* oder *business process*. Diese Erweiterung soll aber nicht Bestandteil des vorliegenden Papers sein.

Auf Abbildung 2 wird die Beziehung zwischen den beiden definierten *Scopes Service* und dem *System* veranschaulicht.

Messwerte eines Services können das System auf zwei unterschiedliche Weisen beeinflussen. In der ersten Möglichkeit bedingt ein Messwert eines Services das ganze System. Dies ist zum Beispiel bei der Namensgebung eines Services der Fall. Ist der Name eines Services schlecht gewählt, leidet die Übersichtlichkeit des ganzen Systems darunter. Wir definieren also:*MeasureService Refines MeasureSystem*.

In der zweiten Variante wird eine *System-Activity* durch eine *Activity* des Services beeinflusst. So definiert sich z.B. die Zuverlässigkeit bzw. Verfügbarkeit eines Systems über die Zuverlässigkeit bzw. Verfügbarkeit aller seiner Services. Diese beiden Wege der Beeinflussung sind in der Abbildung 2 durch die gestrichelten Linien dargestellt. Dies meint, dass sich ein SOA System anhand der Qualität seiner enthaltenen Services definiert. Ein einzelner Service hat somit Einfluss auf das gesamte SOA-System. Die Qualität eines Services steht allerdings nicht in Abhängigkeit des SOA-Systems, in welchem er genutzt wird. Die Qualität des SOA-Systems hat keinen Einfluss auf die Qualität eines enthaltenen Services.

Definition 10 (Measure refinement) *A service-level measure that is referred to in the denition of a system-level measure renes this measure.*[11]

Definition 11 (Activity refinement) *A service-level activity that is relevant for a system-level activity renes this activity.*[11]

Für die Elemente *fact* und *activity* muss in einem Qualitätsmodell für SOA immer der *Scope* mit angegeben werden. Z.B.:

[Service Name | APPROPRIATENESS]$_{\text{Service}} \longrightarrow +[Discover|EFFICIENCY]_{\text{Service}}$

Diese Beziehungen der Grundelemente des Qualitätsmodells werden in Abbildung 2 dargestellt. Hier sind die Beeinflussungen (*impacts*) als blaue Pfeile dargestellt, welche jeweils positiv oder negativ sein können. Die Ausprägungen (*refines*) werden mit roten gestrichelten Linien dargestellt und die Quantifizierungen der Fakten (*facts*) durch die Maßnahmen(*measures*) mit einer grün gestrichelten Linie.

4 Modell Anwendung

In diesem Kapitel soll geschildert werden, wie das in Kapitel 3 definierte Konzept *intuitiv* und *formal* angewendet werden kann. Kapitel 4.1 beschreibt und zeigt die Anwendung von qualitätsbezogenen Aktivitäten (*quality-related activities*). Im Kapitel 4.2 wird dann auf die Service-Eigenschaften (*Facts*) Bezug genommen, die für die Qualität des Systems relevant sind. Abschließend werden in Kapitel 4.3 einige Beispielmaßnahmen (*measures*) vorgestellt.

4.1 Qualitäts Aktivitäten

Viele der *Quality Activities* des SOA Qualitätsmodells stammen aus früheren Arbeiten von Deissenboeck[12], Winter[13] und Wagner[14]. Weitere SOA spezifische *Activities* wurden basierend auf früheren Studien [15] hinzugefügt, wenn sie benötigt wurden. Die Überführung eines "*-ilities*" basierenden Qualitätsmodells in ein *Activity-Based* Qualitätsmodell erfolgt in mehreren Schritten. Als erstes muss eine Sammlung von benötigten *Activities* erstellt werden z. B. aus Literatur oder anderen Quellen. Danach müssen die bestehenden Definitionen der "*-ilities*" Elemente einer oder mehrerer *Activities* zugewiesen werden. Nachfolgend werden nun einige der wichtigsten SOA relevanten *Activities* aufgeführt. Viele *Activities* können erst in verschiedenen Phasen des Betriebs ermittelt werden, z.B. Test, Nutzung oder Wartung. Aus diesem Grund wurde, wenn nötig, eine Phase der Erhebung mit angegeben.

