8 Bewertung von Softwarearchitekturen

To measure is to know.

James Clerk Maxwell[1]

Fragen, die dieses Kapitel beantwortet:

- Warum sollten Sie Architekturen und Code bewerten?
- Was können Sie in der IT überhaupt bewerten?
- Was unterscheidet qualitative von quantitativer Bewertung?
- Wie gehen Sie bei der Bewertung von Architekturen vor?
- Welche Stakeholder sollten bei der Bewertung mitwirken?
- Wie setzen Sie Softwaremetriken sinnvoll ein?

Bewerten Sie schon, oder raten Sie noch?

Bewertung unterstützt Steuerbarkeit

„*You cannot control what you cannot measure*", sagt Tom DeMarco treffend. Positiv formuliert, bedeutet diese Aussage für IT-Projekte und Systementwicklung: Wenn Sie kontrolliert auf Ihre Projektziele zusteuern möchten, müssen Sie regelmäßig Ihre Zielerreichung bewerten, um bei Bedarf steuernd eingreifen und korrigierende Maßnahmen einleiten zu können.

Während der Systementwicklung sollten Sie regelmäßig die Frage „Gut genug?" stellen und beantworten. In Kapitel 3 haben Sie die wesentlichen Aufgaben in der Architekturentwicklung kennengelernt – darin kommt die (regelmäßige) Bewertung vor.

Bild 8.1 zeigt im Überblick, dass Bewertung auf alle übrigen Architektenaufgaben starken Einfluss nehmen kann. Auf Basis systematischer Bewertung können Sie als Softwarearchitekt schlichtweg Ihre eigene Arbeit besser planen und Ihre Ergebnisse verbessern!

In der Praxis finden Bewertungen (unter dem Titel „Audit" oder „Review") oftmals in späten Entwicklungsphasen statt, wenn das betroffene System bereits im produktiven Einsatz läuft. Viel zu selten investieren laufende Projekte in die Bewertung – obwohl das den meisten Projekten/Systemen guttäte.

[1] *http://de.wikipedia.org/wiki/James_Clerk_Maxwell*: Formulierte die Maxwellgleichungen, Grundlagen der Berechnung elektromagnetischer Wellen.

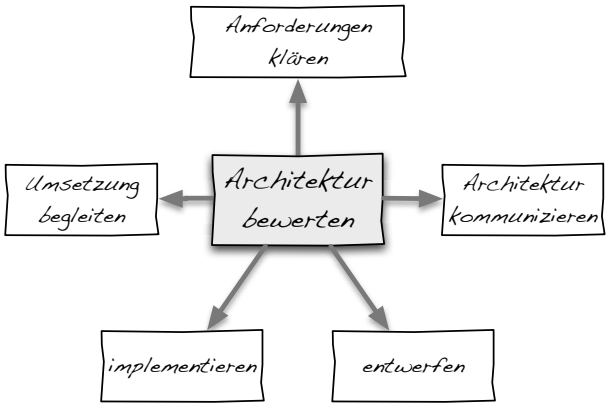

Bild 8.1
Architekturbewertung als relevante Aufgabe

Was Sie in der IT bewerten können – und wie

Sie können in Softwareprojekten zwei Arten von „Dingen“ bewerten:

- Prozesse, wie etwa Entwicklungs- oder Betriebsprozesse: Hierbei können Sie organisatorische Aspekte betrachten oder aber den Einsatz von Ressourcen bewerten. Leider können Sie auf Basis der Prozesse kaum Aussagen über die Qualität der entwickelten Systeme treffen.[2] Dieses Thema werde ich hier nicht weiter vertiefen.
- Artefakte, wie beispielsweise Anforderungen, Architekturen, Quellcode und andere Dokumente. Einige Arten dieser Artefakte (wie beispielsweise Quellcode) können Sie quantitativ, das heißt in Zahlen, bewerten. Andere (wie etwa die Softwarearchitektur) entzieht sich der rein zahlenmäßigen Bewertung. Diese Gruppe von Artefakten bewerten Sie qualitativ, also ihrer Beschaffenheit oder Güte nach. Zur letzten Gruppe gehören Softwarearchitekturen, deren Bewertung *qualitativ* erfolgt.

Zunächst möchte ich Ihnen erläutern, welche Ergebnisse Sie bei Bewertungen grundsätzlich erreichen können. Anschließend lernen Sie (in Abschnitt 8.1) qualitative Architekturbewertung kennen, angelehnt an die verbreitete ATAM-Methode.

Schließlich finden Sie in Abschnitt 8.2 einige Grundlagen quantitativer Bewertung und Metriken.

Bewertung als Soll-Ist-Vergleich

Betrachten Sie qualitative Bewertung als einen Soll-Ist-Vergleich:

- Das Soll beschreibt die Qualitätsanforderungen beziehungsweise Architekturziele. Es charakterisiert die gewünschten Eigenschaften eines Systems.
- Sie vergleichen gegen das (aktuelle oder geplante) Ist-System, möglicherweise auf Basis von Dokumentation, Plänen oder Modellen. Dabei gehen Sie freingranular vor, d. h. Sie vergleichen einzelne Anforderungen des „Soll“ mit ihren Entsprechungen im „Ist“.

