Entwicklung eines objektiven Bewertungsverfahrens für Softwarearchitekturen im Bereich Fahrerassistenz

D. Ahrens[1], A. Frey[1], A. Pfeiffer[1] und T. Bertram[2]

[1]Fahrdynamik
Fahrerassistenz und aktive Sicherheit
BMW Group
80788 München
dirk.ahrens@bmw.de

[2]Lehrstuhl für Regelungssystemtechnik
Technische Universität Dortmund
44221 Dortmund
torsten.bertram@tu-dortmund.de

Abstract: Der vorliegende Beitrag beschreibt ein Vorgehensmodell und die erzielten Ergebnisse im Umfeld der Bewertung von Softwarearchitekturen von Automotive Embedded Software. Softwarearchitektur stellt im Allgemeinen ein entscheidendes Instrument zur Beeinflussung nicht-funktionaler Eigenschaften (z.B. *Skalierbarkeit*, *Erweiterbarkeit*, *Portierbarkeit*) von Software dar, bis heute gibt es jedoch keinen geeigneten Ansatz, die Qualität solcher Architekturen *objektiv* und *quantitativ* zu ermitteln und zu bewerten.

Zunächst wird ein *Qualitätsmodell* mit einer Vielzahl unterschiedlicher Kriterien aufgestellt, welches auf die Bedürfnisse von Automotive Embedded Software angepasst ist. Für die relevanten Kriterien werden auf dieser Basis insgesamt *acht Metriken* zur quantitativen Beurteilung von Automotive Softwarearchitekturen erarbeitet.

Zur automatisierten Anwendung dieser Metriken ist eine Einbettung in den aktuellen Entwicklungsprozess notwendig. Diese Anpassungen und die daraus resultierende und eingesetzte Infrastruktur für den Bewertungsablauf wird ebenso vorgestellt wie der schlussendlich *implementierte Softwareprototyp* auf Java-Basis. Mit diesem Entwicklungswerkzeug ist eine automatisierte, schnelle, komfortable und individuell konfigurierbare Bewertung von vorhandenen Softwarearchitekturentwürfen möglich.

Abgerundet wird der Beitrag durch ausgewählte Anwendungsbeispiele und den Ausblick auf die weiteren anstehenden Arbeiten in diesem Umfeld.

1 Einführung und Problemstellung

Die Entwicklung von Automotive Embedded Software steht momentan vor großen Herausforderungen. Einerseits ist sie längst als wichtigster Schlüssel und Motor für kommende wettbewerbsentscheidende Innovationen identifiziert, andererseits muss die Entwicklung, Umsetzung und Absicherung in immer kürzeren Zeitabständen unter Einsatz von knapper werdenden (menschlichen und finanziellen) Ressourcen für gleichzeitig immer mehr Baureihen, Fahrzeugtypen und Derivate (kurz: Varianten) geschehen. Es gilt somit zwangsläufig einen unvermeidbaren Konflikt zwischen steigender Funktionalität, Komplexität sowie Vernetzung und wesentlichen nicht-funktionalen Anforderungen wie *Skalierbarkeit*, *Erweiterbarkeit*, *Portierbarkeit* und *effizienter Ressourcenausnutzung* so gut wie möglich zu lösen. Durch den Entwurf einer Softwarearchitektur, welcher üblicherweise sehr früh (s. Abb. 1 rechts) im Entwicklungsprozess entsteht, werden die obigen Eigenschaften der Software entscheidend – positiv wie negativ – beeinflusst. Nachträgliche Änderungen an der Architektur sind zu einem späteren Zeitpunkt nicht mehr oder nur durch enormen Aufwand durchführbar. Einem guten und reifen Architekturentwurf vor der Implementierung kommt somit eine entscheidende Bedeutung zu.

In der heute gängigen Praxis bei der Entwicklung von Softwarearchitekturen stellt sich erst am Ende der Entwicklung heraus, ob und wie gut die nicht-funktionalen Anforderungen von der Software(-architektur) erreicht werden. Unzulänglichkeiten können dann kaum noch korrigiert werden. Alle Entwurfsentscheidungen hängen stark von bewährten Mustern (allgemeinen oder firmenspezifischen) ab und sind auf subjektive und intuitive Entscheidungen von Experten und deren persönlicher Erfahrung angewiesen. Dieses individuelle Wissen ist in den Köpfen weniger Mitarbeiter gebunden und kann nur schwer dokumentiert oder an neue Mitarbeiter weiter gegeben werden. Sind neue Personen an der Entwicklung beteiligt, muss also aufgrund mangelnder Erfahrung bei „Null“ angefangen werden. Gleichzeitig bedeutet die Tatsache, dass gewisse Muster oder Praktiken seit längerer Zeit angewendet werden, nicht automatisch, dass diese auch wirklich gut in Hinblick auf die spezifischen Anforderungen sind. Es bleibt unklar, wie viel Potential zu Verbesserungen noch besteht. Gänzlich fehlend sind also Möglichkeiten die „Güte“ einer Softwarearchitektur *zuverlässig*, *reproduzierbar* und vor allem *objektiv* feststellen und mit anderen Entwürfen vergleichen zu können, um dadurch eine zielgerichtete Weiterentwicklung und iterative Evolution der Architektur zu ermöglichen.

