HANSER

Leseprobe

Stefan Zörner

Softwarearchitekturen dokumentieren und kommunizieren

Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten

Geleitwort von Gernot Starke

ISBN: 978-3-446-42924-6

Weitere Informationen oder Bestellungen unter

http://www.hanser.de/978-3-446-42924-6

sowie im Buchhandel.

3 Entscheidungen treffen und festhalten

„The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark."[1]

Philippe Kruchten

Im letzten Kapitel haben Sie Dokumentationsmittel kennengelernt, um die treibenden Einflüsse auf Ihre Softwarearchitektur festzuhalten. Architekturen zu entwerfen, heißt im Wesentlichen, Entscheidungen zu treffen. Den maßgeblichen Entscheidungen räumt dieses Kapitel einen besonderen Stellenwert ein. Ich zeige, wie Sie sie treffen und geeignet festhalten können, um das Ziel der Nachvollziehbarkeit zu adressieren.

3.1 Historisch gewachsen?

1815 – Europa im Umbruch. Napoleon verliert bei Waterloo seine letzte Schlacht. Mittlerweile haben Historiker reichlich Papier veröffentlicht, um die Gefechtsgeschehnisse zu dokumentieren und zu bewerten. Sie sind sich nicht in allen Punkten einig. Die Entscheidungen Napoleons sind zwar in Form seiner Befehle überliefert, nicht immer ist aber klar, welche Alternativen er zuvor gegeneinander abgewogen hat und warum er sich für eine bestimmte Option entschied. In einzelnen Punkten vermuten Geschichtsschreiber, Fehler in seinem Handeln entdeckt zu haben. Sie können aber nur spekulieren, ob Napoleon sie aus Unwissenheit beging oder ob eine Absicht dahinter stand, die sie nicht kennen.

200 Jahre später

Ein Softwaresystem in der Entwicklung. Architekturentscheidungen, also solche, die im weiteren Verlauf nur schwer zurückzunehmen sind, geben den Rahmen für die Umsetzung vor. Sie sind ein zentrales Arbeitsergebnis – Architektur erfolgreich umzusetzen, heißt vor allem, sie im Team zu kommunizieren. Und insbesondere sollten Sie Architekturentscheidungen geeignet festhalten. Warum eigentlich? Damit spätere Geschichtsschreiber Ihr Projekt leichter erforschen können? So lange brauchen Sie in der Regel nicht zu warten. Schon der erste neue Mitarbeiter im Team wird viele Fragen stellen, die auf zentrale Entscheidungen abzielen.

[1] Deutsch etwa: „Das Leben eines Softwarearchitekten ist eine lange (und manchmal schmerzvolle) Folge von suboptimalen Entscheidungen, die teilweise im Dunkeln getroffen werden."

„Warum habt ihr das denn mit X-Framework gemacht? Y-Framework ist doch viel besser!“ Nicht jeder im Team kann sich vielleicht noch erinnern, was damals für X-Framework sprach. Gerade in wachsenden Projekten, die häufig neue Mitarbeiter integrieren, kommt es zu den immer gleichen Debatten, die leicht vermieden oder zumindest abgekürzt werden können. Auch andere Projektbeteiligte haben Fragen. Und neben der Wissensvermittlung und -erhaltung im Team sind dokumentierte Entscheidungen auch für Architekturbewertungen und Reviews unerlässlich.

Wer zu spät kommt …

Wenn Sie und Ihr Team zentrale Entscheidungen hingegen zu spät dokumentieren, sind der Findungsprozess und die Intention dahinter bereits in Vergessenheit geraten. Falls zu einem späteren Zeitpunkt eine Entscheidung neu bewertet und ggf. geändert werden muss, fehlen wichtige Informationen. Werden wichtige Entscheidungen überhaupt nicht dokumentiert, können Sie später bei Fragen neuer Mitarbeiter oder im Architekturreview unter Umständen nur noch antworten mit – Sie erinnern sich –: „Das ist historisch gewachsen.“

3.2 Architekturentscheidungen

Das Wort „Entscheidung“ wird sowohl für die Fragestellung verwendet als auch für das Ergebnis. Einer Dokumentation können Sie typischerweise viele Entscheidungen (im Sinne von Ergebnis) entnehmen. In jedem Vorhaben gibt es einige wenige Fragestellungen, deren Beantwortung einen vergleichsweise großen Einfluss auf die Lösung hat, beziehungsweise wo falsche Entscheidungen das Scheitern bedeuten können (Waterloo-Klasse). Zu diesen zentralen Punkten wollen wir mehr festhalten als das bloße Ergebnis.