[2] Sie können jedoch durch gezielte Reflexionen oder Projekt-Retrospektiven signifikante Verbesserungen erreichen, die zumindest mit hoher Wahrscheinlichkeit positive Auswirkungen auf die Qualität von Systemen haben.

Lassen Sie mich ein fachfremdes Beispiel heranziehen – siehe die nachfolgende Tabelle: Ich möchte qualitativ meinen morgendlichen Grüntee bewerten:

Tabelle 8.1 Beispiel für Soll-Ist-Vergleich

Soll	Ist	Ergebnis des Soll-Ist-Vergleichs
Hellgrüne Farbe	Hellbraune Farbe	Anforderung nicht erfüllt.
Typischer Sencha-Geschmack	Geschmack nach Kirsche und Mandel	Gewünschter Geschmack nicht gegeben, dafür aber Preis-Anforderung besonders gut erfüllt.
Grasiger Geruch	Geruchlos	Problem: Tee erfüllt Geruchsanforderung nicht.
Ungezuckert		Wegen des Kirscharomas nicht prüfbar.
Preis < 9 €/100 g	5 €/100 g	Anforderung (über-)erfüllt.
Serviert in dünnwandiger Tasse	Wandstärke der Tasse < 1,5 mm	Anforderung voll erfüllt, ok.

Bei dieser Art Vergleich können Sie grundsätzlich drei verschiedene Ergebnisse erzielen (siehe auch Bild 8.2):

1. Soll stimmt mit Ist überein – die Architektur bzw. deren Umsetzung ist bezüglich dieser spezifischen Eigenschaft ok.
2. Soll stimmt in mehreren Qualitätsanforderungen nur teilweise mit Ist überein – es wurden Kompromisse geschlossen (einzelne Merkmale wurden verbessert auf Kosten anderer).
3. Einzelne Qualitätsanforderungen können im Ist nicht erreicht werden (Risiko) oder werden nicht erreicht (Problem, eingetretenes Risiko).

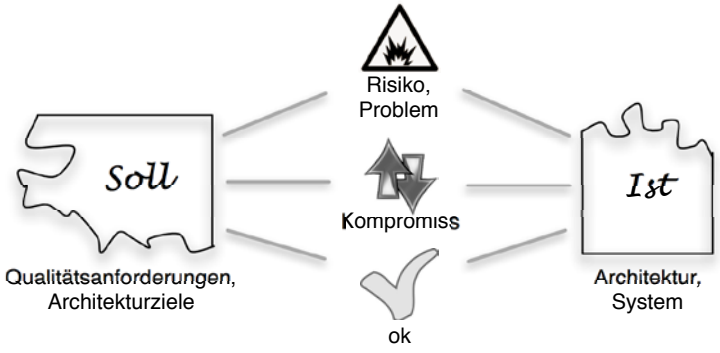

Bild 8.2
Bewertung als Soll-Ist-Vergleich

Für qualitative Bewertung ist die Granularität der Anforderungen entscheidend: Sie benötigen einen detaillierten und möglichst konkreten Katalog an Qualitätsanforderungen und Architekturzielen!

Als Vorbild könnten Sie die bekannten Produktbewertungen der „Stiftung Warentest“ heranziehen, die oftmals sehr detaillierte Kriterienkataloge (=Anforderungen) als Basis ihrer Bewertungen verwenden.

8.1 Qualitative Architekturbewertung

Bewertung von Qualitätsmerkmalen

Bewertung von Softwarearchitekturen liefert qualitative Aussagen: Sie bewerten Architekturen danach, inwieweit sie die Erreichung bestimmter Qualitätsanforderungen ermöglichen oder behindern.

Beispielsweise treffen Sie Aussagen darüber, in welchem Maß das jeweilige System den Anforderungen an Wartbarkeit, Flexibilität, Performance oder Sicherheit genügt.

Dabei liefert eine Architekturbewertung kein absolutes Maß, d. h., sie bewertet nicht die „Güte-über-alles", sondern nur im Hinblick auf spezifische Kriterien. Architekturbewertung hilft, Risiken zu identifizieren, die sich möglicherweise aus problematischen Entwurfsentscheidungen ergeben.

Bewertungsgrundlagen für Softwarearchitekturen

In Kapitel 3 (Vorgehen bei der Architekturentwicklung) haben Sie wichtige Qualitätsmerkmale von Softwaresystemen kennengelernt. Ich habe Ihnen gezeigt, wie Sie mit Hilfe von Szenarien diese Qualitätsmerkmale beschreiben und operationalisieren können.

Mit Hilfe von Szenarien können Sie spezifische Messgrößen für Ihre Systeme definieren oder, noch besser, von den maßgeblichen Stakeholdern definieren lassen.