Ziel der vorliegenden Arbeit ist es einen Ansatz zu entwickeln, mit dem geeignete Kriterien zur Softwarearchitektur-Bewertung gefunden, definiert, formalisiert und *quantifiziert messbar* gemacht werden können, um im Anschluss als *objektive Metriken* automatisiert auf eine in einem definierten, noch festzulegenden Datenformat gegebene Softwarearchitektur angewendet zu werden.

Bisherige Ansätze [Bo78][ED96][ISO01][Oe06] beschäftigen sich bisher ausschließlich mit der Bewertung von bereits implementierter Software und nicht mit den im Entwicklungsprozess deutlich früher zu entwerfenden Architekturen oder sie beschäftigen sich zu oberfläch mit den Möglichkeiten zur Bewertung von Softwarearchitekturen. Es werden meist nur allgemeine Kriterien, Empfehlungen und Ideen geschildert ohne objektivierte oder formalisierte Hilfsmittel anzubieten [Ab96][EHM08][Lo03][PBG07][RH06].

Der hier vorgestellte Ansatz schließt diese Lücke und bietet dem Anwender erstmals ein echtes Hilfsmittel zur *objektiven und quantifizierten Bewertung von Softwarearchitekturen*, welches einen entscheidenden Beitrag zur Verbesserung von Softwarequalität bezüglich nicht-funktionaler Anforderungen leisten kann. Dazu werden die Kriterien aus den bestehenden Ansätzen gesammelt, kombiniert und an die speziellen Bedürfnisse der Automotive Softwareentwicklung am Beispiel von Fahrerassistenzsystemen angepasst. Für das daraus entstandene *Qualitätsmodell* wird untersucht, welche Kriterien besonders relevant für die Domäne und welche davon überhaupt quantitativ erfassbar sind. Die erarbeiteten Metriken werden gesamthaft und an einigen Beispielen auch im Detail vorgestellt. Darüber hinaus wird zum besseren Verständnis ebenfalls der *umgebende Entwicklungsprozess* als notwendige Infrastruktur sowie der implementierte *Software-Prototyp*, welcher eine automatisierte Anwendung der Metriken auf gegebene Softwarearchitekturen ermöglicht, präsentiert. Aktuelle Erkenntnisse aus der Anwendung der gesamten Wirkkette sowie der Ausblick auf weitere Arbeiten runden den Beitrag ab.

2 Einbettung in den Entwicklungsprozess

Wichtig zum weiteren Verständnis des Bewertungsverfahrens ist die Kenntnis der Randbedingungen zu seinem Einsatz. Es wurde im Vorfeld ein Entwicklungsprozess (Abb. 1) ausgearbeitet, welcher eine hohe Durchgängigkeit, Formalismen sowie automatisierte (teil- und vollautomatisierte) Übergänge zwischen Prozessschritten ermöglicht und somit als Ergänzung beziehungsweise Erweiterung der bisher in der Praxis üblichen Entwicklungsschritte eingesetzt werden kann und soll.

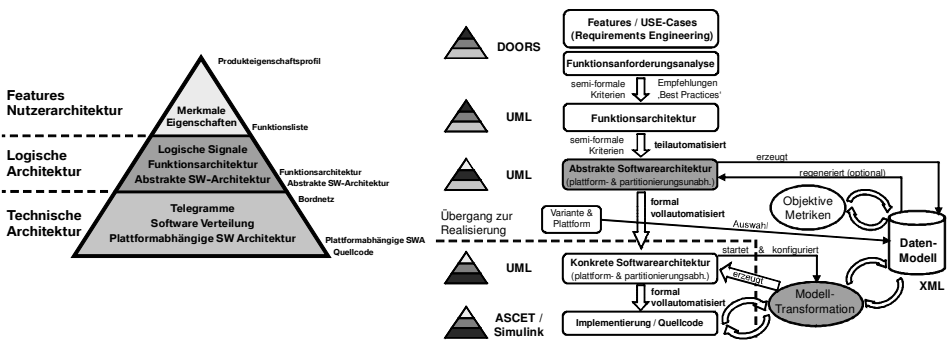

Abbildung 1: Rahmen bildendender Softwareentwicklungsprozess

Details zum umgebenden Prozessrahmen (Abb. 1 links), den detaillierten Prozessschritten (Abb. 1 rechts) sowie den entwickelten Konzepten und Übergängen können [Ah08] [Ah10][APB08] entnommen werden. Für das Verständnis der vorliegenden Arbeit wesentlich sind die auf der rechten Seite der Abbildung hervorgehobenen beiden Prozessschritte. Dabei handelt es sich zum einen um die abstrakte Softwarearchitektur, eine auf einem UML Metamodell [OMG05] basierende Darstellung speziell erstellt für Automotive Softwarearchitekturen [Ah08][Ah09][APB08] und zum anderen um das Konzept der automatisierten Modelltransformation, welche die bidirektionale Überführung zwischen der UML-Darstellung und den Implementierungswerkzeugen ermöglicht.

