Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Dieter Masak

Der Architekturreview

Vorgehensweise, Konzepte und Praktiken

Springer

Dr. Dieter Masak
plenum Management Consulting GmbH
Hagenauerstraße 53
65203 Wiesbaden
dieter.masak@plenum.de

ISSN 1439-5428
ISBN 978-3-642-01658-5 e-ISBN 978-3-642-01659-2
DOI 10.1007/978-3-642-01659-2
Springer Heidelberg Dordrecht London New York

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandgestaltung: KünkelLopka, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.de)

Danksagung

. . . für Christiane. . .

Dr. Dieter Masak

Inhaltsverzeichnis

Kapitel 1
Hintergründe

Those who cannot remember the past are condemned to repeat it.

The Life of Reason
George Santayana (Jorge Augustín Nicolás Ruiz de Santayana)
1863–1952

Kein Unternehmen, keine Organisation agiert in einem Vakuum, alle sind miteinander verbunden. Viele Menschen, Kontexte, Technologien, Märkte üben Kräfte auf die Aktivitäten eines Unternehmens aus und beeinflussen dessen Systeme, speziell die Softwaresysteme und auch die Entwicklung dieser Systeme nachhaltig. Als Folge davon sind die Organisationen und – direkt oder indirekt – auch ihre Systeme einer großen Anzahl von spezifischen Faktoren ausgesetzt. Unter anderem sind dies:

- Globalisierung,
- konstanter Wandel,
- ressourcenbasierte Strategien,
- stetig wachsende Komplexität der IT-Systeme,
- Veränderung des Kundenverhaltens.

Damit eine Organisation diesen Herausforderungen begegnen kann, muss diese Organisation besondere Fähigkeiten entwickeln, damit durch die gewonnenen Fähigkeiten dem Wandel und der zunehmenden Komplexität begegnet werden kann. Die Fähigkeit System, Software und Architekturen zu verändern, zu steuern und zu beurteilen ist von zentraler Bedeutung für jedes Unternehmen, welches Software in großem Maßstab einsetzt. Dabei ist überhaupt nicht ausschlaggebend, ob das Unternehmen Software selbst entwickelt oder nicht, in beiden Fällen muss es in der Lage sein, seine Architekturen zu steuern. Mit dem Begriff Architektur wird die Abstraktion von Systemen belegt, dabei wurde das Konzept der Architektur der Konstruktion und Planung von Gebäuden entlehnt. Die Architektur von Gebäuden hat eine sehr lange Tradition und ist in unserer Kultur tief verankert. Diese Analogie ist zum einen hilfreich und zum anderen auch hinderlich: Eine Ähnlichkeit zwischen Gebäuden und Systemen wird durch diese Analogie impliziert, welche nicht vollständig gegeben ist, auf Grund der Tatsache, dass viele Systeme durch ihre Dynamik bestimmt sind, während bei Gebäuden die Architektur meist durch ihre Statik festgelegt wird. Auf der anderen Seite erleichtert diese Analogie auch den Einstieg und macht abstrakte Konzepte „fassbarer“.

Heute ist die Software zu dem dominanten Träger von Information in allen Lebensbereichen des Menschen geworden. Diese Dominanz führt zu einer immer stär-

DOI 10.1007/978-3-642-01659-2, © Springer 2010

keren Durchdringung der gesamten Lebens- und Arbeitswelt mit Software und damit zu immer größeren und komplexeren Systemen. Es gibt keine Unternehmen mehr, welche keine Software einsetzen, in der Praxis gilt die Faustformel: Je größer das Unternehmen, desto größer die Softwaresysteme. Diese großen Systeme sind fast immer dadurch gekennzeichnet, dass sie:

- sehr komplex sind,
- sich nichtdeterministisch verhalten,
- eine Mischung aus menschlichen Aktionen und Software darstellen.

Folglich wird es in der Zukunft nicht mehr ausreichen, allein die Softwareentwicklung zu betrachten. Auch andere Wissenszweige wie Psychologie, Soziologie und Systemwissenschaften müssen bei der Beurteilung von großen Softwaresystemen zu Rate gezogen werden.

Alle heutigen Unternehmen, welche sich mit der Erstellung oder dem Einsatz von Software befassen, werden bewusst oder unbewusst mit vielen architektonischen Fragen konfrontiert. Im Falle von Organisationen, die selbstständig Software entwickeln ist dies offensichtlich, aber auch solche, welche nur Software einsetzen werden mit einer Vielzahl von architektonischen Problemen konfrontiert, denn jede Software enthält eine Architektur oder lässt in ihrem Einsatz nur bestimmte Architekturformen zu. Insofern hat jede Software einen Einfluss auf die eingesetzte Architektur und umgekehrt haben aber auch organisatorische Merkmale einen Einfluss auf die Architektur der Software. Leider ist der Begriff der Architektur nicht eindeutig, sondern wird in diversen Kontexten inhaltlich unterschiedlich interpretiert, daher ist es wichtig zunächst einmal den Begriff der Architektur zu klären, bevor der Begriff Architekturreview in den Vordergrund gerückt werden kann.

Alle Systeme, unabhängig von der jeweiligen fachlichen Domäne, werden gebaut um eine Menge an fachlichen Funktionen für den Endbenutzer zu liefern. Ein Fundament für diese Fähigkeit ist die architektonische Integrität der Software, wobei Architektur innerhalb der Software viel mehr mit Nutzbarkeit zu tun hat als mit Ästhetik.[1] Zwar ist es möglich, dass ein schlecht gebautes Softwarepaket die momentanen fachlichen Anforderungen eines Benutzers erfüllt, es wird jedoch in Bezug auf Zuverlässigkeit, Verfügbarkeit und Performanz i. d. R. versagen. Obwohl in der Softwarearchitektur die nichtfunktionalen Aspekte im Vordergrund stehen, sind es die funktionalen Aspekte einer Software, die ihre Existenz überhaupt legitimieren.

Die Nutzung von Architekturen hat innerhalb von Organisationen drei elementare Aufgaben:

- Sie dienen als Vehikel für die Kommunikation zwischen den Stakeholdern. Da die Architektur stets eine Abstraktion eines Systems darstellt,[2] bildet sie eine Basis auf der gemeinsamer Konsens zwischen den diversen Stakeholdern erreicht werden kann. Eine Architektur ist immer eine Art abstrakte, mehr oder minder formale, Beschreibung eines Systems und die Auswirkungen dieser Architektur betreffen jeden, der Kontakt zum System hat.

[1] Im Gegensatz hierzu hat Architektur im Bauwesen sehr viel größere ästhetische Anteile.

[2] Dies geht so weit, dass es eigenständige Blueprints oder auch Marketecture (von **Marke**ting und Archi**tecture**) gibt, die zum „Verkaufen“ und Überzeugen dienen.

- Die Architektur eines Systems ist auch stets eine Manifestation der frühen Designentscheidungen und ermöglicht damit, die gewählten Kompromisse und Implementierungsstrategien aufzuzeigen. Die frühen Designentscheidungen determinieren i. d. R. die meisten der zukünftigen Qualitätsattribute (s. Kap. 2). Dadurch, dass es einen engen Zusammenhang zwischen Architektur und Organisation gibt[3], lässt sich an dieser Stelle auch viel über mögliche zukünftige Organisationen sagen, die diese Software einsetzen können. Außerdem produzieren die meisten der frühen Designentscheidungen eine Reihe von „Zwangsbedingungen" auf ein zukünftiges System, die meist architekturell verankert sind.
- Eine Architektur ist, theoretisch gesehen, eine wiederverwendbare übertragbare Abstraktion eines Systems.

Diese konstruktivistische Sicht auf die Architektur sollte aber mit einem gewissen Maß an Vorsicht gepaart werden, denn aus Sicht des Entwurfs ist eine Architektur auch immer eine Wette; eine Wette darauf, dass das System ein Erfolg wird.[4] Die Qualitäten oder Eigenschaften einer Architektur lassen sich in zwei unterschiedlichen Kategorien einteilen:

- Laufzeitqualitäten – Hier werden Anforderungen daran gestellt, wie gut ein System „das macht, was es tun soll". Dieses „wie gut" wird durch äußere mehr oder minder messbare Größen bestimmt. Typische Ursachen für nichtfunktionale Anforderungen in dieser Kategorie sind:
  - Zwangsbedingungen an das System, die durch die Umgebung bestimmt sind, so bspw. Aufbewahrungszeiten von Unterlagen oder vorgegebene Geschwindigkeiten.
  - Alle Größen, welche durch den Nutzer in seinem Umgang mit dem System bestimmt werden.
  - Produkt- oder Servicefeatures, welche mit anderen Produkten am Markt konkurrieren.
- Entwicklungszeitqualitäten – Solche Qualitäten beziehen sich stets auf die Erstellung oder Veränderung des Systems und versuchen neben rein ökonomischen Aspekten auch Fragen nach Veränderbarkeit und Flexibilität zu beantworten.