4.1.1 Understanding

Um einen Service gut nutzen zu können, muss er vom Anwender (*Consumer*) verstanden werden. Je einfacher ein Service verstanden werden kann, desto besser kann er genutzt werden. Diese *Activity* ähnelt der in der ISO 25010 beschriebenen Definition für die "*Appropriateness Recognizability*" wo es heißt: *The degree to which users can recognise whether the product is appropriate for their needs. [...] Appropriateness recognisability will depend on the ability to recognise the appropriateness of the functions from initial impressions of the product and/or any associated documentation.*[16]. Diese *Activity* drückt die Verständlichkeit eines Services aus und fällt damit in den *Scope* Service.

[Understand | EASE] Service

4.1.2 Consumption

Diese Activity sagt aus, inwieweit ein Service den Anforderungen von einem Nutzer (Consumer) gerecht wird. Sie ist von der in der ISO 25010 beschriebenen Definition *Functional Appropriateness* abgeleitet, in der es heißt: "*The degree to which the functions are suitable for specied tasks and user objectives.*"[16]

Die Definition für das SOA-Qualitätsmodell lautet also: [Consumption | EFFECTIVENESS]$_{\text{Service}}$

4.1.3 Support

Diese *Activity* ist eine sehr wichtige für SOA, da sie beschreibt, wie einfach *Consumer* eines Services bzw. eines Systems technisch supported werden können. Diese *Activity* ist nicht von der ISO 25010 abgedeckt, weil sie sehr SOA spezifisch ist. Da diese Activity für Services und Systeme gilt, ist sie in beiden Scopes enthalten. [Support | EFFICIENCY]$_{\text{Service}}$ [Support | EFFICIENCY]$_{\text{System}}$

4.1.4 Service Reuse

Dies ist eine sehr bedeutsame *Activity*, nicht nur in der SOA Umgebung. Sie ist abgeleitet von der ISO 25010 Definition "*Reusability*" in der es heißt: "*The degree to which an asset can be used in more than one system, or in building other assets.*"[16].

Da die Wiederverwendung eines ganzen SOA-Systems sehr unwahrscheinlich ist, befindet sich diese *Activity* nur in dem Scope des Services. [Composition | EFFICIENCY]$_{\text{Service}}$

4.1.5 Extension

Diese *Activity* drückt die Einfachheit aus, mit der eine neue Funktion in ein SOA-System hinzugefügt werden kann. Dies ist etwas komplexer, denn sie kann auf zwei unterschiedliche Arten interpretiert werden.

Die erste Möglichkeit bildet die Erweiterbarkeit eines Services, welche eine Änderung / Erweiterung des Quellcodes des Services zur Folge hätte. Dieser Ansatz soll nicht von einem SOA Qualitätsmodell abgedeckt werden, da dies eine Aussage über die Quellcodequalität wäre. Und diese Anforderung muss von einem Qualitätsmodell, das sich auf die Programmiersprache bezieht, erfasst werden. Z. B. von dem *Quamoco base model* [7].

Die zweite Möglichkeit beinhaltet das Hinzufügen eines neuen Services in das SOA-System, um diese neue Funktionalität im Sytsem zu ergänzen. Jener Service kann aber Abhängigkeiten zu anderen Services im SOA-System haben. Die Erweiterbarkeit eines SOA-Systems ergibt sich also daraus, wie einfach es ist, einen neuen Service in dem SOA-System mit den bestehenden Services zu verbinden. Es ergibt sich eine Definition, die folgendermaßen aussieht: [Composition | EFFICIENCY]$_{\text{Service}} \longrightarrow +[Extension|EFFICIENCY]_{\text{System}}$

4.2 Qualitätsfaktoren

System-Facts bestehen aus einem Attribut, einer Entität sowie einer Beschreibung und werden kurz als *Fact* (Fakt) bezeichnet. Bei einem Fakt müssen seine Einflüsse auf die *Activities* beachtet werden, da ein Fakt nur dann sinnvoll ist, wenn er eine qualitätsrelevante *Activity* betrifft. Dies kann eine *Activity* aus dem *Scope* Service oder System sein und die Beeinflussung kann entweder *positiv* oder *negativ* sein. Des Weiteren benötigt jeder Fakt eine Berechnungsregel, damit eine Aussage über seine Erfüllung getroffen werden kann. Im Folgenden werden einige der wichtigsten Fakten näher beschrieben. Sie sind alle mit einer eindeutigen ID gekennzeichnet, um anschließend Verbindungen zu den Maßnahmen aus Kapitel 4.3 herstellen zu können.