Zu diesem Zweck wird als Zwischenschritt ein formales konsistenzwahrendes Datenmodell auf XML-Basis [W3C98], beschrieben durch ein XML-Schema, verwendet.

Die Anwendung der Metriken selbst soll auf Basis dieser Datenbank erfolgen, so dass sowohl ein Top-Down-Vorgehen für in der UML erstellte Entwürfe als auch ein Bottom-Up-Vorgehen zur Bewertung und Neustrukturierung bereits bestehender Architekturen (kommend von den Implementierungs-Tools) möglich ist. Details zum Datenmodell sowie der konkreten Umsetzung und Anwendung der Metriken finden sich in Kapitel 4.

3 Qualitätsmodell und die erarbeiteten Bewertungsmetriken

Bisher wurde der Begriff „Softwarearchitektur“ bereits mehrfach verwendet. Da sowohl in der Literatur als auch im fachlichen Sprachgebrauch keine allgemeingültige Definition zu finden ist, auch wenn sich der Grundgedanke meist sehr ähnelt, soll kurz vorgestellt werden, was im Rahmen dieser Arbeit unter einer Softwarearchitektur verstanden wird.

> „Eine Softwarearchitektur beschreibt die Struktur eines Softwaresystems. Diese umfasst die statischen Strukturelemente (z.B. Komponenten, Module), ihre extern sichtbaren Eigenschaften (z.B. Attribute, Methoden) sowie ihre nach außen sicht- und nutzbaren Schnittstellen und Interaktionen.“

Aus dieser Definition wird klar erkennbar, welche Elemente und Informationen für eine Bewertung der Softwarearchitektur überhaupt zur Verfügung stehen. Somit sind die Grenzen der objektiven Bewertung klar absteckbar. Gleichwohl bestand im Rahmen der Untersuchungen ebenfalls die Möglichkeit bestimmte Informationen für die Darstellung (UML-Profil) und Datenablage (formales Datenmodell) zusätzlich zu fordern und als dauerhafte Erweiterung der beiden genannten Dokumentationsformen fest zu etablieren.

3.1 Das Qualitätsmodell

Basis für die konkreten zu entwickelnden und zu implementierenden Metriken bildet zunächst eine Übersicht über die Gesamtheit aller grundsätzlich möglichen Qualitätskriterien. Darin sind zunächst sowohl funktionale als auch nicht-funktionale Kriterien enthalten, wobei im Folgenden der Fokus klar auf die nicht-funktionalen gelenkt werden soll. Bei der Sammlung der Kriterien konnte auf bestehende Ansätze zur Bewertung von Softwarequalität zurückgegriffen werden [Bo78][Lo03][Mc94][ISO01][Oe06][Ra93]. Ergebnis ist das in Abb. 2 dargestellte, so genannte *Qualitätsmodell*, welches sieben Oberkriterien mit jeweils mehreren (mit Ausnahme der Konformität) Unterkriterien enthält. Oberkriterien, von jetzt an auch als *Qualitätskriterien* bezeichnet, sind nicht direkt messbar sondern fassen vielmehr ein oder mehrere direkt messbare Kriterien, von jetzt an auch als *Qualitätsattribute* bezeichnet, zu einer Obergattung zusammen. Die obigen Ansätze schlagen die Kriterien wie erwähnt primär für den Einsatz zur Bewertung von Software vor, die Bewertung einer Softwarearchitektur kann in diesem Falle jedoch als Spezialfall einer Software-Bewertung aufgefasst werden, so dass die Kriterien grundsätzlich kompatibel sind.

Das hier vorgestellte Qualitätsmodell kombiniert, erweitert und ergänzt die aus den unterschiedlichen Ansätzen entnehmbaren Kriterien und spezialisiert diese im Anschluss daran an die speziellen Bedürfnisse des vorliegenden Anwendungsfalls. Dabei handelt es sich um Softwarearchitekturen von Automotive Embedded Software, so dass eine zweifache Spezialisierung vorgenommen wird. Dies ist für alle Kriterien und Attribute notwendig, weil diese je nach Anwendungsbereich (Domäne) stark unterschiedlichen Charakter aufweisen können und somit nicht direkt miteinander vergleichbar bzw. übertragbar sind. Ein Beispiel hierfür ist die Skalierbarkeit, welche in Softwareprodukten vieler Anwendungsbereiche ein wesentliches Qualitätsmerkmal darstellt, aber aufgrund der stark unterschiedlichen Eigenarten dieser Domänen ganz unterschiedlich ausgeprägt ist. Während beispielsweise in webbasierten Datenbankanwendungen Zugriffszeiten und Datendurchsätze sich entsprechend der Skalierung der Datenbank verhalten sollten, steht im Automobilbereich das ressourceneffiziente Hinzufügen und Entfernen von Funktionalität für unterschiedliche Kunden- oder Plattformausprägungen von bestimmten Funktionen/Systemen im Vordergrund, um je nach Ausprägung mit unterschiedlich leistungsstarken Plattformen (Steuergeräten) auszukommen und somit Kosten zu sparen. Die Skalierbarkeit äußert sich somit in der einfachen Entfern- und Hinzufügbarkeit von Funktionalität und baut somit auf einen modularen Aufbau von Software. Solche speziellen Randbedingungen gilt es für alle Kriterien zu berücksichtigen.