Qualität ist spezifisch für Stakeholder

Dabei gilt es zu ermitteln, welche spezifischen Kriterien die Architektur zu erfüllen hat. Anhand dieser Kriterien können Sie dann die „Güte" oder „Eignung" der Architektur ermitteln.

Eine Architektur ist in diesem Sinne geeignet, wenn sie folgende Kriterien erfüllt:

- Das System (das mit dieser Architektur entwickelt wird oder wurde) erfüllt sowohl seine funktionalen als auch nichtfunktionalen Anforderungen (Qualitätskriterien).
- Insbesondere erfüllt das System die spezifischen Anforderungen an Flexibilität, Änderbarkeit und Performance-Verhalten.
- Das System kann mit den zur Verfügung stehenden Ressourcen realisiert werden. Sowohl das Budget wie auch der Zeitplan, das Team, die beteiligten Fremd- und Legacy-Systeme, die vorhandene Basistechnologie und Infrastruktur sowie die gesamte Organisationsstruktur beeinflussen die Umsetzbarkeit einer Architektur.

Zur Bewertung einer Architektur ist es daher von entscheidender Bedeutung, die spezifischen Qualitätskriterien des betroffenen Systems zu kennen. In der Praxis werden diese Kriterien oder Qualitätsziele oft erst bei einer Architekturbewertung expliziert.

Bereits in Kapitel 3 habe ich Ihnen empfohlen, die Qualitätsanforderungen an Ihre Systeme explizit durch Szenarien zu dokumentieren. Diese Szenarien bilden eine wertvolle Orientierungshilfe für weitere Entwicklungsschritte und eine ausgezeichnete Basis für die Architekturbewertung!

Qualitative Bewertung nach ATAM

Ich möchte Ihnen im Folgenden ein methodisches Vorgehen zur Architekturbewertung vorstellen, orientiert an der ATAM[3]-Methode von [Bass+03] und [Clements+03]. Bild 8.3 gibt einen Überblick.

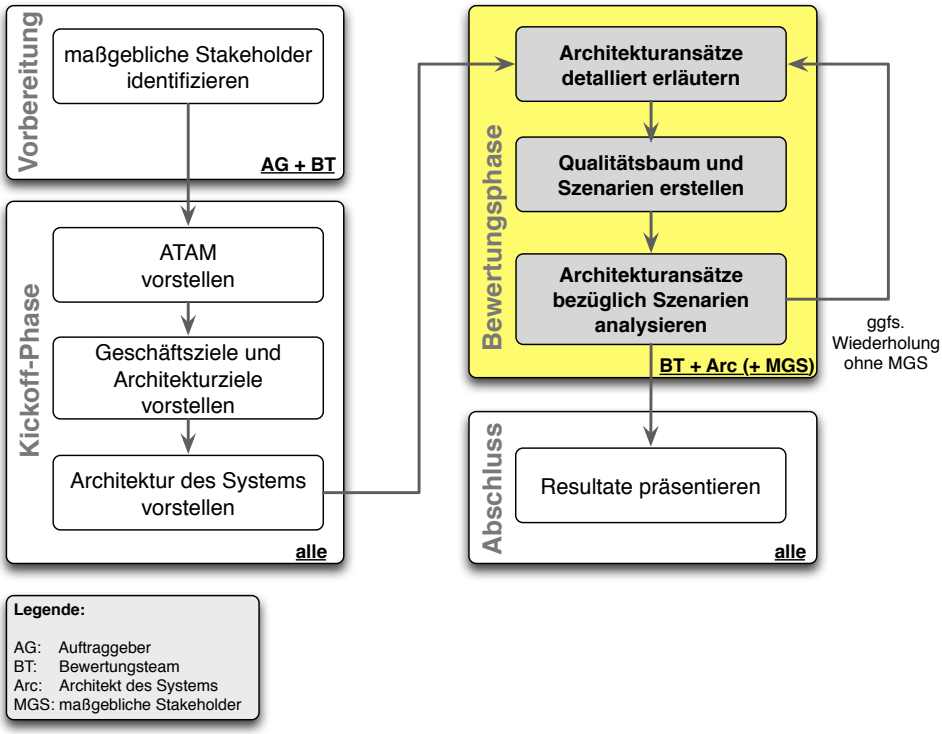

Bild 8.3 Vorgehen bei der Architekturbewertung nach ATAM

Der fundamentale Schritt bei der Architekturbewertung ist die Definition der von den maßgeblichen Stakeholdern geforderten Qualitätsmerkmale in möglichst konkreter Form. Als Hilfsmittel dazu verwenden Sie Qualitätsbäume und Szenarien (siehe auch Abschnitt 3.6.2). Ein Beispiel sehen Sie in Bild 8.4.

Maßgebliche Stakeholder identifizieren

Da die konkreten Ziele Ihrer Architekturen ausschließlich durch die spezifischen Ziele Ihrer Stakeholder vorgegeben werden, muss der Auftraggeber oder Kunde im ersten Schritt jeder Architekturbewertung die hierfür entscheidenden („maßgeblichen“) Stakeholder identifizieren. In der Regel geht eine Architekturbewertung von wenigen Stakeholdern aus, häufig aus dem Management oder der Projektleitung.