Die Einführung und Nutzung von Architekturen innerhalb von Organisationen läuft nicht ohne Probleme ab, denn die Nutzung von Architekturen zur Problemlösung schafft oft neue Probleme, daher auch der Wunsch nach Architekturreviews um den Status-Quo und die Zukunft besser beurteilen zu können. Zur Zeit existieren drei Basisansätze um die Übereinstimmung zwischen Architektur und Implementierung eines Systems zu erreichen, dabei versteht man unter dem Begriff Architekturrekonstruktion die Herstellung einer Architektur – sprich Abstraktion – auf der Grundlage des tatsächlich vorhandenen Systems:

- Konsistenz durch Konstruktion,
- Architekturrekonstruktion aus den statischen Artefakten,

[3] Das sogenannte Conwaysche Gesetz (s. S. 177).

[4] Software development is betting on the future.

- Architekturrekonstruktion aus der Beobachtung des Laufzeitverhaltens eines Systems.

Im ersten Fall wird versucht die Architekturartefakte in die Implementierungsartefakte einzubetten. Dies geschieht entweder durch explizite Nutzung eines Frameworks oder eines generativen Ansatzes wie z. B. MDA (**M**odel **D**riven **A**rchitecture). Speziell für Systeme, welche aus vielen unterschiedlichen Teilen bestehen, ist dies nur sehr schwer möglich. Im zweiten Fall wird primär Codeanalyse genutzt um die tatsächlich implementierte Architektur zu extrahieren, dies stellt damit eine Teilaufgabe des „klassischen" Reengineerings dar. Der dritte Fall ist methodisch der schwierigste, stellt jedoch auch den allgemeinsten Fall – im Sinne der Anwendbarkeit – dar und ist auch auf nichtsoftwarebasierte Systeme anwendbar.

Ein System hat nicht ein einziges Architekturmodell, sondern simultan multiple Architekturmodelle,[5] angefangen von der Laufzeitarchitektur über die Datenfluss- oder die Sicherheits- bis hin zur Codearchitektur. Alle diese Teile können Betrachtungsgegenstand eines Architekturreviews sein. Eines der Ziele eines Architekturreviews zur Bestimmung der Qualitätsattribute kann auch die Evaluierung des Potentials einer Architektur für die tatsächliche Erfüllung der benötigten Eigenschaften sein. Obwohl der Schwerpunkt dieses Buchs auf der Betrachtung von Softwarearchitekturen liegt, lassen sich große Teile der Methodiken und Erfahrungen auch auf nichtsoftwaredominierte Systeme (von denen es in unserer hochtechnisierten Welt kaum noch welche gibt)[6] übertragen, diese Übertragung ist außerdem auf alle Teile einer Enterprise Architektur machbar. Wo dies direkt möglich ist, wird im nachfolgenden stets Bezug auf die Enterprise Architektur (s. Anhang C) genommen und in anderen Fällen wird sich auf die Untermenge der Softwarearchitektur beschränkt.

Im Grunde existieren zwei unterschiedliche Klassen von Architekturreviews (s. Tabelle 3.3–3.4):

- metrikbasierte,
- interviewbasierte.

Beim ersten Typ wird versucht die Artefakte zu messen und so Aussagen über Qualitätsattribute der Architektur zu erhalten, der zweite Typ ist meist qualitativer Natur. Im Bereich der interviewbasierten Architekturreviews werden meist Szenarien (s. Abschn. 3.2) eingesetzt. Qualitative Vorgehensweisen stützen sich sehr stark auf subjektive Interpretationen und sind daher nur bedingt wiederholbar. Unternehmen versuchen sich oft dadurch zu beruhigen, dass sie den Architekturreview von Experten durchführen lassen um somit eine Objektivierbarkeit zu erreichen. Dies ist aber nicht möglich, das Ergebnis ist immer die subjektive Einschätzung des Experten.[7, 8]

---

[5] Im Sinne von Sichten

[6] Brücken, Gebäude oder die Maschine zum Ziehen der Lottozahlen sind i. d. R. softwarefrei, nicht so jedoch Glücksspielautomaten, diese sind voller Software und nicht wirklich zufällig.

[7] Besonders auffällig sind hierbei einige Unternehmensberatungen, die ihre Juniorberater (wenig Erfahrung) mit ausgefeilten Spreadsheets versehen um diesen den Flair eines Experten zu verleihen oder quantitative Messbarkeit vorzutäuschen (s. S. 261).

[8] Alle sogenannten Reifegradmodelle sind subjektiv-qualitative Verfahren.

Quantitative Verfahren beziehen sich immer auf messbare und wiederholbare formale Metriken und versuchen durch die Messung bestimmte Aspekte zu analysieren. Beide Verfahrenstypen komplementieren sich in der Praxis gegenseitig und werden meist zusammen eingesetzt.

Bewertungen und Messungen durchdringen faktisch jeden Aspekt unseres Lebens und unserer täglichen Aktivitäten. Man misst und bewertet täglich eine ganze Reihe von verschiedenen Dingen angefangen von Gewicht über Zeiten oder Entfernungen, aber auch Dinge wie Blutdruck und Temperatur oder Luftfeuchtigkeit. Viele der Entscheidungen, die man trifft, basieren auf dem Ergebnis einer vorangegangenen Messung oder Bewertung. Dabei werden die Bewertungen aus diversen Gründen durchgeführt, unter anderem:

- um festzustellen, wie gut oder schlecht etwas ist, sei es ein Prozess, ein Service oder ein Produkt,
- zur Kontrolle, ob Projekte oder Menschen das „Richtige“ tun,
- zur Bestimmung des notwendigen Budgets,
- zur Vertriebsunterstützung, mit der Frage, ob die gewünschte Zielgruppe auch erreicht wird,
- zum Lernen,
- zur Verbesserung.

Alle diese diversen Zielrichtungen werden durch eine Bewertung unterstützt bzw. erst ermöglicht, wobei sich der Begriff Bewertung noch genauer fassen lässt:

> *Unter dem Begriff der Bewertung versteht man die Einschätzung des Wertes oder der Bedeutung eines Sachverhaltes oder Gegenstandes.*

Eigentlich kann man jederzeit einen Architekturreview durchführen, obwohl der „klassische“ Zeitpunkt für einen Architekturreview nach dem Design und vor der Implementierung des jeweiligen Systems ist. Dieser Zeitpunkt verliert jedoch zunehmend an Bedeutung, da:

- Ein solcher fester Zeitpunkt zwischen Design und Implementierung in der Reinform nur im Wasserfallmodell existiert, bei allen zyklischen Modellen oder auch agilen Vorgehensweisen es einen solchen, singulären Zeitpunkt nicht mehr gibt.
- Im Rahmen von Maintenance es einen solchen Zeitpunkt nicht gibt, denn die Maintenance stellt fast immer eine Art Kontinuum dar.
- Der starke Einsatz von COTS-Software (s. Abschn. 4.4) oft eine Änderung von der intendierten Architektur im Rahmen einer Systemimplementierung erzwingt.
- Aus dem Blickwinkel der Enterprise Architekturen (s. Anhang C) sich auch andere Architektur- oder Softwaresichten bewerten lassen.

Trotzdem kann man Architekturreviews im Rahmen von Projekten in zwei Typen klassifizieren: Zum einen frühe und zum anderen späte Architekturreviews. Charakteristisch für frühe Architekturreviews ist die Tatsache, dass die Architektur noch nicht vollständig spezifiziert ist. Bei frühen Architekturreviews kann noch ein starker Einfluss auf die zukünftige Architektur genommen werden, dafür ist aber die Bewertung nicht vollständig; man spricht hier oft von einer „Protoarchitektur“.