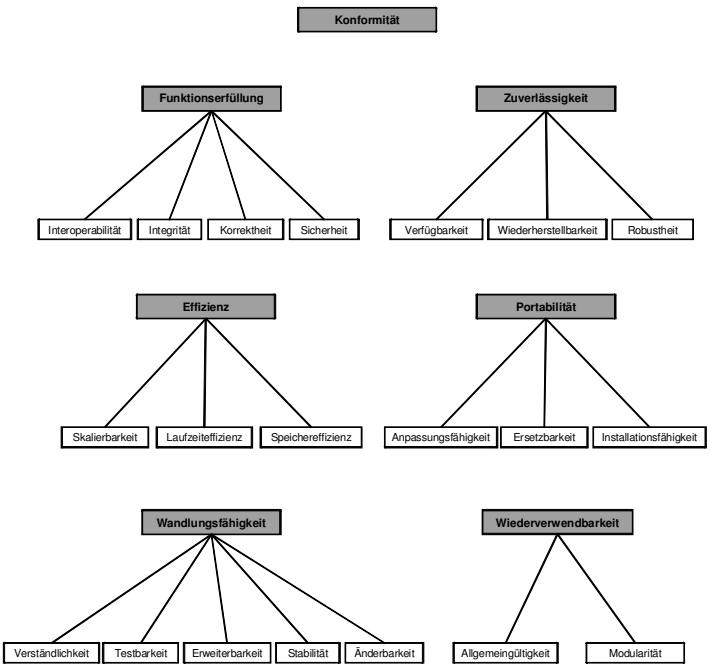

Abbildung 2: Qualitätsmodell zur Klassifizierung und Bewertung von Softwarearchitektur

Neben der Definition und Anpassung ist ein weiterer wesentlicher Schritt die Relevanz und Bedeutung der Kriterien und Attribute für die Automobildomäne zu ermitteln, um so eine Priorisierung in Hinblick auf die Entwicklung der Metriken vornehmen zu können.

Eine Einteilung der Qualitätskriterien kann entsprechend der drei Faktoren *Einsatzbedingung*, *Produktionskosten* und *Entwicklungskosten* vorgenommen werden (Abb. 3).

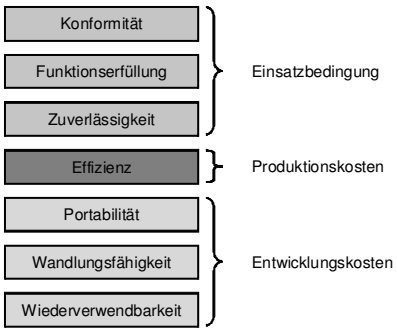

Abbildung 3: Relevante Qualitätskriterien für den Automotive-Bereich

In der BMW Group existieren bereits verschiedene Verfahren und Prozesse, um die drei Qualitätskriterien Konformität, Funktionserfüllung und Zuverlässigkeit sicherzustellen. In der vorliegenden Arbeit wird daher der Fokus zur Entwicklung von Metriken auf die zwei Schwerpunkte Produktions- und Entwicklungskosten gelegt. Eine objektive Bewertung unterschiedlicher Architekturentwürfe soll, wenn möglich, nach den Kriterien *Effizienz*, *Wandlungsfähigkeit* und *Wiederverwendbarkeit* vorgenommen werden.

3.2 Die objektiven Bewertungsmetriken

Ziel aller Architekturmetriken ist das messbare beziehungsweise formale „greifbar machen“ von vormals subjektiven Designentscheidungen, Architekturmustern und Best Practices. Es galt somit das subjektive und schwer zugängliche Wissen erfahrener Softwarearchitekten „anzuzapfen“ und in eine reproduzierbare, objektive und quantifizierte Form zu überführen. Dadurch wird eine Korrelation zwischen subjektiven und objektiven Kriterien und Entwurfsmustern geschaffen. Die Basis für die Metriken waren somit intensive Fachgespräche und Diskussionen mit entsprechenden Mitarbeitern des Unternehmens. Über standardisierte Fragebögen und das Durchgehen konkreter (für alle Interviews identischer) Fallbeispiele wurde die Grundlage zur Messbarkeit geschaffen.