[3] Architecture Tradeoff Analysis Method

Stimmen Sie mit Ihren Auftraggebern gemeinsam ab, welche Stakeholder bei der Festlegung der Bewertungsmaße mitarbeiten sollen.* Denken Sie – neben Ihren Auftraggebern oder Kunden – in jedem Fall an Endanwender und Betreiber.

* Ich nenne diese spezielle Gruppe von Stakeholdern gerne die *maßgeblichen* Stakeholder.

Bewertungsmethode (ATAM) vorstellen

Bevor Sie mit der Definition der Bewertungsziele beginnen, sollten Sie den maßgeblichen Stakeholdern die Bewertungsmethode vorstellen. Insbesondere müssen Sie die Bedeutung nachvollziehbarer und konkreter Architektur- und Qualitätsziele klarstellen.

Verdeutlichen Sie den Beteiligten, dass es bei der qualitativen Architekturbewertung um die Identifikation von Risiken und Nicht-Risiken sowie um mögliche Maßnahmen geht, nicht um die Ermittlung (quantitativer) *Noten*.

Geschäftsziele vorstellen

Im Idealfall stellt der Auftraggeber die geschäftlichen (Qualitäts-)Ziele des Systems vor, das zur Bewertung ansteht. Dabei sollte der gesamte geschäftliche Kontext zur Sprache kommen, die Gründe für die Entwicklung des Systems sowie dessen Einordnung in die fachlichen Unternehmensprozesse.

Falls diese Ziele in Ihren Projekten bereits in den Anforderungsdokumenten aufgeführt sind, lassen Sie dennoch die *maßgeblichen Stakeholder* zu Wort kommen: Bei der Bewertung geht es um deren aktuelle Sichtweise. Außerdem sind die Zielformulierungen aus den Anforderungsdokumenten von Systemanalytikern gründlich gefiltert worden.

Ich teile die Erfahrung von [Clements+03], die von *Aha-Erlebnissen* bei der Vorstellung der aktuellen Ziele berichten: Die unterschiedlichen Teilnehmer von Bewertungsworkshops sitzen häufig das erste Mal gemeinsam an einem Tisch und lernen aus erster Hand die Zielvorstellungen anderer Stakeholder kennen. Dabei kommt es zu wertvollen „Ach sooo war das gemeint“-Erlebnissen.

Architektur vorstellen

Anschließend sollte der Architekt des Systems die Architektur vorstellen. Dazu eignet sich die Gliederung der Architekturpräsentation, die Sie in Kapitel 4 kennengelernt haben.

Es sollte in jedem Fall der komplette Kontext des Systems ersichtlich werden, das heißt sämtliche Nachbarsysteme. Außerdem gehören zu einer solchen Vorstellung die Bausteine der oberen Abstraktionsebenen sowie Laufzeitsichten einiger wichtiger Use-Cases oder Änderungsszenarien.

Architekturansätze erläutern

Welche Maßnahmen innerhalb der Architektur oder der Implementierung wurden zur Erreichung der wesentlichen (Qualitäts-)Anforderungen ergriffen? Wie wurden, strukturell oder konzeptionell, die wesentlichen Probleme oder Herausforderungen gelöst? Was sind die wesentlichen, prägenden Architekturentscheidungen?

Architekt oder Entwickler des Systems sollten diese Fragen beantworten, als Ergänzung zur vorangegangenen Überblicksvorstellung.

Einen spezifischen Qualitätsbaum erstellen

Lassen Sie Ihre Stakeholder im Anschluss an die Vorstellung der Geschäftsziele sowie der Architektur im Rahmen eines kreativen Brainstormings die wesentlichen geforderten Qualitätsziele erarbeiten. Ordnen Sie diese anschließend in (informeller) Baum-Form an: Die allgemeineren Merkmale stehen weiter links (oder oben im Baum), die spezielleren Anforderungen weiter rechts (oder unten).

Mindmaps können bei der Erstellung und Dokumentation von Qualitätsbäumen gute Dienste leisten. Sie sind leicht verständlich und insbesondere für ein *Brainstorming* von Qualitätsmerkmalen geeignet.

Ein Beispiel eines solchen Qualitätsbaumes finden Sie in Bild 8.4 und in Bild 8.5. Dort sind als globale Qualitätsziele Performance, Erweiterbarkeit und Verfügbarkeit eingetragen und durch Latenz, Durchsatz etc. verfeinert.