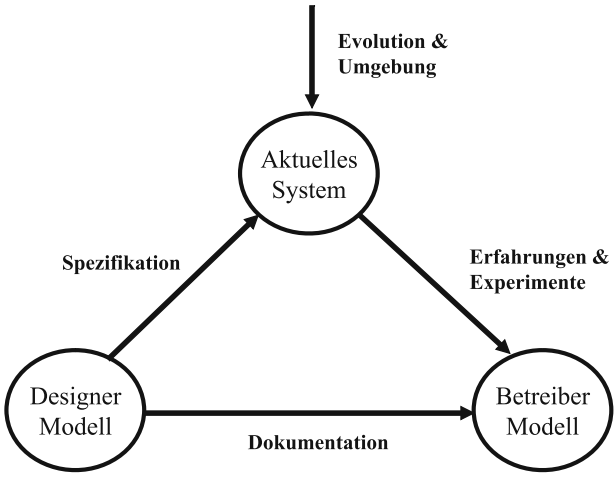

Abb. 1.1 Die unterschiedlichen mentalen Modelle von Designer, tatsächlichem System und dem Modell der Betreiber

1.1 Architekturdefinition

Was ist eine Architektur?[9] Der Begriff der Architektur wird oft, genau wie der Begriff des Architekten, sehr unscharf benutzt, quasi für jede Form der Abstraktion eines Systems. Das vorliegende Buch lehnt sich stark an die ISO[10]/IEC[11] 42010:2007 Norm an:

> Eine Architektur ist: *The fundamental organization of a system. embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.*

Eine solche Definition ist sehr gut anwendbar aus systemtheoretischer Sicht (s. Kap. 5 und Anhang B) und damit auch für alle Formen oder Sichten auf eine Architektur. Wird der Begriff der Architektur aus dem konstruktiven Blickwinkel des Designs heraus betrachten, dann lässt sich eine Architektur auch anders definieren:

> *Eine Architektur ist die Menge an grundsätzlichen Designentscheidungen, welche über ein System gemacht werden oder wurden.*

Wird diese Definition gewählt, so kann man jede Menge an Designentscheidungen als eine Architektur betrachten, mit der Konsequenz, dass sich über die gesamte Lebensdauer eines Systems hinweg diese Menge und damit auch die Architektur ändern kann. Folglich hat eine Architektur auch immer einen temporalen Aspekt.

---

[9] Aus dem Griechischen: $\alpha\rho\chi\iota\tau\epsilon\kappa\tau o\nu\iota\kappa\eta$, abgeleitet von Baumeister oder Gärtner. Der Begriff Architekt wird jedoch erst seit dem Ende der Renaissance im Bauwesen in seiner heutigen Form benutzt.

[10] **I**nternational **O**rganization for **S**tandardization

[11] **I**nternational **E**lectrotechnical **C**ommission

Die meisten praktischen Systeme sind so groß und so komplex, dass sie nie vollständig dokumentiert oder erfassbar sind, außerdem unterliegen sie einem permanenten Wandel, der auch als Evolution bezeichnet wird (s. Abschn. 6.3), so dass es faktisch nie eine feste Architektur gibt. Daher sind in den Fällen großer Systeme meist nur zwei mentale Modelle zugänglich (s. Abb. 1.1):

- Designermodell – Das Designermodell ist eine Idealisierung, welche vor der Implementierung entstanden ist und oft wenig mit der konkreten Ausführung zu tun hat. Zusätzliche Veränderungen, im Allgemeinen als Maintenance bezeichnet, verstärken noch den Unterschied zwischen dem tatsächlichen System und dem Designermodell. Eine andere Form der Idealisierung ist der Sourcecode eines Softwaresystems, der Sourcecode ist im Grunde auch ein Designermodell und muss nicht exakt dem tatsächlichen System entsprechen.[12]
- Betreibermodell – Das Betreibermodell basiert zum einen auf Ausbildungen oder Schulungen am System, meist durch die Designer, und zum anderen auf den „experimentellen" Erfahrungen Einzelner mit der Nutzung des Systems bzw. dessen semantischer Reinterpretation. Hierbei lassen sich deutlich drei Gruppen unterscheiden:

  – Endbenutzer,
  – Systemadministratoren,
  – Systemintegratoren und Operateure.

  Die beiden ersten Gruppen sind im Grunde sehr ähnlich, allerdings mit dem Unterschied, dass sie andere Interfaces nutzen und andere Machtverhältnisse widerspiegeln. Die letzte Gruppe hat die meiste Erfahrung darin, wie das System aus architektureller Sicht auf Veränderung oder auf den Kontakt mit anderen Systemen reagiert.

1.2 Anforderungen

Die Anforderungen an ein System gehören mit zu den wichtigsten Bestandteilen der System- und speziell der Softwareentwicklung überhaupt, schließlich legen sie das Verhalten und die Arbeitsweise des Systems und der Software fest. Obwohl Anforderungen die zentrale Rolle in der Softwareentwicklung spielen, gehört das Anforderungsmanagement zu den Herausforderungen einer jeden Entwicklung. Anforderungen sind Spezifikationen darüber, was gebaut werden soll; sie beschreiben, wie das System oder Teile des Systems sich zur Laufzeit verhalten sollen.

Die Anforderungen werden üblicherweise in eine Reihe von unterschiedlichen Kategorien unterteilt (s. Abb. 1.2). Die bekanntesten Kategorien sind:

[12] Allerdings liegt der Sourcecode dem System meist näher als die ursprüngliche Designdokumentation, was Softwareentwickler mit folgendem Bonmot ausdrücken:

Im Code liegt die Wahrheit...

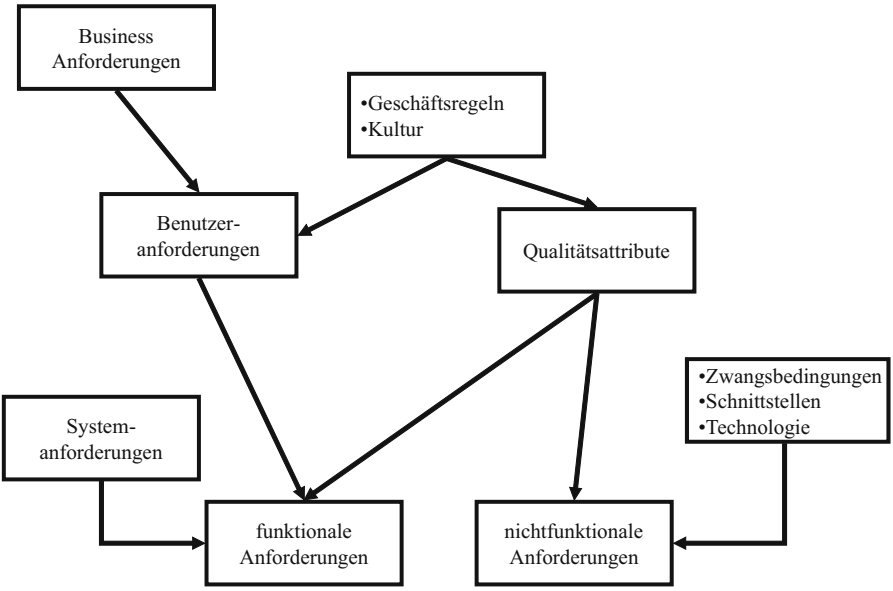

Abb. 1.2 Die Anforderungstypen und ihre Hauptbeziehungen

- Business Anforderungen – Diese Kategorie beschreibt, warum ein Unternehmen ein bestimmtes System oder eine bestimmte Software benötigt. Dabei werden i. d. R. die Vorteile aufgezeigt, die durch den Einsatz des Systems oder der Software erwartet werden.
- Benutzeranforderungen – Diese Anforderungen der Benutzer sind vom Typ des „Was soll geliefert werden?“, wobei beschrieben wird, was der Benutzer mit der Software machen kann, welche Aufgaben und Ziele er durch die Software erfüllen kann. Den verbreitetsten Einsatz finden hier Use Cases, User Stories und Ereignistabellen als Hilfsmittel zur Dokumentation von Benutzeranforderungen.
- Geschäftsregeln und kulturelle Übereinkünfte – Zu dieser Kategorie zählen neben den gesetzlichen Regularien und Regelwerken auch die Übereinkünfte, die in bestimmten Domänen getroffen werden oder durch die kulturelle Erfahrungen[13] gesetzt sind. Zum größten Teil sind diese Anforderungen überhaupt nicht explizit dokumentiert, sondern werden implizit angenommen (s. S. 12).[14]
- Qualitätsattribute – (s. Abschn. 2.2).
- Systemanforderungen – Hierunter wird i. d. R. die Anforderung an das Gesamtsystem verstanden. Die meisten neueren Systeme sind nicht monolithisch, sondern aus einer Reihe von Subsystemen mit dedizierten Aufgaben aufgebaut. In

[13] So ist bspw. die Farbe Schwarz nur im europäisch-amerikanischen Kontext eine Trauerfarbe. Auch Piktogramme oder Leserichtungen, Datumsformate und Kalenderformen zählen zu den kulturellen Übereinkünften.

[14] . . . das weiß doch jeder. . .
. . . das machen wir schon immer so. . .

einem solchen Fall gibt es Anforderungen, die sich auf das Gesamtsystem beziehen. Im Fall von COTS-Systemen (s. Abschn. 4.4) werden diese Anforderungen meist als Produkteigenschaften oder Features bezeichnet.
- funktionale Anforderungen – Diese Anforderungen werden manchmal auch „Behavioural Requirements“ genannt. Sie beschreiben, was der Entwickler bauen sollte und sind meist aus den Benutzeranforderungen abgeleitet.
- nichtfunktionale Anforderungen – Das Hauptgebiet der Anforderungen an die Architektur, z. T. deckungsgleich mit den Qualitätsattributen.
- Zwangsbedingungen – Durch die Wahl bestimmter Implementierungstechnologien oder dem Vorhandensein von Schnittstellen zu anderen Systemen ist jede Software und jedes System einer Reihe von externen Zwangsbedingungen ausgesetzt.