Durch die Gespräche hat sich gezeigt, dass grundsätzlich für nahezu alle der relevanten Attribute der drei Kriterien eine Möglichkeit besteht mit Hilfe der in der Softwarearchitektur zur Verfügung stehenden Informationen eine quantitative Metrik herzuleiten. Lediglich für die Kriterien *Erweiterbarkeit* und *Allgemeingültigkeit* ließ sich zum jetzigen Zeitpunkt kein geeignetes Verfahren finden. Dies liegt in beiden Fällen daran, dass reines Strukturwissen nicht ausreichend ist, um eine zuverlässige Aussage zu treffen. Jede Architektur ist grundsätzlich erweiterbar, wie „gut“ oder einfach allerdings eine solche Erweiterung vorgenommen werden kann, liegt ganz konkret an der Funktionalität, die hinzugefügt werden soll. Somit ist Kontextinformation, mit anderen Worten Detailwissen über mögliche künftige Erweiterungen, zwingend erforderlich. Ähnlich verhält es sich mit der Allgemeingültigkeit, die ebenso nicht ausschließlich auf Strukturinformationen fußend bewertet werden kann.

Die im Rahmen dieser Arbeit entwickelten, objektiven Softwarearchitektur-Metriken finden sich in Abb. 4. Auf zwei ausgewählte Metriken, *Strukturverständlichkeit* sowie *Modularität*, soll im Folgenden kurz eingegangen werden, um einen Eindruck über Umfang und Charakter der Metriken zu vermitteln.

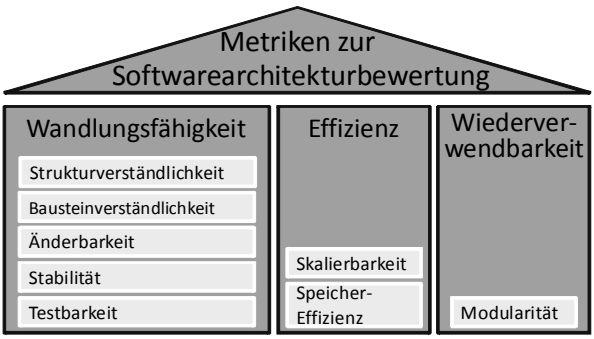

Abbildung 4: Übersicht über die entwickelten und implementierten Metriken des Qualitätsmodells

Metrik zur Strukturverständlichkeit:

Idee:
Die Struktur der Softwarearchitektur hat einen großen Einfluss auf ihre Verständlichkeit für die beteiligten Stakeholder beziehungsweise Entwickler. Durch Abstraktion und Verfeinerung können Softwarekomponenten in Subkomponenten zerlegt oder zu übergeordneten Bausteinen zusammengefasst werden. So entstehen verschiedene Sichten mit unterschiedlichem Detaillierungsgrad. Nach [Mi56] sind Systeme, die aus 7±2 Subsystemen bestehen, für einen Menschen optimal verständlich. Daher soll in dieser ersten Metrik unter der obigen Prämisse die Anzahl der Subsysteme über die gesamte Softwarearchitektur bewertet werden.

Metrik:
Um eine Gewichtung und Eingrenzung des Wertebereichs vornehmen zu können, wird für die Anzahl der Subsysteme einer Komponente eine Klassifizierung vorgenommen. Zu diesem Zweck wird der *Component Complexity Factor (CCF)* eingeführt, welcher sich nach Tabelle 1 zusammensetzt. Der CCF wiederum wird in der Berechnungsvorschrift der Metrik benötigt.

Tabelle 1: Berechnung des Component Complexity Factors

# Subkomponenten	CCF
0	0
1-4	2
5-9	0
10-11	3
12-14	5
>14	10

$$M1_i = \frac{CCF_i + \frac{\sum_{j=1}^{n} M1_j}{n}}{2}$$

$1 \leq j \leq n$
i = aktuelle Komponente
n = Anzahl aller Subkomponenten von i
CCF_i = Component Complexity Factor von Komponente i
$M1_i$ = Strukturverständlichkeit der aktuell untersuchten Komponente i
$M1_j$ = Strukturverständlichkeit von Subkomponente j

Erläuterungen:
Die Metrik arbeitet rekursiv über alle Hierarchieebenen hinweg. In der aktuellen Hierarchieebene wird ein Wert für die Strukturverständlichkeit des jeweiligen Strukturelements gebildet und zur Ebene darüber hochgegeben, um dort mit den Werten der übrigen Bausteine des gleichen Levels gemittelt zu einem Gesamtwert des umgebenden Strukturelements verarbeitet zu werden. Der Wertebereich dieser Metrik liegt zwischen 0 und 10. Aus Platzgründen kann an dieser Stelle leider nicht eingehender auf die Metrikanwendung eingegangen werden.

Metrik zur Modularität:

Idee:
Eine gute Modularität zeichnet sich durch eine starke Kohäsion und schwache Kopplung aus [BC99]. Logisch zusammenhängende Softwaremodule, welche viele Informationen austauschen, sollten demnach zu einer Komponente zusammengefasst werden. Eine starke Zusammengehörigkeit und schwache äußere Abhängigkeit kann in der modellbasierten Entwicklung vor allem über das Verhältnis interner zu externer Kommunikation überprüft werden. Dies gilt für alle Strukturelemente auf beliebigen Ebenen.