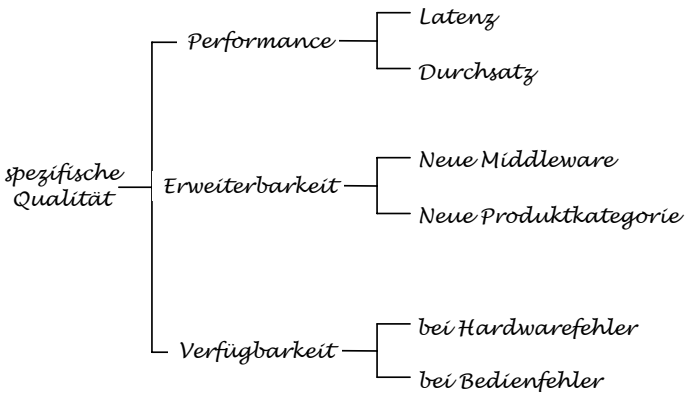

Bild 8.4
Beispiel eines Qualitätsbaumes

Entwickeln Sie Ihre Qualitätsbäume in kleinen und streng fokussierten Workshops, im Stile von Brainstorming-Sitzungen. Lassen Sie die Teilnehmer zu Beginn Architektur- und Qualitätsziele in ihren eigenen Worten beschreiben, ohne sie unmittelbar einzuordnen oder zu bewerten.

Falls Ihre Stakeholder etwas Starthilfe benötigen, können Sie die Architektur- und Qualitätsziele sowie Mengengerüste aus Anforderungsdokumenten in diese Diskussion einbringen. Ich finde es jedoch produktiver, die Teilnehmer einer solchen Diskussion möglichst wenig durch bestehende Dokumente zu belasten.

Anschließend strukturieren Sie gemeinsam mit den Teilnehmern die Beiträge in Form des Qualitätsbaumes. Dabei können Sie die Wurzel des Baumes auch anders benennen, etwa: „*Nützlichkeit*“ oder „*wichtigste Architekturziele*“.

Damit haben Sie den ersten Schritt, die Beschreibung der Bewertungsziele, bereits erledigt.

Qualitätsmerkmale durch Szenarien verfeinern

Szenarien: Abschnitt 3.6.2

Jetzt verfeinern Sie gemeinsam mit den maßgeblichen Stakeholdern die Qualitätsanforderungen und Architekturziele des Systems durch Szenarien. Wichtig ist dabei, die Formulierung der Szenarien möglichst konkret und operationalisiert zu halten. In Bild 8.5 sehen Sie einen um zwei Szenarien angereicherten Qualitätsbaum.

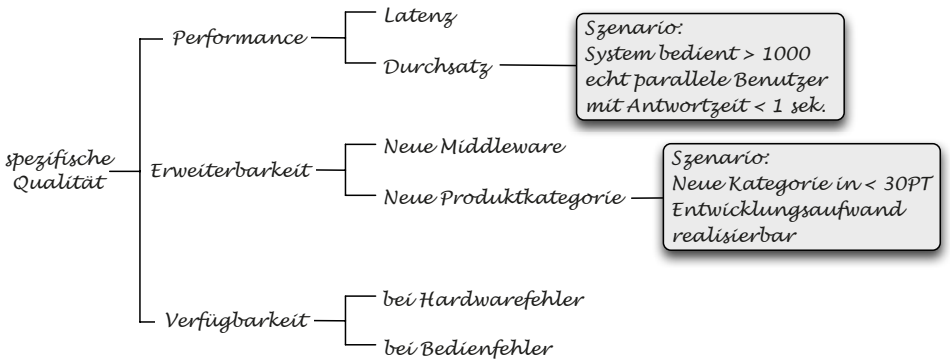

Bild 8.5 Qualitätsbaum mit Szenarien

Oftmals fällt es den Projektbeteiligten leichter, in konkreten Szenarien zu denken als in abstrakten Qualitätsmerkmalen. Beginnen Sie in solchen Fällen ruhig mit einem Brainstorming von Szenarien, und ordnen Sie die zugehörigen Qualitätsanforderungen erst im Nachgang als Baum oder Mindmap.

Mehr als 50 ausführlich formulierte praktische Beispiele zu Szenarien finden Sie im freien arc42-Subprojekt „quality-requirements“ unter:
https://github.com/arc42/quality-requirements

Bereits für mittelgroße IT-Systeme können Sie in solchen Brainstorming-Sitzungen durchaus 30 bis 50 verschiedene Szenarien finden. Im Normalfall wird Ihnen für die detaillierte Bewertung derart vieler Szenarien jedoch keine Zeit bleiben, sodass Sie sich auf einige wenige beschränken müssen. Dabei helfen Prioritäten.

Lassen Sie Ihre maßgeblichen Stakeholder die gefundenen Szenarien nach ihrem jeweiligen geschäftlichen Nutzen oder Wert priorisieren. Ich bevorzuge dabei eine kleine Skala, etwa A = wichtig, B = mittel, C = weniger wichtig.

Machen Sie deutlich, dass es bei den Prioritäten lediglich um die Reihenfolge bei der Bewertung geht! Auch ein für die Bewertung mit C priorisiertes Szenario wird das Entwicklungsteam liefern – sofern irgendwie möglich!