Im Umgang mit Anforderungen zeigt sich in der Praxis eine Reihe von Problemen, die eine große Auswirkung auf das entstehende System haben können. Da die Anforderungen das Fundament für jede Form der Software- und Systementwicklung sind, spielt es keine Rolle, wie gut ein Projektprozess ausgeführt wird oder welche Werkzeuge eingesetzt werden, wenn die Anforderungen nicht richtig sind, ist das System ein Fehlschlag. Daher sollte man den Prozess zur Bestimmung der Anforderungen auch als eine Form des Entdeckungsprozesses sehen und nicht als das bloße „Einsammeln“ von Anforderungen. Außerdem sollte man stets berücksichtigen, dass sich Anforderungen verändern. Diese Veränderung ist recht rapide, im Schnitt verändern sich ca. 2% aller Anforderungen pro Monat. Diese Zahl hört sich zunächst klein an, aber nach 24 Monaten haben sich dann schon über 50% aller Anforderungen geändert, d. h. so etwas wie permanente Stabilität ist nichtexistent.[15] Was das ganze Problem noch verstärkt, ist die Tatsache, dass Anforderungen nie perfekt spezifiziert werden.

Die entstehenden Systeme setzen sich folglich einer gewissen Unschärfe bezüglich der formulierten und antizipierten Erwartungen durch die Benutzer und andere Stakeholder aus. Aber nicht nur der fachliche (funktionale) Aspekt des Systems ist hiervon betroffen sondern auch dessen Architektur.

1.3 Stakeholder und Qualität

Die Zahl der an einer technischen Architektur und damit an einem entsprechenden Architekturreview interessierten Gruppen kann durchaus sehr hoch sein. Diese diversen Gruppen haben im Rahmen einer Organisation durchaus multiple und auch widersprüchliche Interessen, trotzdem lassen sie sich bei Softwaresystemen in drei einfache Kategorien einteilen:
- Softwareerzeuger:
  - IT-Architekt –
  - Softwareentwickler –

[15] Es gibt Ausnahmen, mathematische Algorithmen sind sehr stabil.

  - Tester –
  - Integrator –
  - Maintenanceentwickler –
  - Projektleiter –
  - CIO[16] –
- Softwarenutzer:
  - CSO[17] –
  - Businessmanager –
  - Businessanalyst –
  - Endbenutzer –
- Provider:
  - Systemadministrator –
  - Netzwerkadministrator –
  - Datenbankadministrator –
  - Support und Hotline –

Ein Qualitätsattribut ist eine Eigenschaft einer Architektur, durch die ein Stakeholder in die Lage versetzt werden kann, die Qualität des betrachteten Systems zu beurteilen. Zu den typischen Größen für Qualitätsattribute zählen:

- Performanz,
- Sicherheit,
- Zuverlässigkeit,
- Veränderbarkeit,
- Nutzbarkeit,
- Portabilität,
- Rückwärtskompatibilität,
- Skalierbarkeit,
- Interoperabilität.

Bei den Qualitätsattributen handelt es sich um nichtfunktionale Anforderungen an ein System und unter dem Begriff Qualität wird dann der Grad verstanden, mit dem ein vorhandenes System genau die geforderten Qualitätsattribute erfüllt. In aller Regel sind die einzelnen Attribute zueinander widersprüchlich, mit der Folge, dass nicht alle gleichzeitig erfüllt sein können. Design kann man daher auch als die Suche nach dem „optimalen" Kompromiss im Raum der Qualitätsattribute verstehen. Aus dieser Perspektive heraus betrachtet ist es die Aufgabe eines Architekturreviews festzustellen, ob die Attribute ausreichend erfüllt wurden, oder ob es das Potential zu einer Erfüllung beim gewählten Design gibt. Insofern lassen sich Architekturreviews nicht nur auf vorhandene Architekturen anwenden, sondern können auch während des Softwaredesignprozesses nutzbringend eingesetzt werden. Je früher ein solcher Architekturreview stattfindet, desto früher lassen sich Fehlentwicklungen vermeiden, daher ist ein frühzeitiger Einsatz eines Architekturreviews fast immer sinnvoll.[18]

---

[16] **C**hief **I**nformation **O**fficer. Spötter behaupten CIO sei die Abkürzung für ***C**areer **i**s **o**ver...*

[17] **C**hief **S**ecurity **O**fficer

[18] Pathologische Ausnahmen sind Entwicklungsprojekte, die exakt einer Referenzarchitektur folgen, wobei hier die Frage nach der Adäquatheit der Referenzarchitektur gestellt werden kann.

1.4 Architekturwissen

Das Konzept des Architekturwissens besteht aus vier verschiedenen Komponenten (s. Abb. 1.3), welche auch als die Kernkomponenten des Architekturwissens bezeichnet werden. Diese vier Kernkomponenten sind:

- Prozesse – Architekturen beeinflussen die Prozesse einer Organisation und umgekehrt werden die Architekturen durch die organisatorischen Prozesse beeinflusst. Der Zusammenhang zwischen Organisation und Architektur wird durch das Conwaysche Gesetz (s. S. 177) beschrieben.
- Personen – Viele verschiedene „Stakeholder“ sind innerhalb von architekturalen Prozessen beteiligt bzw. haben ein valides Interesse an den eigentlichen Architekturen. Ein notwendiger Schritt ist es hier Kompromisse zwischen den verschiedenen Beteiligten zu etablieren.
- Entscheidungen – Damit man zu einem Architekturentwurf kommen kann, sind Entscheidungen notwendig. Diese benötigen oft die Kompromisse der Stakeholder, damit sie überhaupt stattfinden. Das Wissen über diese Entscheidungen ist wichtig, da sie Grundlagen für den Architekturentwurf bilden bzw. diesen bis zu einem gewissen Grad auch legitimieren.
- Design – Der eigentliche Architekturentwurf, das Design, wird meist als Diagramm dargestellt und bezeichnet einen Kulminationspunkt des Architekturwissens.

In Bezug auf die Architektur muss hinsichtlich des Wissens zwischen dem Problemraum und dem Lösungsraum unterschieden werden. Der Problemraum spannt die Menge der signifikanten Architekturanforderungen auf, während der Lösungsraum durch Referenzarchitekturen und Architekturentscheidungen immer stärker

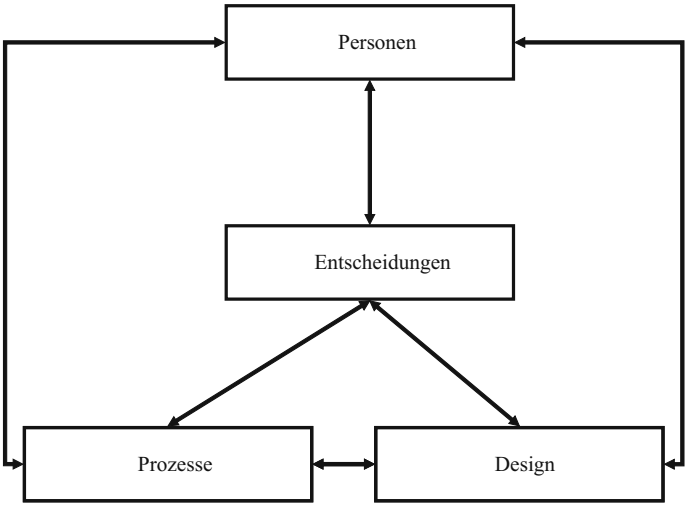

Abb. 1.3 Die vier Teile des Architekturwissens

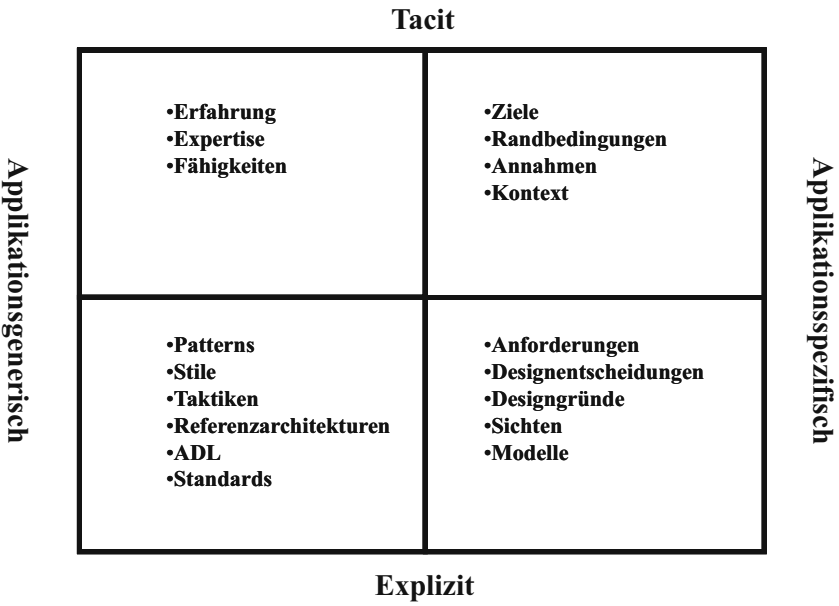

Abb. 1.4 Die vier Dimensionen des Architekturwissens

eingeengt wird. Architekturentscheidungen überdecken stets sowohl Problem- als auch Lösungsraum, indem sie den Problemraum erweitern und den Lösungsraum verengen, da jede Architekturentscheidung Folgeanforderungen nach sich ziehen kann und gleichzeitig die Menge der möglichen Lösungen durch ihre Existenz einschränkt. Der Entscheidungsprozess als solcher läuft daher meist iterativ ab, bis eine hinreichend gute Lösung gefunden wurde, oder der momentane Zyklus als nicht zielführend verworfen wird.