F1: [ServiceName | APPROPRIATENESS]$_{\text{Service}}$

Nach Shim[17] kennzeichnet die Namensgebung eines Services in einem System einen wichtigen Faktor, denn je aussagekräftiger der Name eines Services ist, desto einfacher kann er von möglichen Konsumenten gefunden oder von anderen Services verwendet werden. Diese beiden Faktoren haben wiederum Einfluss auf die Verständlichkeit(*understandability*) und die Wiederverwendbarkeit(*reusability*) des Services. Diese Definition bei der Software-Entwicklung gleicht der von Deissenboeck und Pizka [8]. Der Fakt wird von der Maßnahme (*Measure*) M1 (*Appropriateness of Service Name*) verwendet, die wiederum aussagt, dass der Name eines Services seine Funktionalität vollumfänglich beschreiben muss. Diese Auswertung kann nicht maschinell gemacht werden, da eine Maschine dies nicht leisten kann.

$\longrightarrow +[Discovery|EFFICIENCY]_{\text{Service}}$

$\longrightarrow +[Understanding|EASE]_{\text{Service}}$

F2: [OperationName | APPROPRIATENESS]$_{\text{Service}}$

Eigentlich die gleiche Messung wie F1, nur nicht auf Service Ebene, sondern auf Operationsebene. So soll der Name einer Operation dessen Funktionalität vollumfänglich beschreiben. Dieser Fakt wird von der Maßnahme M2 (*Appropriateness of Operation Name*) verwendet. Die beiden Fakten F1 und F2 werden von Shim[17] in einem Fakt zusammengefasst. Sie werden hier allerdings getrennt betrachtet, da sie unterschiedlich starken Einfluss auf die Qualität eines Services haben. So wird F2 wie folgt definiert:

$\longrightarrow +[Understanding|EASE]$

Ein gut gewählter Name einer Operation unterstützt den Konsumenten darin, die Funktion und den Funktionsumfang zu verstehen.

F3: [InterfaceDescription | PERSPICUITY]$_{\text{Service}}$

Um einen Service gut nutzen zu können, ist eine genaue Definition seines *Interfaces* nötig. Dies wurde in einer Arbeit von Chen und Huang [18] analysiert. Hierbei ist nicht nur der technische Aspekt im Sinne einer WSDL wichtig, sondern auch eine menschenlesbare Beschreibung des *Interfaces*, denn eine gute Darstellung des *Interface* eines Services erhöht die Wiederverwendbarkeit eines Services wie in 4.1.4 angeführt.

$\longrightarrow +[Composition|EFFICIENCY]_{\text{Service}}$

F4: [Operation j FUNCTIONAL ATOMICITY]$_{\text{Service}}$

Dieser Fakt sagt aus, dass eine Operation eines Services so autonom wie möglich sein sollte[19], was bedeutet, dass eine Operation immer nur eine Funktion ausführen sollte. Dies ist essentiell für eine hohe Wiederverwendbarkeit der Operationen in einem Service, denn je autonomer eine Operation ist, desto eher kann sie von anderen Services genutzt werden.

$\longrightarrow +[Composition|EFFICIENCY]_{\text{Service}}$

F5: [Interface | COMPLEXITY]$_{\text{Service}}$

Ein Service, dessen Interface sehr komplex ist, sind für den Nutzer sehr schwer zu verstehen [17]. Deswegen ist es relevant, das *Interface* eines Services so einfach wie möglich zu halten. Dies betrifft nicht nur die Anzahl der Operationen eines Services, sondern auch die Anzahl sowie die Komplexität der zu übergebenden Parameter und der zu erwartenden Rückgabewerte

$\longrightarrow +[Understand|EASE]_{\text{Service}}$

$\longrightarrow +[Composition|EFFECTIVENESS]_{\text{Service}}$

F6: [Interface | COHESION]$_{\text{Service}}$

Die Kohäsion eines Services wird anhand der Verbindungen seiner Operationen gemessen. Je stärker die Operationen eines Services miteinander in Verbindung stehen, desto größer ist die Kohäsion des Services.

Eine hohe Kohäsion der Operationen eines Webservices ist anzustreben, da hoch kohäsive Services meist einfacher zu verstehen und dadurch leichter wiederzuverwenden sind. Des Weiteren bilden bei hoch kohäsiven Services meist die Eingabeparameter die Rückgabeparameter einer anderen Operation desselben Services, weswegen diese gut miteinander verbunden werden können.