3.2.1 Architekturentscheidung (Dokumentationsmittel)

In der Praxis haben sich einfache Templates bewährt. Für jede zentrale Entscheidung fragen sie die gleichen Dinge ab und stellen so alle Architekturentscheidungen der Lösung gleichförmig dar.

Bild 3.1 zeigt einen Strukturierungsvorschlag[2], der zweierlei leistet: Erstens hilft er Ihnen, die Entscheidung zu bearbeiten und zu einem Ergebnis zu führen. Und zweitens gibt er eine Gliederung vor, wie Sie die Entscheidung geeignet festhalten. Beispielsweise mit Hilfe einer Vorlage – die Hauptäste ergeben dann die Kapitelüberschriften.

An den einzelnen Ästen der Mindmap sind Fragen angehängt, die Sie in Ihre Vorlage mit aufnehmen können.[3] Sie müssen diese Fragen nicht sklavisch beantworten. Vielmehr sollen sie Sie bei einer konkreten Architekturentscheidung, etwa in einem Workshop, leiten, anregen und unterstützen. Gehen wir die Äste der Reihe nach durch!

[2] Eine erste Fassung dieser Struktur ist in [Zörner2008] erschienen.

[3] Die Webseite zum Buch bietet entsprechende Vorlagen als Startpunkt für Sie zum Download an.

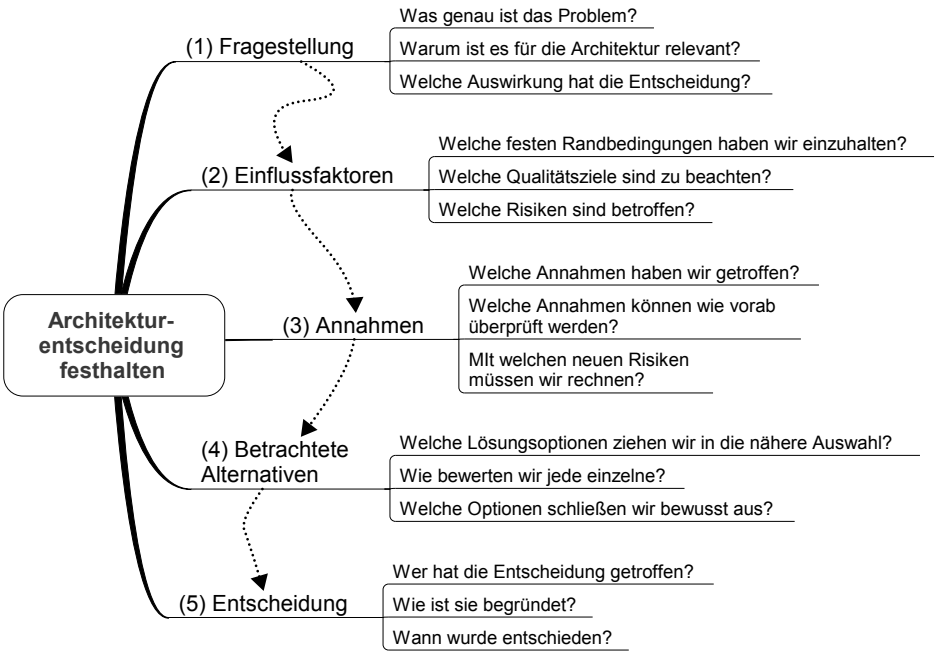

BILD 3.1 Struktur und Leitfragen zur Bearbeitung einer Architekturentscheidung

Zur Fragestellung

Entscheidungen im Projekt gibt es viele. Stellen Sie überzeugend dar, warum die hier betrachtete Fragestellung tatsächlich eine Architekturentscheidung ist. Die Auswahl der richtigen Fragestellungen für die Architekturbeschreibung ist ein wesentlicher Erfolgsfaktor für die Nachvollziehbarkeit.