Neben dem eigentlichen Prozess muss das angehäufte Wissen über Architektur bzw. über Architekturentscheidungen auch vorhanden und zugänglich sein. In diesem Kontext lassen sich drei Ebenen des Wissens unterscheiden:

- Implizites Wissen[19] – In vielen Fällen sind weder die Architektur und noch weniger die einzelne Architekturentscheidung explizit beschrieben oder modelliert, sondern dieses Wissen wird als gegeben und allgemein verfügbar angenommen – meist im Kopf eines einzelnen Architekten.[20] Dieses implizite Wissen explizit zu machen ist sehr aufwändig und auch z. T. – meist aus „politischen“ Gründen – nicht gewünscht. Daher sind in den meisten Fällen große Teile der Architektur überhaupt nicht dokumentiert.
- Dokumentiertes Wissen – Als Grundlage der Dokumentation der Architektur werden meist die Architektursichten (s. Abschn. 6.1) für die diversen Stakeholder genutzt.

[19] Tacit Knowledge

[20] Geheimnisse für sich zu behalten kann auch die Machtposition des Architekten im Unternehmen stärken.

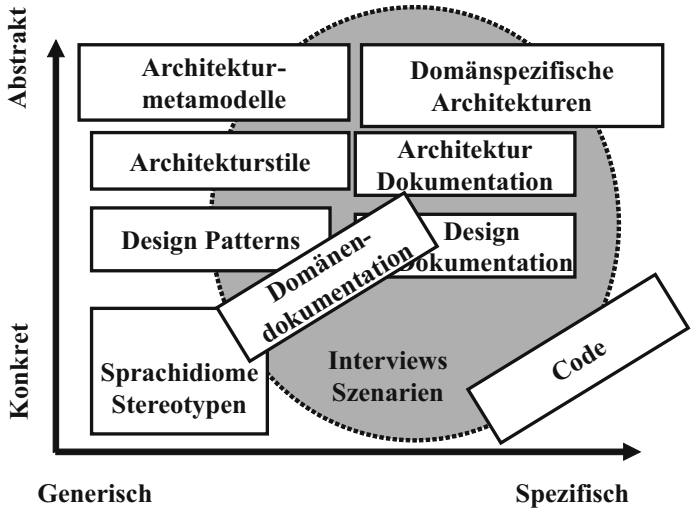

Abb. 1.5 Der Informationsraum, aus dem ein Architekturreview Kenntnisse über das System gewinnen kann

- Formalisiertes Wissen – Dies ist eine spezielle Form des dokumentierten Wissens, hierbei werden vorgeschriebene Diagramme und Sprachen (ADL)[21] zur Dokumentation eingesetzt (s. Abschn. 2.8).

Es ist sehr selten, dass sich eine Architektur in der Praxis vollständig dokumentiert und in sich konsistent vorfinden lässt; in den meisten Fällen ist die Dokumentation weder hinreichend gut noch vollständig oder aktuell, sondern geprägt durch folgende Phänomenologie:

- heterogen – Es existiert nicht „die Architekturdokumentation", sondern die Architektur ist meist versteckt in diversen Dokumenten mit unterschiedlichster Notation und Syntax.
- unvollständig – Die meisten Dokumentationen, die im Laufe der Entwicklung eines Systems entstehen, sind geprägt vom Gedanken bestimmte Aspekte im nächsten Schritt der Entwicklung voranzutreiben, daher sind sie, im Sinne einer Überdeckung, meist unvollständig.
- verändernd – Alle realen Systeme verändern sich permanent, mit der Folge, dass auch die Dokumentation sich mit verändert – allerdings meist sehr zeitversetzt und a posteriori.

Die große Menge an implizitem Wissen im Bereich der Architektur führt zu Problemen im allgemeinen Design. Oft beobachtbare Effekte durch zu starkes implizites Wissen sind hier:

- Designentscheidungen – Die Designentscheidungen sind meist querschnittlich und stark untereinander verwoben. Typischerweise haben Designentscheidungen

[21] **A**rchitecture **D**escription **L**anguage

Auswirkungen auf viele Teile des entstehenden Systems; wenn diese nicht explizit in der Architektur dokumentiert werden, dann ist das architekturelle Wissen notwendigerweise fragmentiert. Außerdem ist es sehr schwer, einmal getroffene Entscheidungen nachträglich zu verändern, da die Auswirkungen oft unbekannt sind.

- Designregeln – Die Designregeln werden oft verletzt, wenn diese nicht explizit dokumentiert werden. Die Folge ist eine „Verwässerung" der Architektur.[22]
- Überflüssiges – Obsolete Designentscheidungen werden nicht eliminiert, dadurch steigt die Entropie innerhalb des Systems auf Dauer schneller an. In fast allen heutigen Designvorgehensweisen wird versucht die Rücknahme von Designentscheidungen zu vermeiden, da man sich über die Auswirkungen einer Rücknahme nicht klar ist.[23]

Eine langfristige Auswirkung dieser Phänomene ist, dass solche Systeme sehr teuer zu verändern sind und die Systeme sehr schnell fragil werden.

1.5 Architekturkompetenz

Für die Erstellung von Software ist eine Architekturkompetenz auf Seiten des Einzelnen, am Softwareprozess beteiligten Individuums und eine analoge Kompetenz auf Seiten der Organisation notwendig. Ohne diese beiden Formen der Architekturkompetenz ist eine zielgerichtete Softwareentwicklung nicht möglich. Im Bereich der Baubranche geht dies so weit, dass die entsprechende Kompetenz in den Berufsbezeichnungen verankert wird:

- Architekt als generischer Begriff,
- Innenarchitekt als eine Person, die nur das Innere von Häusern entwirft,
- Baustatiker als Zuständiger für das Qualitätsattribut Stabilität,
- Architekturbüro und Ingenieurbüro für die jeweiligen organisatorischen Kompetenzen.

Von einem solch hohen Grad an Spezialisierung und Verankerung in Rollen und Kompetenzen ist man in der Softwarebranche noch sehr weit entfernt. Aber trotz aller aktuellen Defizite muss man in der Lage sein, die Architekturkompetenz sowohl einzelner Personen als auch ganzer Organisationen messen zu können, um überhaupt die Fähigkeit zu haben, eine solche Kompetenz zu steuern. Beide Kompetenzformen sind jedoch eng miteinander verwoben, denn eine Organisation ist stets aus Individuen aufgebaut und erst diese ermöglichen eine Veränderung der Organisation selbst. Auf der anderen Seite existiert eine Organisation nicht in einem

[22] Besonders unangenehm ist dies im Fall der Verletzung von Frameworks. So ist zu beobachten, dass beim Wechsel eines Frameworks oder einer Plattform (z. B. von COBOL nach J2EE oder von einem Fat-Client nach einem Web-Client) die „alten" Designmaximen mitgenommen werden. Es entsteht dann der Eindruck, dass gegen das „neue" Framework angekämpft werden muss.

[23] EXtreme Programing ist trotz des angestrebten Refaktorings hierbei keine Ausnahme, da sich dieses meist auf lokale Strukturen bezieht. Auch im eXtreme Programing werden selten Entscheidungen mit systemweiter Durchdringung gefällt.

Vakuum, sondern sie ist stets in ein komplexes Geflecht aus anderen Organisationen, politischen, markttechnischen und soziologischen Strömungen eingebettet, welche wiederum Einfluss sowohl auf konkrete Architekturen als auch auf Kompetenzen haben. Insofern reicht eine alleinige Betrachtung eines fertigen Architekturartefakts nicht aus, um die Ursachen für Defizite aufzudecken, diese dienen i. d. R. als Symptome für ein viel tiefer liegendes Problem.[24]

Bei der Betrachtung der individuellen Architekturkompetenz wird recht schnell klar, dass für die Kompetenz des Individuums vier Eigenschaften eine wichtige Rolle spielen:

- Aufgaben,
- Fähigkeiten,
- Wissen,
- Performanz.