Die Kohäsion eines Services ist schwer zu messen, da der Begriff "Verbindung" sehr abstrakt ist und deswegen auf unterschiedlichste Weise gemessen werden kann. Allgemein ist zu sagen, dass wenig kohäsive Services meist weniger geeignet sind, um eine vollumfängliche Abdeckung der Anforderungen des Konsumenten zu erreichen [20, 17].

Um die Komplexität dieses Faktes aufzuzeigen, wird dieser in den Maßnahmen M3 (*Common Parameter and Return Types*), M4 (*Number of Clients Using an Operation*) und M5 (*Sequential Operation Relation*) verwendet.

$\longrightarrow +[Understand|EASE]_{\text{Service}}$

$\longrightarrow +[Composition|EFFICIENCY]_{\text{Service}}$

$\longrightarrow +[Consumption|EFFECTIVENESS]_{\text{Service}}$

4.3 Maßnahmen

Im folgenden Kapitel werden nun einige Beispiele zu Maßnahmen (*Measures*) im SOA-Qualitätsmodell aufgezeigt. Alle Maßnahmen sind wie die Fakten mit einer eindeutigen ID gekennzeichnet, um sie mit den entsprechenden Fakten zu verbinden. Eine Maßnahme fasst ein oder mehrere Fakten zu einer messbaren Größe zusammen, sie werten sozusagen bestimmte Fakten aus. Diese Auswertung kann manuell, aber auch maschinell mithilfe bestimmter Tools geschehen. Nachfolgend werden Beispiele für beide Arten von Auswertungen dargestellt.

M1: Appropriateness of Service Name

Diese Maßnahme wertet den unter F1 erstellten Fakt "*[ServiceName | APPROPRIATENESS] Service*" aus. Die Entscheidung, ob der Servicename angemessen gewählt wurde, kann keine Maschine treffen. Sie erfordert daher eine manuelle Auswertung dieser Maßnahme. Die Beurteilung, ob ein Servicename angemessen ist, muss von einem Experten abgegeben werden. Dieser Experte muss neben dem guten technischen Verständnis auch ein sehr gutes Business Verständnis aufweisen, da er entscheiden muss, ob der Servicename sowohl die technische Funktion, als auch die businessmäßige Funktion angemessen beschreibt.

M2: Appropriateness of Operation Name

Diese Maßnahme ist der in M1 genannten Maßnahme sehr ähnlich, sie betrifft allerdings ausschließlich die Operationsnamen und nicht den Namen des Services. Wie M1 muss auch diese manuell von einem Experten ausgewertet werden, da die Entscheidung nicht maschinell getroffen werden kann.

M3: Common Parameter and Return Types

Dies ist eine von drei Maßnahmen zur Auswertung des Faktes F6 "*[Interface | COHESION] Service*". Die Auswertung dieser Maßnahme nach Perepletchikov, Ryan und Tari [20] umfasst die Ermittlung der Anzahl der Parameter, sowie der Rückgabetypen der Operationen des Services, um so einen Faktor für die Kohäsion der Operationen innerhalb des Services zu kalkulieren. Eine volle Kohäsion ist dann erreicht, wenn alle Parameter und Rückgabetypen der Operationen auf derselben *Typendefinition* basieren. Da diese Auswertung mittels *Parsen* des Quellcodes durchgeführt werden kann, ist eine maschinelle Auswertung möglich.

M4: Number of Clients Using an Operation

Dies ist ein weiterer Aspekt einer Maßnahme für F6. Die grundlegende Idee ist, dass eine Operation, die nur von wenigen Konsumenten genutzt wird, nur eine geringe Verbindung mit den anderen Services hat. Perepletchikov, Ryan und Tari [20] bezeichnen diese Art der Kohäsion als "*Externe Kohäsion*". Die Auswertung kann allerdings erst nach der Entwicklung des Services gemacht werden, also während der *Testphase* oder sogar erst während des *produktiven Betriebes*. Sie kann jedoch maschinell, z.B. mithilfe der Auswertung von Logs, erfolgen.

M5: Sequential Operation Relation

Das letzte Beispiel für eine Maßnahme des Fakts F6. Sequentiell verbunden bedeutet, dass der Output einer Operation als Input einer anderen Operation dient [20]. Um dies zu gewährleisten, ist eine hohe Kohäsion des Services wichtig wie in M3 (*Common Parameter and Return Types*) beschrieben. Die sequentielle Verbundenheit kann nicht nur innerhalb eines Services gemessen werden, sondern auch über alle Services in einem SOA-System. Die Daten über die sequentielle Verbundenheit mit anderen Services können weitgehend maschinell ermittelt werden.