Relevante Einflussfaktoren

Mit den Dokumentationsmitteln aus Kapitel 2 haben Sie mögliche Einflussfaktoren auf die Softwarearchitektur festgehalten. Von diesen sind bestimmte Randbedingungen, Qualitätsziele, Risiken etc. tonangebend für diese Fragestellung. Identifizieren und nennen Sie diese.

Annahmen

Die beschriebenen Einflussfaktoren grenzen die Fragestellung nicht völlig ein, sondern lassen Ihnen Entscheidungsspielraum. Sie treffen Annahmen, welche die möglichen Optionen weiter reduzieren oder sich auf die Bewertung der Optionen auswirken. Im Grunde wird jede Architekturentscheidung unter gewissen (oft impliziten) Annahmen getroffen.

> *„Entscheidungen sind immer dann nötig, wenn Wissen fehlt und trotzdem gehandelt werden muss.“ [Wohland+2007]*

In vielen Fällen machen getroffene Annahmen dem Außenstehenden erst begreiflich, warum eine Entscheidung so und nicht anders ausgefallen ist. Halten Sie sie fest (und machen Sie sie sich dadurch auch noch einmal bewusst). Annahmen zählen zu den Dingen, die im Nachhinein beinahe unmöglich zu rekonstruieren sind.

Betrachtete Alternativen

Je nach Fragestellung nehmen Sie zwei oder mehr Optionen in die engere Auswahl, wobei Sie die Anzahl aufgrund des Aufwandes (für die Entscheidung und auch für die Dokumentation) in der Regel klein halten. Machen Sie klar, wie Sie zur Auswahlliste gekommen sind. Gibt es auf den ersten Blick naheliegende Optionen, die Sie aufgrund von Einflussfaktoren oder Annahmen bewusst ausgeschlossen haben?

Bewerten Sie dann die Alternativen bezüglich der Fragestellung. Wie verhalten diese sich in Bezug auf die Einflussfaktoren?

Zur Entscheidung

Wie sieht das Ergebnis aus, und wie begründen Sie es? Um spätere Rückfragen schnell adressieren zu können, halten Sie denjenigen, der entschieden hat (oder diejenigen), fest oder zumindest die an der Entscheidung Beteiligten. Ebenso den Zeitpunkt, wann die Entscheidung getroffen wurde. Manche Optionen gab es zu diesem Zeitpunkt vielleicht noch gar nicht, oder die betreffenden Lösungen waren nicht ausgereift genug.

3.2.2 Fallbeispiel: Spannende Fragen „DokChess“

Im Rahmen der Realisierung der Schach-Engine DokChess stellte sich eine ganze Reihe von Fragen. Hier eine Auswahl zur Illustration:[4]

- Wie zerfällt das System in Teile? Wie hängen die verschiedenen Teile voneinander ab?
- Wie wird die Spielsituation („Stellung“) als Datenstruktur abgebildet?
- Sind Stellungsobjekte veränderlich oder nicht?
- Wie kommuniziert die Engine mit der Außenwelt?
- Welches Eröffnungsbibliotheksformat wird unterstützt? Wie?
- Wie stellen wir fest, ob die Engine gut genug spielt?

Im Folgenden finden Sie einige Hinweise, damit solche Fragen in einer Architekturdokumentation nicht wie hier im Beispiel einfach vom Himmel fallen, sondern sich in das Ganze einfügen.

3.2.3 Tipps zur Formulierung von Fragestellungen

Wie kommt man auf Fragestellungen für Entscheidungen?

Auch wenn sich viele Fragen dem Team oft von ganz allein stellen („Wie [machen | erreichen | lösen | verhindern | ...] wir eigentlich XY?“), möchte ich Ihnen Werkzeug an die Hand geben, um Kandidaten für Fragestellungen methodisch herzuleiten. In einem guten Architekturüberblick passt alles zueinander („Roter Faden“). Deswegen basieren die folgenden Ideen für Ihr Brainstorming auf den bisher besprochenen Dokumentationsmitteln.

[4] Zwei besonders interessante Fragen finden Sie in Kapitel 9 („Architekturüberblick DokChess“) nach dem vorgeschlagenen Schema bearbeitet.