Ohne eine explizite Betrachtung aller dieser Eigenschaften lässt sich im Allgemeinen sehr wenig über die Kompetenz eines einzelnen Architekten aussagen. Für die Bedeutung der organisatorischen Aspekte der Architekturkompetenz stellt sich eine Reihe von Fragen:

- Wie sind die Tätigkeiten der Architekten in die Prozesse der Organisation eingebettet?
- Wie werden diese Architekturaktivitäten koordiniert?
- Wie wird das jeweilige individuelle und organisatorische Wissen vermittelt und weitergegeben?
- Wie werden die Artefakte des Architekturprozesses benutzt und verwaltet?

Zu den Faktoren, die auf Seiten der Organisation die meisten Limitierungen produzieren, gehört die Frage der Koordination von Prozessen. Mangelnde Koordinationsfähigkeit innerhalb der Prozesse äußert sich nicht nur bei der Architektur, sondern bei allen Produkten oder Services, die von solchen Organisationen erzeugt werden. Allerdings kann das Defizit bei der Architektur besonders stark auftreten. Dieses Phänomen beruht auf dem relativ jungen Alter expliziter Architekturtätigkeiten in der IT, in anderen Prozessbereichen gibt es innerhalb der Organisation meist eine Reihe von „Work-Arounds“ oder auch „Shortcuts“ um mit den Koordinationsdefiziten umgehen zu können. Je jünger eine Disziplin ist, desto geringer ist der Anteil an implizitem Wissen über diese Disziplin bei allen Beteiligten, mit der Folge, dass Defizite besonders transparent werden.

Auch die Organisation als Ganzes hat einen großen Einfluss auf die Architektur bzw. auf die gewählte Lösung, denn i. d. R. wurde in der Vergangenheit einer Organisation schon eine gewisse Menge an Investitionen in den Bereichen Ausbildung, Infrastruktur und Architektur vorgenommen. Diese wiederum erlaubt, bei endlichem Aufwand, oft nur eine beschränkte Menge an Architekturvarianten, daher macht die Aussage: „Dies ist eine gute Architektur“, wenn überhaupt, nur innerhalb eines bekannten organisatorischen Kontextes Sinn.

[24] Selbstverständlich gibt es hier auch Ausnahmen: Selbst der beste Architekt oder der beste Designer macht Fehler!

Die Softwarearchitektur als eine spezielle Architekturform liegt an einem Schnittpunkt zwischen Programmiersprachen auf der einen Seite und der menschlichen Gesellschaft auf der anderen. Programmiersprachen an sich sind präzise und formal, wohingegen Menschen durchgehend unpräzise und widersprüchlich erscheinen. Die größte Herausforderung ist es genau diese Ungenauigkeit zu modellieren und in ein exaktes Modell zu gießen. Man kann diese Situation mit der Physik vergleichen, bei der die vielfältige Natur in mathematisch mehr oder minder geschlossene Modelle abgebildet wird. In der Physik ist dieses Vorgehen sehr erfolgreich gewesen, ganz im Gegensatz zur Softwareentwicklung. Schon in den Siebzigerjahren des letzten Jahrhunderts war man der Ansicht dieses Problem gelöst zu haben, aber es wird heute immer noch diskutiert. Einer der Gründe für diese Diskrepanz ist, dass die Physik ein einzelnes System – die Natur – modelliert, während die Softwareentwicklung hingegen Methoden kreiert um Systeme zu modellieren. Als solches sollte eine gute Designmethode in der Lage sein, genau wie die Physik Systeme zu modellieren. Aber im Gegensatz dazu beschäftigt sich die Softwareentwicklung mit dem Entwurf neuer – bis dato unbekannter – Systeme.

Neben dem Wachstum in der Komplexität der Systeme und Software (s. Abschn. 6.3) werden Veränderungen an bestehenden Systemen sehr selten aufgezeichnet, mit der Folge, dass jede Form der Dokumentation auf Dauer von der eigentlichen Implementierung des Systems abweicht.[25, 26] Viele Softwareentwickler, welche mit der Maintenance beschäftigt sind, ignorieren jedwede Form der Dokumentation, da sie oft falsch oder unvollständig ist. Oft wird an dieser Stelle der feinsinnige Unterschied zwischen einer „As-Is“- und einer „As-Designed“-Architektur gemacht, ein solcher Unterschied ist jedoch irreführend und veranlasst die Beteiligten nur dazu in Legitimationsargumenten zu verharren. Viel wichtiger für die System- und Softwarearchitektur ist die Ist-Architektur! Falls jedoch der eigentliche Entwicklungsprozess hinterfragt werden soll, so kann obiger Vergleich herangezogen werden.

Trotz all dieser Überlegungen sollte eins beachtet werden: Die Erforschung von Legacysystemen oder von Frameworks erzeugt den Eindruck, dass Architektur als Phänomen gut verstanden ist. Dem ist nicht so! Die Techniken, die zum Bau von Software eingesetzt werden, unterliegen auch heute einer konstanten Evolution und Veränderung; mit der Evolution dieser Techniken muss sich ex- oder implizit auch der Einsatz und – vor allen Dingen – das Verständnis von Architektur verändern.

1.6 Instrumentalisierung

Wie jede andere Form des Gutachtens oder eines Qualitätssicherungsprozesses kann auch ein Architekturreview zur Durchsetzung organisatorischer oder persönlicher Ziele einzelner Führungskräfte instrumentalisiert werden. Meist wird für den Architekturreview ein externer „Experte“ eingesetzt, dessen Aussagen bei den Betrof-

[25] Ob die erstmalige Dokumentation je einem Zustand der Implementierung entsprochen hat, ist sowieso meist fraglich.

[26] Die meisten Programmierer hassen es Dokumentationen zu verfassen.

fenen ein gewisses Gewicht haben. Dies weckt, im schlimmsten Fall, das Interesse von Führungskräften einen „getürkten“ Architekturreview zu initiieren oder die Ergebnisse eines zunächst objektiv verstandenen Architekturreviews als Mittel zur Unterstreichung ihrer Position einzusetzen. Diese beiden Formen der Motivation finden sich in der Praxis gar nicht so selten, allerdings nicht in Reinform, sondern meist als ein zusätzlicher Aspekt für einen angestrebten Architekturreview. Der Reviewexperte sollte sich dieser Problemlage stets bewusst sein – dies bedeutet nicht die Produktion von Gefälligkeitsreviews, sondern des aktiven Umgangs mit einer „hidden Agenda“ auf Seiten des Auftraggebers. In den meisten Fällen wird jedoch erst das Ergebnis eines Architekturreviews reinterpretiert und für organisatorische Zwecke unausgesprochen und oft zeitversetzt genutzt. Dieser zweite Fall lässt sich a priori nicht verhindern und muss als solcher leider hingenommen werden. Ein mit der Instrumentalisierung verwandtes Phänomen bezüglich des Ergebnisses eines Architekturreviews bzw. der Findung von Ergebnissen ist die menschliche Eigenschaft, die eigenen Vorurteile bestätigen zu wollen. Der Empfänger eines Reviewergebnisses tendiert dazu, seine eigenen Vermutungen über die Qualitäten und Eigenschaften der reviewten Architektur besonders intensiv wahrzunehmen und andere Ergebnisse zu unterdrücken. In anschließenden Diskussionen werden dann diese Teilergebnisse selektiv genutzt um bestimmte Positionen zu unterstreichen, ohne dass ihr jeweiliger Kontext berücksichtigt wird. Solche Mechanismen lassen sich nicht verhindern! Viel problematischer sind jedoch die Vorurteile des Reviewexperten, denn auch dieser versucht seine Vorurteile bestätigt zu bekommen. Gegenüber diesem Phänomen helfen nur folgende Maßnahmen, wenn sie zusammen eingesetzt werden:

- Einsatz mehrerer Reviewexperten simultan oder sequenziell,
- Nutzung eines Architekturreviewmodells,
- Quantifizierung von Architektureigenschaften.