5 Conclusion and Outlook

In dem vorliegenden Paper wurde ein *Metamodel* zur Ermittlung der Qualität von Services und SOA-Systemen erstellt. Dieses Metamodell besteht aus *Activities*, *Entities*, *Attributes*, *Facts* und *Measures* sowie deren *Beziehungen* zueinander. Außerdem wurden *Scopes* definiert, um die Elemente im Qualitätsmodell zu spezifizieren. Es wurden Definitionen für *Fakten*, *Maßnahmen* und *Attribute* sowie verschiedene Beispiele gegeben.

Um dieses Qualitätsmodell für ein konkretes SOA-System zu nutzen, müssten die *Measures* sowie die *Facts* noch gewichtet werden. Nur so können aussagekräftige Ergebnisse für die Qualität des Systems gewonnen werden.

Außerdem müssten noch Tools zur maschinellen Auswertung angebunden werden, so wie Deissenboeck [9] es gezeigt hat. Den nächsten Schritt bildet die Evaluation des Qualitätsmodells in der Industrie, um die Akzeptanz und die Bereitschaft zur Anwendung zu erhöhen. Diese Evaluation war für das Jahr 2012 geplant. Das hier vorgestellte SOA Qualitätsmodell wurde als Teil der *Quamoco family of domain-specic quality models* im Jahr 2012 veröffentlicht.

Literatur

[1] N. Mazzocca V. Casola, A. R. Fasolino and P. Tramontana. A framework for evaluating quality and security in service oriented architectures.

[2] D. Masak. Moderne enterprise architekturen. 2005.

[3] J.-P. Garbani. Future trends in the enterprise software market. 2009.

[4] The European Network of Excellence in Software Services and Systems (S-Cube). Quality reference model for sba. 2008.

[5] OASIS. Quality model for web services (wsqm2.0). *Working Draft*.

[6] J. Hundlin. Modellierung von qualitatsmerkmal für services. *ACM Conf. on Computer Supported Cooperative Work (CSCW)*.

[7] K. Lochmann and L. Heinemann. Integrating quality models and static analysis for comprehensive quality assessment. *Workshop on Emerging Trends in Software Metrics (WETSoM 2011)*.

[8] K. Lochmann M. Klas and L. Heinemann. Evaluating a quality model for software product assessments - a case study. *Workshop zur Software-Qualitatsmodellierung und -bewertung (SQMB '11)*.

[9] M. Herrmannsdoerfer K. Lochmann F. Deissenboeck, L. Heinemann and S. Wagner. The quamoco tool chain for quality modeling and assessment. *Int. Conf. on Software Engineering*.

[10] S. Nunnenmacher S. Wagner M. Herrmannsdorfer M. Klas, C. Lampasona and K. Lochmanan. How to evaluate meta-models for software quality? *DASMA Metrik Kongress (MetriKon '10)*.

[11] Andreas Goeb and Klaus Lochmann. A software quality model for soa. 2011.

[12] M. Pizka S. Teuchert F. Deissenboeck, S. Wagner and J. F. Girard. An activity-based quality model for maintainability. *IEEE Int. Conf. on Software Maintenance*.

[13] S. Wagner S. Winter and F. Deissenboeck. A comprehensive model of usability. *Engineering Interactive Systems Conf. (EIS '07)*.

[14] S. Islam S. Wagner, D. Mendez Fernandez and K. Lochmann. A security requirements approach for web systems. *Workshop Quality Assessment in Web (QAW 2009)*.

[15] D. Voelz and A. Goeb. What is different in quality management for soa? *14th Int. IEEE EDOC Conf.*

[16] ISOn. Iso/iec 25010. 2011.

[17] S. Kim B. Shim, S. Choue and S. Park. A design quality model for service-oriented architecture. *15th Asia-Pacific Software Engineering Conf. (APSEC '08)*.

[18] J. Chen and S. Huang. An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software*.

[19] J. Barnard. A new reusability metric for object-oriented software. *Software Quality Journal*.

[20] C. Ryan M. Perepletchikov and Z. Tari. The impact of service cohesion on the analyzability of service-oriented software. *IEEE Transactions on Services Computing*.