Architektur hinsichtlich der Szenarien analysieren

Mit den (priorisierten) Szenarien für die aktuellen Qualitätsanforderungen und Architekturziele besitzen Sie nun einen Bewertungsmaßstab für Ihre konkrete Architektur. Jetzt folgt die eigentliche Bewertung, die Sie in der Regel in einer kleinen Gruppe gemeinsam mit den Architekten des Systems durchführen. Weitere Stakeholder sind dabei im Normalfall nicht anwesend.[4]

Gehen Sie bei dieser Kernaufgabe gemäß den Prioritäten der Szenarien vor, beginnend mit den wichtigen und schwierigen! Lassen Sie sich in Form eines *Walkthrough* von Architekten oder Entwicklern erläutern, wie die Bausteine des Systems zur Erreichung dieses Szenarios zusammenspielen oder welche Entwurfsentscheidungen das jeweilige Szenario unterstützen.

Als Bewerter oder Auditor einer Architektur untersuchen Sie die zentralen Architekturziele und Qualitätsanforderungen auf Basis der jeweils definierten Szenarien gemeinsam mit dem Architekten des Systems. Lassen Sie sich die zugehörigen Architekturentscheidungen und -ansätze erläutern, und beantworten Sie die folgenden Fragen:

- Welche Architekturentscheidungen wurden zur Erreichung eines Szenarios getroffen?
- Welcher Architekturansatz unterstützt die Erreichung des Szenarios?
- Welche Kompromisse ging man mit dieser Entscheidung ein?
- Welche anderen Qualitätsmerkmale oder Architekturziele beeinflusst diese Entscheidung darüber hinaus?
- Welche Risiken erwachsen aus dieser Entscheidung oder aus diesem Ansatz?
- Welche Risiken gibt es für die Erreichung des jeweiligen Szenarios und der damit verbundenen Qualitätsanforderungen?
- Welche Analysen, Untersuchungen oder Prototypen stützen diese Entscheidung?

Leider muss ich Ihnen an dieser Stelle eine schlechte Nachricht überbringen: Es gibt keinen Algorithmus, der für Sie die Zielerreichung einer Architektur hinsichtlich der Szenarien bestimmt. Ihre Erfahrung (oder die des Bewertungsteams) ist gefragt – auch Ihre subjektive Einschätzung.

Ergebnisse: Risiken + Kompromisse

Als Resultate erhalten Sie einen Überblick über die Güte der Architektur hinsichtlich konkreter Szenarien, und damit auch hinsichtlich der spezifischen Architektur- und Qualitätsziele. Sie finden heraus,

- welche **Risiken** bezüglich der Erreichung oder Umsetzung wesentlicher Qualitätsanforderungen oder Architekturziele existieren;
- welche Szenarien auf jeden Fall (d. h. risikolos) erreicht werden; ([Clements+03] spricht hier von *Non-Risks*);
- welche **Kompromisse** bei Entwurf und Entwicklung eingegangen wurden (beispielsweise wenn zugunsten hoher Performance die Komplexität gesteigert und die Änderbarkeit reduziert wurden).

[4] Ziehen Sie andere Beteiligte hinzu, wenn es während der Bewertung Rückfragen zu den Szenarien (= Qualitätsanforderungen) gibt!

Krönender Abschluss: Maßnahmen definieren

Offiziell gehört es nicht mehr zu ATAM – aber zu jeder realen Bewertung: die Formulierung von Maßnahmen, Strategien oder Taktiken gegen die gefundenen Risiken.

Das ist ganz normale Architektur- und Entwicklungsarbeit – nur können Sie jetzt auf Basis sehr spezifischer Anforderungen (= Szenarien + erkannte Risiken) konstruktive Entscheidungen treffen.

Positive Nebenwirkungen von Architekturbewertungen

Neben den wertvollen Aussagen hinsichtlich Zielerreichung und Risiken bringen qualitative Architekturbewertungen oftmals eine Reihe positiver Nebeneffekte hervor.[5]

- Die maßgeblichen Stakeholder diskutieren über ihre eigenen (Qualitäts-)Ziele.[6] Wesentliche Anforderungen werden dadurch präzisiert – sehr zur Freude der Entwicklungsprojekte.
- Zentrale Architekturentscheidungen werden offengelegt und hinsichtlich ihrer Risiken und eingegangenen Kompromisse transparent.
- Dokumentation wird besser, weil die Bewertung einzelner Szenarien ohne Dokumentation häufig kaum durchführbar ist.

Schließlich kommt es vor, dass maßgebliche Stakeholder sich des hohen Risikos bewusst werden, das durch einige ihrer Anforderungen und Qualitätsziele entsteht.

Bewerten Sie die Architektur so früh wie möglich. Die Ergebnisse qualitativer Architekturbewertung sind für Projekte und Systeme meistens von langfristigem Wert. Investition in Architekturbewertung lohnt sich.*

* Zumal auch kurze Bewertungsworkshops von wenigen Stunden Dauer schon kritische Risiken in Architekturen aufdecken können – erheblich schneller als umfangreiche Code- oder Schwachstellenanalysen.

[5] Obwohl Seiten- und Nebeneffekte im aktuellen Trend „funktionaler Programmierung" ja als böse gelten – bei qualitativer Bewertung sind sie ausdrücklich erwünscht. ☺

[6] Diese Diskussion der Stakeholder hätte bereits in der Analysephase stattfinden sollen. Leider besteht in der Realität hinsichtlich Architekturzielen und Qualitätsanforderungen meistens weder Klarheit noch Einigkeit.