- Wandern Sie den Systemkontext ab. Wie binden Sie die Akteure jeweils an?
- Gehen Sie die Qualitätsziele durch. Ist klar, mit welchen Strategien Sie diese jeweils erreichen? Welche Alternativen sehen Sie?
- Welche Qualitätsszenarien sind Ihrer Einschätzung nach schwer umzusetzen? Woran liegt das?
- Welche Highlevel-Entscheidung ließe sich im weiteren Projektverlauf nur schwer zurücknehmen (und ist keine Rahmenbedingung)?
- Wo lassen Ihnen die Rahmenbedingungen viel Spielraum?
- Welche Fragen werfen die identifizierten Risiken auf? Gibt es Optionen, um sie zu beherrschen?

Was sind typische Fragestellungen für Architekturentscheidungen?

Jedes Vorhaben ist anders und verfolgt seine eigenen Architekturziele. Auch vermeintlich „kleine" Fragestellungen können große Auswirkungen haben. Nichtsdestotrotz gibt es bestimmte Typen von Fragen („übliche Verdächtige"), die immer wieder als Architekturentscheidungen auftauchen:

- Welche Oberflächen erhalten die Benutzer?
- Wie binden wir Fremdsystem XY an?
- Wie kommunizieren Bestandteile unseres Systems untereinander?
- Wie adressieren wir querschnittliche Themen (zum Beispiel Persistenz, Verteilung ...)?
- Implementieren wir eine geforderte Funktionalität selbst, oder verwenden wir eine bestehende Lösung („Make or Buy", „Make or Take")?
- Welches Produkt/welche Technologie verwenden wir für XY?

Wie viele Ausgänge sollte eine Fragestellung haben?

Eine Fragestellung hat mindestens zwei Ausgänge, damit eine sinnvolle Entscheidung getroffen werden kann. Manchmal stoße ich in Dokumentationen auf Fragen, die „gefühlt" nur einen haben, zum Beispiel in der Form:

> Sollen wir OpenDuperPlus benutzen?

Auf diese Frage gibt es zwei Antworten (ja und nein), aber eine impliziert sofort die nächste Frage: Was machen wir, wenn wir OpenDuperPlus nicht verwenden können oder wollen? Bei manchem Leser sicher auch: Was ist OpenDuperPlus[5]?

In solch einem Fall gilt es eine ergebnisoffene Fragestellung herauszuarbeiten. Was bezwecken Sie mit der Verwendung von OpenDuperPlus? Welches Problem löst es? Das konkrete Produkt sollte dann eine Alternative sein.

Es ist kein Zufall, dass Fragen wie die obige insbesondere in solchen Architekturbeschreibungen zu finden sind, wo im Nachhinein dokumentiert wurde. Der Ausgang ist dann natürlich, dass OpenDuperPlus benutzt wird. Das betreffende Unterkapitel der Architekturbeschreibung heißt nur „OpenDuperPlus" oder „Warum OpenDuperPlus?". Es stellt die Vorzüge der gewählten „Alternative" dar. Genau solche Effekte soll die hier vorgeschlagene Struktur verhindern.

[5] Den Namen habe ich mir ausgedacht, um den Effekt auch bei Ihnen zu erzielen. Siehe auch „Wie betitelt man eine Architekturentscheidung geschickt?"

Auch die Vergabe eines guten Titels kann wirkungsvoll unterstützen. Nennen Sie die Unterkapitel nicht „Entscheidung 1“, ... „Entscheidung N“ (schon gesehen).

Wie betitelt man eine Architekturentscheidung geschickt?

In eine Architekturbeschreibung werden die zentralen Entscheidungen typischerweise zu einzelnen (Unter-)Kapiteln, in einem Wiki zu einzelnen Seiten. In beiden Fällen spielt der Titel (also die Kapitel- bzw. Seitenüberschrift) eine große Rolle. Er landet im Inhaltsverzeichnis und je nach Formatierung in Kopf- oder Fußzeile. Im Falle des Wikis tritt er prominent in Suchergebnissen auf.

Während der Titel bei vielen anderen vorgestellten Werkzeugen klar ist (man nennt das Kapitel einfach wie das Dokumentationsmittel), gibt es bei Architekturentscheidungen zwei typische Optionen:

- Thema oder Fragestellung der Entscheidung (z. B. „Persistenz“, „Wie persistieren wir ...?“)
- Ergebnis der Entscheidung (z. B. „O/R-Mapping mit Hibernate zur Persistenz“)

Die erste Option hat den Vorteil, dass Sie ihn bereits vergeben können, bevor das Ergebnis feststeht, also schon während der Erarbeitung. Das entspricht dem wünschenswerten Vorgehen, Entscheidungen bereits festzuhalten, während man sie fällt. Wird die Entscheidung getroffen oder geändert, kann der Titel stabil gehalten werden. Ich persönlich präferiere diese Option und empfinde dabei eine richtige Fragestellung (nicht nur das Thema) als sehr lebendig.