Metrik und Erläuterungen:
Für die Modularitäts-Metrik wird kein zusätzlicher Faktor benötigt, maßgeblich ist für jeden Baustein allein das Verhältnis zwischen Botschaften, die auf derselben Hierarchieebene ausgetauscht werden, zu der Gesamtzahl aller (internen und externen) Botschaften. Zunächst ermittelt die Formel das Verhältnis zwischen interner und externer Kommunikation einer Komponente. Mit einer Gewichtsverteilung von 50:50 fließen die Modularitätswerte der gegebenenfalls vorhandenen untergeordneten Komponenten – ebenso durch ein rekursives Verfahren – in die Berechnung ein. Der Wertebereich liegt zwischen 0 und 1.

$$M7_i = \frac{\frac{\sum Mess_{ext,i}}{\sum Mess_{ext,i} + \sum Mess_{int,i}} + \frac{\sum_{j=1}^{n} M7_j}{n}}{2}$$

$1 \leq j \leq n$

i = *aktuelle Komponente*

n = *Anzahl aller Subkomponenten von i*

$Mess_{ext,i}$ = *ein- oder ausgehende Schnittstelle der Komponente i zu einer anderen Komp.*

$Mess_{int,i}$ = *Schnittstelle zwischen Subkomponenten der Komponente i*

$M7_i$ = *Modularität der aktuell untersuchten Komponente i*

$M7_j$ = *Modularität von Subkomponente j*

3.3 Anwendung der Metriken

Alle Metriken können grundsätzlich unabhängig voneinander eingesetzt werden. Dies ermöglicht einen flexiblen und individuellen Einsatz je nach den Bedürfnissen des bewertenden Entwicklers. Für eine gesamthafte Architekturbewertung werden alle einzelnen Metriken berechnet und in einer bestimmten Gewichtung ins Verhältnis gesetzt. Auch diese Gewichtung ist im Tool (Kapitel 4) individuell einstellbar, als Ausgangsgewichtung wurde die in Tabelle 2 dargestellte Gewichtung vorgenommen, welche aus den subjektiven Einschätzungen der befragten Entwickler zur Wichtigkeit abgeleitet wurde.

Tabelle 2: Initiale Gewichtung aller Kriterien und Attribute

Kriterium	Gewicht	Metrik	Gewicht
Wandlungsfähigkeit	2	Strukturverständlichkeit	1
		Bausteinverständlichkeit	1
		Änderbarkeit	3
		Stabilität	1
		Testbarkeit	2
Effizienz	2	Skalierbarkeit	2
		Speicher-Effizienz	1
Wiederverwendbarkeit	1	Modularität	1

Auch wenn eine absolute Bewertung einer einzigen Architektur problemlos möglich ist, hat sich gezeigt, dass ein relativer Vergleich zweier Vergleichsarchitekturen zweckmäßiger ist. Dies liegt zum einen daran, dass die Metriken prinzipbedingt unterschiedliche Wertebereiche haben, was ein Gewichten zu einem einzelnen absoluten Gesamtwert erschwert. Zum anderen ist in der frühen Phase des Softwarearchitektur-Entwurfs, zu der oftmals noch gar nicht die endgültigen (funktionalen und nicht-funktionalen!) Anforderungen an das Softwaresystems vorliegen, eine isolierte Einordnung der quantitativen Metrikwerte schwierig. Erst mit steigender Erfahrung durch zahlreiche Architekturbewertungen kann für vergleichbare Projektumfänge auch über den einzelnen Zahlenwert eine Einschätzung der Architekturgüte vorgenommen werden. Die Autoren schlagen daher zunächst eine vergleichende Architekturbewertung vor, bei der grundsätzlich immer eine Referenz- und eine Vergleichsarchitektur bewertet werden. Die Metriken werden im Hintergrund für jede Architektur berechnet, anschließend wird eine prozentuale Veränderung angegeben. Dadurch wird die Problematik der unterschiedlichen Wertebereiche umgangen, es kann somit auch ein (gewichteter) Gesamtwert der relativen Verbesserung ermittelt werden.

In den bisherigen Bewertungen von Softwarearchitekturen hat sich dieses Verfahren bewährt und dient zukünftig der Feinabstimmung der Metriken in Hinblick auf Wertebereich, Faktoren und Gewichtung.

4 Softwareprototyp und Praxisbeispiel

Alle acht vorgestellten (Attribut-)Metriken sind inzwischen in einem Softwareprototyp umgesetzt. Das Werkzeug dient momentan in der Entwicklung von Fahrerassistenzsystemen der Sammlung von Erfahrungen im Umgang mit objektiven Softwarearchitekturbewertungen und dient der Evaluierung des Ansatzes. Aufgrund der automatischen Überführung der Softwarearchitektur in das angesprochene Datenformat können auch große und komplexe Softwarearchitekturen von realen Fahrerassistenzsystemen schnell und komfortabel bewertet werden.