8.2 Quantitative Bewertung durch Metriken

Es gibt eine ganze Reihe quantitativer Metriken, die Sie für Ihre Projekte und Ihren Quellcode ermitteln können. Einige Beispiele:

- Für Anforderungen: Anzahl der geänderten Anforderungen pro Zeiteinheit.
- Für Quellcode: Abhängigkeitsmaße (Kopplung), Anzahl der Codezeilen, Anzahl der Kommentare in Relation zur Anzahl der Programmzeilen, Anzahl statischer Methoden, Komplexität (der möglichen Ablaufpfade, *cyclomatic complexity*), Anzahl der Methoden pro Klasse, Vererbungstiefe und einige mehr.
- Für Tests und Testfälle: Anzahl der Testfälle, Anzahl der Testfälle pro Klasse/Paket, Anzahl der Testfälle pro Anforderung, Testabdeckung.[7]
- Für ganz oder teilweise fertige Systeme: Performance-Eigenschaften wie Ressourcenverbrauch oder benötigte Zeit für die Verarbeitung bestimmter Funktionen oder Anwendungsfälle.
- Für Fehler: Mittlere Zeit bis zur Behebung eines Fehlers, Anzahl der gefundenen Fehler pro Paket/Subsystem.
- Für Prozesse: Anzahl der implementierten/getesteten Features pro Zeiteinheit, Anzahl der neuen Codezeilen pro Zeiteinheit, Zeit für Meetings in Relation zur gesamten Arbeitszeit, Verhältnis der geschätzten zu den benötigten Arbeitstagen (pro Artefakt), Verhältnis von Manager zu Entwickler zu Tester.

Sie sehen, eine ganze Menge von Eigenschaften können Sie quantitativ charakterisieren.

Dennoch: Metriken als Unterstützung der Entwicklung haben sich nach meiner Erfahrung in der Praxis bisher kaum durchgesetzt. Insbesondere messen viele Unternehmen nur punktuell, anstatt Messungen kontinuierlich mit der Entwicklung zu verzahnen.

Beispiel: In einem mittelgroßen Client/Server-Projekt gab der Kunde einem Forschungsinstitut den Auftrag, die etwa 1000 entwickelten und bereits produktiv (und zur Zufriedenheit des Kunden) eingesetzten Java-Klassen durch Metriken zu analysieren. Ziel dieser Untersuchung sollte sein, spätere Wartungsaufwände besser abschätzen zu können.

Im ersten Bericht der Forscher wurde die Qualität der Software heftig kritisiert, weil verschiedene Metriken starke Abweichungen vom wissenschaftlichen Ideal zeigten.

Kunde, Architekt, Entwickler und Projektleiter waren entsetzt – die Software funktionierte einwandfrei, dennoch wurde die Qualität bemängelt. Damit war die Akzeptanz von Softwaremetriken im Entwicklungsteam auf ein Allzeit-Tief gesunken.

Übrigens stellte sich später heraus, dass ein übereifriger Forscher neben den Klassen der Anwendung den gesamten GUI-Framework sowie die zugekaufte Middleware mit analysiert hatte und die bemängelten Metrikdefekte ausschließlich in den Schnittstellen zu diesen Frameworks lagen.

[7] Testabdeckung ist eine verbreitete Metrik, jedoch auch eine sehr kritische. [Binder2000] und [Marick97] beschreiben einige der damit verbundenen Probleme im Detail.

Metriken über Zeit beobachten

Einige Ratschläge zum Einsatz von Metriken, insbesondere Codemetriken:

- Metriken können gute Hinweise für strukturelle Veränderungen von Software geben – über die Funktionsfähigkeit und Qualität zur Laufzeit sagen Codemetriken hingegen nichts aus. Setzen Sie Metriken daher kontinuierlich ein, und beobachten Sie deren Veränderungen über die Zeit.
- Metriken benötigen einen fachlichen und technischen Kontext, um vergleichbar zu sein. Sammeln Sie Daten aus Vergleichsprojekten.
- Metriken können bei unvorsichtiger Anwendung wichtige strukturelle Aussagen über Entwürfe im Zahlendschungel verbergen. Daher: Minimieren Sie die Anzahl der Metriken, und konzentrieren Sie Analysen auf überschaubare Ausschnitte. Weniger ist oft mehr!
- Insbesondere sollten Sie Abhängigkeiten im Quellcode durch ein Werkzeug bei jedem Build* messen und überwachen.

* Sie haben doch hoffentlich einen Daily Build etabliert, inklusive täglicher Durchführung sämtlicher Unit-Tests, oder? Solche Werkzeuge (wie Jenkins, Hudson, Gradle oder Maven) können Metriken mit jedem Build berechnen.