Eine andere Form der Instrumentalisierung ist ein zertifikationsorientierter Architekturreview. Typischerweise wird eine Zertifizierung von Dritten vorgenommen, so z. B. TÜV[27] oder ISO-9000-Zertifizierer. Diese Form der Architekturreviews dient i. d. R. zur Absicherung Dritter, nicht zur eigentlichen Qualitätssicherung des Herstellers. So zertifiziert *Microsoft* die Fähigkeit einer Software zur Laufzeit unter der *Windows*-Plattform laufen zu können oder das CMMI-Modell[28] zertifiziert den Reifegrad einer Softwareentwicklungsorganisation, damit diese Aufträge vom amerikanischen Verteidigungsministerium erhalten kann und ähnliches mehr. Grundlage solcher Formen von Zertifikationen ist die Existenz eines „anerkannten“ Standards.[29]

[27] **T**echnischer **Ü**berwachungs-**V**erein, entstand in Deutschland aus den lokal organisierten Dampfkessel-Überwachungs- und Revisions-Vereinen.

[28] **C**apability **M**aturity **M**odel **I**ntegration

[29] Viele Menschen empfinden Standards als etwas „Gottgebenes“, auf das man nur begrenzt Einfluss nehmen kann. Dabei sind die meisten Standards sehr wohl aktiv von einzelnen Akteuren gesteuert worden. Betrachtet man die vergangenen Jahre, dann versuchen immer wieder einzelne Unternehmen, Standards vorzugeben. Der Grund dafür liegt meist im eigenen ökonomischen Nutzen. Was als Standards bezeichnet wird, ist für die großen Unternehmen ein „Machtmittel“.

Ein anderer möglicher Grund für ein Architekturreview ist die Tatsache, dass Fehler im Design und in der Architektur sehr kostspielig sind, da typischerweise 50–70% aller Kosten durch die Wahl der Architektur bedingt sind. Außerdem wachsen die Kosten für die Fehlerbeseitigung pro Entwicklungsphase um jeweils eine Größenordnung an.[30] Mögliche Architekturdefizite sind i. d. R. sehr kostspielig, da dieser Fehler in der Architektur sich mit den diversen fachlichen Implementierungen multipliziert und daher um Größenordnungen in den Kosten anwachsen kann. Zusätzliche Steigerungen der Kosten können sich im Falle von Architekturfamilien (s. S. 152) ergeben, da sich dann diese fehlerhaften Strukturen in einer ganzen Reihe von implementierten Architekturen wiederfinden lassen.

In vielen Disziplinen der exakten Wissenschaften beeinflusst der Beobachter durch seine Beobachtung das Phänomen an sich. Besonders in der Quantenmechanik ist dies ersichtlich, wo die Beobachtung eines Teilchens zur Fixierung seines quantenmechanischen Zustands führt. In den Sozialwissenschaften ist dies unter dem *Hawthorne*-Effekt[31] bekannt. Aber auch auf dem Gebiet der Softwareentwicklung existieren solche Beobachtereffekte:[32]

- Dadurch, dass Benutzer beobachtet werden oder die Software zu deren Beobachtung instrumentiert wird, wirkt die zu untersuchende Software weniger „reaktiv“ bzw. der Benutzer reagiert bewusster, mit der Folge, dass das tatsächlich ausgeführte Szenario nicht dem „üblichen“ Szenario entspricht.
- Tracing-Operationen, speziell durch Debugger, können die Ausführungsreihenfolgen in Programmen verändern und so zu falschen Schlüssen führen.
- Die vermeintliche Architektur des Systems dient als Strukturierungshilfsmittel des Beobachters, so dass am Ende diese auch als erreicht gilt.[33]

1.7 Gute Architekturen

Was bedeutet es, wenn eine Architektur gut genannt wird? In einigen Fällen ist dies eine sehr subjektive Bewertung des Designers, welcher damit zum Ausdruck bringt, dass er das Ergebnis seiner Tätigkeit bewundert haben möchte. In anderen Fällen löst eine Architektur Probleme, welche der Betrachter, der dieses Werturteil abgibt, schon lange in einem anderen Kontext hatte, und erzeugt somit das Bild einer guten Lösung; ob diese Architektur allerdings die Probleme löst, für die sie angetreten ist, sei dahingestellt. Für einen künstlerischen Zugang mag dies ausreichen, für einen wissenschaftlichen auf gar keinen Fall.[34]

---

[30] Faustregel: Ein Fehler, der in der Architektur einen Beseitigungspreis von 1 hat, erzielt in der Produktion einen Preis von 100.

[31] Benannt nach den *Hawthorne Works* in Chicago, einer Firma für elektronische Geräte. Hier wurden Produktivitätsstudien durchgeführt, bei denen es sich herausstellte, dass die Probanden sehr stark darauf reagierten Teil einer Studie zu sein und nicht auf die Parameter der Studie.

[32] . . . obwohl die wenigsten dies gerne zugeben.

[33] If you got a hammer, everything looks like a nail.

[34] In gewisser Weise spiegelt sich hier das veränderte Selbstbild von Softwaredesignern oder -architekten wider: Auf der einen Seite empfinden sie sich als „Künstler“, auf der anderen Seite würden sie gerne als Ingenieure wahrgenommen werden.

Die Frage ob eine gegebene Architektur gut oder schlecht ist, lässt sich nicht direkt beantworten. Gut oder schlecht sind absolute Bewertungen, die für Architekturen sowieso nicht möglich sind.[35] Eine Architektur lässt sich nur in einem festgelegten Kontext beurteilen, d. h.: Wie gut löst eine Architektur ein vorgegebenes Problem? Selbst diese eingeschränkte Frage lässt sich nicht mit gut oder schlecht beantworten! Für diese Beobachtung gibt es diverse Gründe:

- Es existiert keine singuläre, allgemein akzeptierte Metrik um eine Architektur zu beurteilen. Selbst wenn man in der Lage ist metrische Größen einer gegebenen Architektur zu bestimmen, so ist die Bewertung des Ergebnisses meist sehr unklar.
- In fast allen Fällen sind die Anforderungen an die Architektur grob und z. T. nur schwer zu fassen.
- Die Anforderungen sind vielfältig und – i. d. R. – nicht widerspruchsfrei, mit der Folge, dass die Architektur einen Kompromiss aus diesen Widersprüchen bilden muss.
- Die Zielsetzungen einer Architektur sind heute noch nicht quantifizierbar, was sie explizit nicht messbar macht (s. Anhang A).
- Viele Zielsetzungen beziehen sich auf ein langfristiges Ziel, welches in der Zukunft liegt und dessen Erreichung von vielen Faktoren außerhalb der Architektur abhängt. Hier kann eine Bewertung nur eine subjektive Extrapolation darstellen.[36]

Trotz aller Einschränkungen lassen sich dennoch eine Reihe von Eigenschaften für die Architektur fordern, die zumindest ein Indiz über die Adäquatheit einer gewählten Architektur liefern. Solche Eigenschaften werden als Qualitäten (s. Abschn. 2.1) bezeichnet. Allerdings beschränken diese Qualitäten sich auf die nichtfunktionalen Aspekte einer Architektur, so dass die Angemessenheit gegenüber dem fachlichen Problem zusätzlich in Betracht gezogen werden muss.

Eine Architektur ist immer das Ergebnis einer Reihe von geschäftlichen und technischen Entscheidungen, sie entsteht daher nicht im luftleeren Raum, sondern wird durch diverse Faktoren und Kräfte beeinflusst. Eine der Schwierigkeiten hinter der Entwicklung einer Architektur ist die Tatsache, dass die Summe der Anforderungen an ein System eine Menge an Eigenschaften des Systems explizit macht, aber eben auch nicht alle. Die Nichterfüllung der impliziten Eigenschaften macht ein System genauso wenig nutzbar wie die Verletzung von expliziten Anforderungen. Eine der großen Herausforderungen für einen Architekten ist das „Explizitmachen“ der impliziten Anforderungen. Doch nicht nur die Anforderungen sind unterschiedlich, auch die Personengruppen, die diese Anforderungen stellen, sind dies (s. Abschn. 1.3). Ohne dass jedoch die entsprechenden Stakeholder bekannt sind, lässt sich eine Architektur nur sehr beschränkt begutachten oder beurteilen.

Im Gegensatz zu der Lehrmeinung, die in Seminaren und Vorlesungen vermittelt wird, ist die Softwareentwicklung und damit das Softwareprodukt nicht mit

[35] Selbst im Bauwesen, wo man über eine dreitausendjährige kulturelle Tradition hat, lässt sich die Bewertung gut oder schlecht nicht absolut anwenden. Ein Parkhaus mag für einen Autofahrer sehr gut sein, wohingegen es als Wohnraum denkbar ungeeignet ist.

[36] Software development is betting on the future.

dem ersten Release beendet. Ganz im Gegenteil, alle ernstzunehmenden Softwareprodukte durchlaufen eine Anzahl von Releases, Versionen und Bugfixes. Dies bedeutet, dass der initiale Entwurf zwar relevant, aber aus ökonomischer Perspektive nicht unbedingt der wichtigste Fokus ist. Auf Dauer stellt sich der Maintenanceprozess als ein besonders wichtiger Prozess für den Lebenszyklus von Software dar.