Der Softwareprototyp wurde mit der Open-Source Entwicklungsumgebung Eclipse auf Java-Basis umgesetzt und bedient sich der JAXB-Technologie [SUN03]. JAXB ist eine Programmierschnittstelle in Java, welche es ermöglicht XML-Dateien, die einem XML-Schema konform aufgebaut sind, durch einen so genannten „unmarshall-Vorgang“ in Java-Klassen zu überführen. Durch herkömmliche Java-Programmierkonstrukte werden die in den Klassen hinterlegten Struktur- und Attributinformationen ausgelesen und entsprechend der vorgegebenen Metrik-Algorithmen verarbeitet. Die Bedienung erfolgt für den Benutzer komfortabel mit einer grafischen Bedienoberfläche (GUI). Dort können die zu bewertenden XML-Artefakte (Softwarearchitekturen) eingestellt und die Bewertung individuell konfiguriert werden. Die zu berechnenden Einzelmetriken sowie deren Gewichtung können ebenso frei eingestellt werden. Das Tool wurde in Form einer jar-Datei so umgesetzt, dass es auf beliebigen (Java-)Plattformen eigenständig lauffähig ist.

Eine beispielhafte und tabellarisch aufbereitete Bewertung von zwei Softwarearchitekturvarianten ist im Folgenden kompakt dargestellt, die Ergebnisse sind als relative Änderung aufgelistet (Tabelle 3). Referenzarchitektur ist die aktuelle Softwarearchitektur der Fahrerassistenzsysteme der Längsdynamik der BMW Group, Vergleichsarchitektur ist ein erster Entwurf zur Neustrukturierung und Optimierung. In diesem konkreten Fall wurden gezielt wenige Maßnahmen durchgeführt, welche sich nur auf die beiden vorgestellten Teilmetriken auswählen. Dabei handelt es sich um strukturelle Maßnahmen zur Änderung der Komposition von feingranularen zu grobgranularen Bausteinen, welche sich sowohl auf die Strukturverständlichkeit (‚Aufbau und Anzahl von enthaltenen Subkomponenten‘) als auch die Modularität (‚innere zu äußere Kommunikation‘) auswirken. Es bietet sich augenscheinlich zum Kennenlernen der Metriken – z.B. Ausnutzung des Wertebereichs, Sensitivität etc. – an, kleinere, isolierte Umstrukturierungsmaßnahmen vorzunehmen und einzeln zu bewerten.

Dies entspricht dem grundsätzlich iterativ-inkrementellen Prozess des schrittweisen Softwarearchitektur-Entwurfs, bei welchem die vorgestellte objektive Softwarearchitekturbewertung künftig einen entscheidenden Stellenwert einnehmen soll. Kleinere Werte entsprechen dabei grundsätzlich Verbesserungen, somit sind im relativen Vergleich zweier Architekturvarianten negative Vergleichswerte anzustreben.

Tabelle 3: Exemplarische relative Bewertung von zwei Softwarearchitekturvarianten

Metrik		Werte-bereich	Gewicht	Referenz-SWA	Umstrukturierung	
				Wert	Wert	Diff.
Verständlichkeit	M1	0-10	1	0,645	0,570	-11,63%
Verständlichkeit	M2	0-60	1	11,370	11,370	0,00%
Änderbarkeit	M3	0-240				
Stabilität	M4	0-1	1	0,588	0,588	0,00%
Testbarkeit	M5	0-240	2	23,926	23,926	0,00%
Skalierbarkeit	M6	0-50		17,481	17,481	0,00%
Modularität	M7	0-1		0,470	0,424	-9,79%

Qualitätskriterium	Gewicht	Wert
Wandlungsfähigkeit	2	-2,33%
Effizienz	2	0,00%
Wiederverwendbarkeit	1	-9,79%
		-2,89%

5 Zusammenfassung und Ausblick

Der in dieser Arbeit vorgestellte Ansatz beschäftigt sich mit den Möglichkeiten zur objektivierten Bewertung von Automotive Embedded Softwarearchitekturen am konkreten Beispiel der Fahrerassistenz. Es wird eine Methode vorgestellt, welche in dieser Form erstmalig Metriken bereitstellt, mit denen derartige Softwarearchitekturen *objektiv* und *quantifiziert* bewertet werden können. Zu diesem Zweck wurde ein *Qualitätsmodell* entwickelt, welches aus bekannten Ansätzen zur Softwarebewertung Kriterien und Attribute zusammenträgt und für den speziellen Anwendungsfall spezialisiert und anpasst. Für die Mehrzahl der für den Automotive-Bereich relevanten Attribute wurden insgesamt acht Metriken entwickelt und an ausgewählten Beispielen vorgestellt.

Die Metriken ermöglichen eine automatisierte Bewertung einer in einem vorgegebenen selbst entwickelten Datenformat vorliegenden Softwarearchitektur. Dabei sind sowohl absolute als auch relative Architekturbewertungen möglich. Die Metriken wurden in einem Softwareprototyp bereits umgesetzt und können somit produktiv für die Entwicklung von iterativ und zielgerichtet optimierten Softwarearchitekturen der Fahrerassistenz eingesetzt werden. Verschiedene Anwendungsbeispiele runden den Beitrag ab.