Systeme nur auf Basis von Quellcodemetriken zu bewerten, ist riskant, weil grundlegende *strukturelle* Schwachstellen dadurch möglicherweise überhaupt nicht aufgedeckt werden. So kann es vorkommen, dass Sie qualitative Defizite erst spät im Entwicklungsprozess entdecken.[8] Möglicherweise notwendige Änderungen der Architektur sind dann in der Regel teuer und aufwendig. Daher möchte ich mich auf den folgenden Seiten auf die qualitative Bewertung von Architekturen konzentrieren.[9]

Messungen am laufenden System liefern nur Indikatoren

Metriken können jedoch zur Systemlaufzeit (etwa: Performance-Werte, Ressourcenverbrauch oder auch Fehlerzahlen) wertvolle Indikatoren für Probleme liefern. In jedem Fall müssen Sie auf Basis solcher Messwerte noch *Ursachenforschung* betreiben!

Metriken können Ihnen zeigen, ob Sie Ihren Code gut geschrieben haben. Ob Sie allerdings den richtigen Code haben, können Sie nur durch qualitative Analysen herausfinden.

Beginnen Sie bereits in frühen Entwicklungsphasen mit der Erfassung von Laufzeitmetriken, insbesondere Performance-Daten (etwa: Durchlaufzeiten wichtiger Anwendungsfälle und Ressourcenverbrauch). Ermitteln Sie diese Werte automatisiert, und achten Sie auf Trends.

[8] Iterative Entwicklungsprozesse mit frühzeitigem Feedback vermeiden dieses Problem.

[9] Ich rate Ihnen aber, in Entwicklungsprojekten Codemetriken zur Steigerung der Qualität einzusetzen. Sie sollten in jedem Fall Komplexitäts- und Abhängigkeitsmaße auf Codeebene messen und überwachen – viele Entwicklungsumgebungen unterstützen Sie dabei.

Bewerten Sie insbesondere Architekturen

Von den vielen Artefakten, die innerhalb von Softwareprojekten entstehen, eignen sich Architekturen meiner Meinung nach besonders für die Bewertung: Architekturentscheidungen haben oftmals große Tragweite, sowohl für die zu entwickelnden Systeme als auch für die Projekte und Organisationen, die diese Systeme realisieren, benutzen und betreiben.

Viele dieser Entwurfsentscheidungen müssen Teams oder Architekten unter Unsicherheit oder (vagen) Annahmen treffen – das birgt Risiken! Architekturbewertung hilft, diese Risikofaktoren frühzeitig und spezifisch zu identifizieren.

8.3 Werkzeuge zur Bewertung

Einige Aspekte der Qualität von Systemen können Sie anhand automatisierbarer Untersuchungen des Quellcodes beobachten. Mittlerweile existieren eine Reihe freier (teilweise quelloffener) Werkzeuge, die auch sprachübergreifende Codequalität messen und analysieren können.

Als Beispiel möchte ich Ihnen die SonarQube®-Plattform[10] skizzieren: Frei nutzbar verfügt sie über Plug-ins für alle erdenklichen Programmiersprachen (Java- und C#, C/C++, Groovy, Scala, Erlang, Drools, PL/SQL, Cobol u. v. m). SonarQube kann:

- Untersuchungsergebnisse in einer Datenbank speichern, und die Veränderungen über die Zeit in einer sogenannten „Time Machine“ sichtbar machen. Ein großartiges Instrument, um in Verbesserungs- oder Refactoring-Projekten Erfolge darzustellen!
- doppelten und ungenutzten Code identifizieren;
- Einhaltung von Coding-Styles prüfen;
- Komplexität im Code messen;
- durch seine Client/Server-Architektur in viele Build-Systeme integriert werden;
- relativ einfach über seine offene Plug-in-Schnittstelle um spezielle Prüfungen erweitert werden.

Es gibt alternative Werkzeuge (z. B. PMD, FindBugs, CheckStyle, Squale, Structure101, SotoGraph) – allerdings schützen Sie diese Werkzeuge natürlich nicht davor, den falschen Code richtig zu entwickeln. Grüne Ampeln bei Werkzeugen bedeuten daher nicht unbedingt glückliche Benutzer oder Stakeholder. Dennoch: Meiner Ansicht nach sollten diese Werkzeuge zum Handwerkszeug praktizierender Softwarearchitekten gehören.

[10] *http://sonarqube.org*.

8.4 Weiterführende Literatur

[Bass+03] und [Clements+03] beschreiben die ATAM-Methode (*Architecture Tradeoff Analysis Method*) ausführlich. Meiner Erfahrung nach ist diese Methode jedoch durch die wiederholte Bewertung der Architekturansätze in vielen Fällen zu aufwendig in der Anwendung. Trotzdem legt sie die Grundlage für den Ansatz der Szenario-basierten Bewertung.

[Kruchten+02] fassen den State of the Art bezüglich der Architekturbewertung zusammen; viele Hinweise, wie eine Bewertung praktisch ablaufen kann.

[Henderson-Sellers96]: ausführliche Darstellung objektorientierter Metriken.

[Lorenz94]: praktische Einführung in Metriken für objektorientierte Systeme.