Was kann man von einem Architekturreview als Ergebnis erwarten? Die Architekturreviews produzieren zunächst einfach Informationen, aber im Besonderen versuchen sie zwei Fragetypen (oft in leicht abgewandelter Form) zu beantworten:

- Ist eine gewählte Architektur geeignet für ein bestimmtes Zielsystem?
- Welcher von mehreren konkurrierenden Architekturvorschlägen passt am besten auf die Aufgabenstellung?

In beiden Fragetypen stellt sich die Schlüsselfrage danach, wie gut eine Architektur „passt". Passen in diesem Kontext bedeutet, dass:

- das entstehende oder vorhandene System den geforderten[37] Qualitäten entspricht, und
- das entstehende oder vorhandene System bau- oder veränderbar ist.[38]

Eine Antwort auf solche Fragetypen kann nicht eine einfache Zahl sein,[39] sondern nur qualitativ in der Form, dass Risiken, Stärken, Schwächen oder Chancen einer oder mehrerer Architekturen aufgezeigt werden.

Es gibt eine Reihe von typischen Fragen, die man sich in Bezug auf Architektur und Architekturstil stellen kann. Folgende Fragestellungen sind hier nur exemplarisch aufgeführt:

- Passt der Architekturstil zur Applikationsdomäne?
  Einfach einen Architekturstil zu benutzen, weil er momentan Mode ist, ist der falsche Weg, obwohl dies recht häufig vorkommt. So sollte man sich Fragen, ob eine Verteilung überhaupt Sinn macht oder ob Services bzw. eine SOA (Service Oriented Architecture) immer die adäquate Antworten auf eine fachliche Problemstellung darstellen.
- Liefert der Architekturstil die benötigten Qualitätsattribute bzw. unterstützt er sie?
  Architekturstile versuchen stets bestimmte Qualitätsattribute zu verstärken, dies geht jedoch nicht ohne eine Form des Kompromisses bei anderen Attributen, mit der Folge, dass nicht jede Fähigkeit gleich gut durch jeden Architekturstil unterstützt wird.

---

[37] Wenn es keine Forderungen an das System gibt, ist jede Architektur a priori gleich gut oder gleich schlecht.

[38] Insbesondere wenn die Veränderungen in einen vernünftigen Kosten- und Zeitrahmen fallen.

[39] *The Answer to Life, the Universe and Everything is 42.*

The Hitchhiker's Guide to the Galaxy
Douglas Adams, 11.3.1952–11.5.2001

- Hat das Entwicklungsteam überhaupt Erfahrung im Umgang mit einem spezifischen Architekturstil?
- Liefert die Architektur die gewünschten Fähigkeiten?
- Wie einfach oder wie komplex ist die Architektur?

Die Fragestellung, ob eine Architektur gut oder schlecht ist, lässt sich nicht wirklich beantworten, da sich Architekturen einer solchen Form der Bewertung entziehen. Man sollte viel eher fragen, ob eine Architektur adäquat zu ihren diversen und widersprüchlichen Anforderungen ist!

1.8 Herausforderungen, Ergebnisse und Auswirkungen

Es gibt im Rahmen eines Architekturreviews auch eine Reihe von Nebenwirkungen, die eine Organisation oder ein Projekt positiv beeinflussen können:

- Die Stakeholder reden miteinander. Der Architekturreview ist oftmals die erste Chance, bei der sich alle am System interessierten Stakeholder treffen und sich gegenseitig als das wahrnehmen, was sie sind: Eine Gruppe von Menschen, die am Erfolg eines Systems interessiert sind.[40] Im Rahmen das Architekturreviews wird ein Forum geschaffen, in dem die unterschiedlichen Interessen und Motivationen erläutert und legitimiert werden. In einer solchen Umgebung lässt sich ein Kompromiss oft leichter erreichen als über den Austausch von Anforderungsdokumenten oder elektronischen Mails.
- Die Stakeholder werden gezwungen ihre Ziele und Qualitätsansprüche zu definieren. Durch einen Dialog entsteht die Möglichkeit, die geforderten Qualitäten in gewisser Weise „messbar“ oder zumindest operationalisierbar zu machen und so ein besseres Verständnis für die tatsächlich geforderten Qualitäten zu haben.[41] Die Beziehungen zwischen Stakeholdern sind typischerweise komplex und dynamisch, oft werden Entscheidungen indirekt durch Beeinflussung gefällt und die eigentliche Machtstruktur ist nicht offensichtlich.[42]
- Eine Priorisierung von Zielen inklusive widersprüchlichen Zielen kann vorgenommen werden.
- Die Architektur wird in der bestehenden Form dokumentiert. Diese Form der Explizitmachung reduziert daher auch den Anteil, der sich auf „tacit Knowledge“ (s. S. 12) verlässt und führt alleine durch den Zwang, es einem Dritten gegenüber

[40] Auch hier gibt es pathologische Ausnahmen. Von Zeit zu Zeit sind Stakeholder anzutreffen, denen das entstehende System völlig egal ist.

[41] Die Formulierung: *95% aller Transaktionen müssen innerhalb von 5 Sekunden abgeschlossen sein*, ist deutlich präziser als: *Das System soll so schnell wie möglich sein.* oder: *Das System soll in der Lage sein auch andere relationale ANSI-Datenbanken zu unterstützen*, ist deutlich präziser als: *Das System soll so flexibel wie möglich sein.*

[42] Die Sicht des Autors ist hier eine pessimistische: Die allermeisten Entscheidungen werden aufgrund von Machtstrukturen bzw. Machterhalt oder -gewinn gefällt, die dann zitierten „objektiven“ Kriterien wie Geld, Ressourcen, Budget oder Funktionalität sind sehr oft der Versuch einer nachträglichen Legitimation der einmal getroffenen Entscheidung.

zu formulieren, zu einer erhöhten Klarheit bei den Stakeholdern, indem diese explizit lernen.
- Die Organisation kann auf Dauer ein höheres Maß an Architekturreife gewinnen.

Jede Form des Architekturreviews muss eine Reihe von Ergebnissen liefern, sonst fehlt ihr die Sinnhaftigkeit. Besonders wichtig ist es, frühzeitig in der Lage zu sein Architekturen zu bewerten. Zu den zu leistenden Zielen einer Methode der Architekturbewertung gehören:

- die Fähigkeit sinnvolle Vorhersagen ohne Zugriff oder Existenz von Sourcecode zu treffen,
- die Fähigkeit sinnvolle Vorhersagen zu treffen ohne die COTS-Komponenten[43] erworben oder einen entsprechenden Geschäftsprozess implementiert zu haben,
- die Bestimmung des „Werts“ eines Designs,
- Vorhersagen auf hoher Abstraktionsebene über Aufwand, Kosten oder Maintenance treffen zu können,
- Design- und modellbasierte Bewertungen,
- Vorhersagen über die Nutzung und den Wert für den Endbenutzer,
- die Kosten und der Nutzen für die Endnutzer wie auch die Fähigkeiten des Systems,
- ein jederzeit einsetzbarer Architekturreview, obwohl die Einsparpotenziale zu Beginn der Entwicklung eines Systems am größten sind.

Ein Architekturreview ist ein flexibler Weg frühzeitig Probleme aufzudecken, trotzdem gibt es bei den Architekturreviews einige Probleme:

- Eine Architekturbewertung benötigt ein Team von Experten um in der Lage zu sein, unvorhergesehene Risiken neben antizipierten Risiken aufzudecken.
- Bei einem sehr abstrakten Design kann es zu Missverständnissen kommen, da Architekten i. d. R. nicht die Gründe für ein Design dokumentieren.
- Die Qualitätsanforderungen sind oft weder explizit formuliert, noch vollständig geklärt, wenn das Design der Architektur schon abgeschlossen ist.

Manche Projekte oder Systeme entstehen nicht aus offensichtlichen und nachvollziehbaren Gründen oder stiften einen Mehrwert für ein Unternehmen, sondern sie entstehen, weil die Beteiligten es unbedingt so wollen. Die Gründe reichen dann von Langeweile,[44] über den Versuch des Arbeitsplatzerhalts bis hin zu einem blanken Machtkampf im Unternehmen. Solche Projekte werden dann durch eine oder mehrere typische Taktiken zur Bewilligung freigegeben. Die hierfür verwendeten Taktiken sind meist:

- Das Projekt oder System wird als „strategisch“ bezeichnet. Durch den angeblich strategischen Status wird versucht den Nachweis über die Sinnhaftigkeit des Vorhabens auszuhebeln und die möglichen Gewinne als nicht vorhersagbar de-

[43] s. Abschn. 4.4

[44] Entwickler, die sich langweilen, erfinden neue Projekte. Nicht genug damit – da diese neuen Projekte meist sehr technisch orientiert sind, brauchen sie noch zusätzliche Unterstützung von externen Experten.