Die Überführung subjektiver Entwurfsmuster und Intuitionen in objektivierte Metriken gestaltet sich erwartungsgemäß schwierig. Die entwickelten Metriken stellen somit eine Ausgangsbasis für weitere Untersuchungen dar. Erst durch zunehmende Erfahrung im Umgang mit dem Bewertungsverfahren kann die Korrelation zwischen subjektiver und objektiver Bewertung verbessert werden. Die Erfahrungen sollen helfen, die Faktoren und Gewichte der Metriken besser abzustimmen. Dadurch wird erreicht, dass sich bei vorgenommenen Änderungen der Architektur die Werte der einzelnen Metriken angemessen und konsistent zueinander ändern. Erst dann erreichen die Metriken eine Reife, die auch absolute Bewertungen einzelner Architekturentwürfe ermöglicht. Für jedes einzelne Attribut werden so Erfahrungswerte je nach Domäne und Komplexität der Softwarearchitektur ermittelt, die eine Einordnung von absoluten Zahlenwerten erst ermöglichen.

In Zukunft sollen die Metriken verstärkt in den Software-Entwicklungsprozess eingebunden werden, um eine effektive und effiziente Entwicklung von Automotive Softwarearchitekturen zu unterstützen. Die dadurch ermöglichte reproduzierbare und objektive Untersuchung der Architekturen auf ihre Eignung und Güte unterstützt die Entwickler beim Erreichen der Vielzahl nicht-funktionaler Anforderungen wesentlich.

Literaturverzeichnis

[Ab96] G. Abowd, L. Bass, P. Clements, R. Kazman, L. Northrop, A. Zaremski: "Recommended Best Industrial Practice for Software Architecture Evaluation," Georgia Institute of Technology, Carnegie Mellon University, Software Engineering Institute, 1996

[Ah08] D. Ahrens, A. Frey, A. Pfeiffer, T. Bertram: „Entwicklungsprozess und abstrakte Softwarearchitektur für modulare Automotive Software," 3. Dortmunder Autotag, 2008

[Ah09] D. Ahrens, A. Frey, A. Pfeiffer, T. Bertram: „Entwicklung einer leistungsfähigen Darstellung für komplexe Funktions- und Softwarearchitekturen im Bereich Fahrerassistenz," VDI Mechatronik 2009, Wiesloch, 2009

[Ah10] D. Ahrens, A. Frey, A. Pfeiffer, T. Bertram: „Designing Reusable and Scalable Software Architectures for Automotive Embedded Systems in Driver Assistance," to appear at SAE World Congress, Detroit, USA, 2010

[APB08] D. Ahrens, A. Pfeiffer, T. Bertram: „Comparison of ASCET and UML - Preparations for an Abstract Software Architecture," Forum on Specification and Design Languages (FDL) 2008, Proceedings, Stuttgart, Germany, 2008

[BC99] C. Baldwin, K. Clark: „Design Rules: The Power of Modularity Volume 1," Cambridge, MA, USA, MIT Press, 1999

[Bo78] B. Boehm, J. Brown, H. Kaspar, M. Lipow, G. MacLeod, M. Merrit: „Characteristics of Software Quality," Elsevier Science Ltd, 1978

[ED96] C. Ebert, R. Dumke: „Software-Metriken in der Praxis," Springer Verlag, 1996

[EHM08] S. Eicker, C. Hegmanns, S. Malich: „Projektbezogene Auswahl von Bewertungsmethoden für Softwarearchitekturen," Multikonferenz Wirtschaftsinformatik, 2008

[ISO01] ISO - International Standards Organization: „Software engineering — Product quality," ISO 9126-1, 2001

[Lo03] F. Losavio, L. Chirinos, N. Levy, A. Ramdane-Cherif: „Quality Characteristics for Software Architecture," Journal of Object Technology Vol. 2, No. 2, 2003, S. 133-150

[Mc94] J. McCall: „Encyclopedia of Software Engineering," 1994, S. 958-969

[Mi56] G. Miller: „The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," The Psychological Review 63, 1956, S. 81-97

[Oe06] K. Oey: „Nutzen und Kosten von serviceorientierten Architekturen," Universität Köln, Dissertation, 2006

[OMG05] OMG – Object Management Group: UML – Unified Modeling Language, www.uml.org

[PBG07] T. Posch, K. Birken, M. Gerdom: „Basiswissen Softwarearchitektur," dpunkt.verlag GmbH, 2007

[Ra93] J. Raasch: „Systementwicklung mit strukturierten Methoden. Ein Leitfaden für Praxis und Studium," Hanser Verlag, 1993

[RH06] R. Reussner, W. Hasselbring: „Handbuch der Software-Architektur," dpunkt.verlag GmbH, 2006

[SUN03] SUN: JAXB - Java Architecture for XML Binding, https://jaxb.dev.java.net

[W3C98] W3C – World Wide Web Consortium: XML – Extensible Markup Language, www.xml.org