Die zweite Variante punktet dadurch, dass bereits an der Überschrift, also beim Überfliegen eines Inhaltsverzeichnisses, das Ergebnis abzulesen ist. Auch in dieser Option sollte nicht nur die favorisierte Alternative selbst als Überschrift gewählt werden („OpenDuperPlus“). Lassen Sie die Problemstellung mit anklingen. Es ist auch ein Kompromiss denkbar, bei dem nach Treffen der Entscheidung das Ergebnis mit im Titel landet. Beim Revidieren einer Entscheidung muss bei dieser Option der Titel angepasst werden.

3.2.4 Fallbeispiel: Fragestellungen „immer-nur-schach.de“

Im Folgenden stelle ich wichtige offene Entscheidungen für immer-nur-schach.de vor und motiviere jeweils die Fragestellung. Für eine dokumentierte Architekturentscheidung ist der kurze Text jeweils ein Beispiel für das einleitende Unterkapitel „Fragestellung“ (vgl. Bild 3.1, Ast 1). Die Entscheidungen sind hier aus Platzgründen nicht vollständig dargestellt. Ausgearbeitete Beispiele für DokChess finden Sie in Kapitel 9.

1. Auf welcher Ablaufumgebung basiert der Server?

Bei immer-nur-schach.de spielen die Gegner auf unterschiedlichen Clients gegeneinander. Die Kommunikation erfolgt über einen in Java zu realisierenden Server. Für diesen gibt es Anforderungen, unter anderem bezüglich Zuverlässigkeit, Portierbarkeit und Sicherheit. Das legt die Verwendung eines Java-basierten Applikationsservers nahe. Hier ist eine konkrete Lösung auszuwählen. Offen ist dabei, ob ein kommerzielles Produkt oder eine Open-Source-Implementierung verwendet wird. Wird gezielt für einen konkreten Server realisiert, oder soll durch Beschränkung auf Standards (z. B. Java EE 5, Java EE 6 Web Profile) mehr Flexibilität erreicht werden?

2. Wo wird der Zustand einer laufenden Online-Partie gehalten?

Während eine Partie läuft, tauschen beide Gegner ihre Züge über den Server aus. Der Zustand einer Partie umfasst unter anderem die Situation auf dem Brett und den bisherigen Verlauf. Am Ende einer Partie muss diese persistiert werden, um später darauf zurückgreifen zu können. Wo liegt der Zustand vor Spielende? Zu den Optionen zählen die Clients der Spieler, der Server und eine Datenbank. Auch eine Kombination ist denkbar. Die Entscheidung hat Einfluss auf die Benutzbarkeit und Zuverlässigkeit von immer-nur-schach.de.

3. Welches Programmiermodell/Framework wird für den Server verwendet?

In Abhängigkeit von der Ablaufumgebung (siehe oben) stehen verschiedene querschnittliche Funktionalitäten zur Verfügung (Komponentenmodell, Konfiguration, Transaktionen, Persistenz, Sicherheit) oder auch nicht. Soll auf ein Applikationsframework zurückgegriffen werden, um fehlende Aspekte und zukünftige Erweiterungen (Stichwort Wartbarkeit) leichter umsetzen zu können oder reicht das Programmiermodell des Servers aus? Zu den Alternativen zählen Java EE, das Spring Framework und OSGi, wobei sich die genannten nicht wechselseitig ausschließen.

4. Wo werden Daten persistiert?

Im Rahmen von immer-nur-schach.de sind unterschiedliche Dinge zu speichern, zum Beispiel Mitgliederdaten und gespielte Partien. Sie wachsen mit der Zeit an; Mitglieder und Schachpolizisten müssen sie effizient abfragen können. Als Speichertechnologien kommen grundsätzlich (mindestens) Dateien, relationale Datenbanken und NoSQL-Lösungen in Betracht. Dabei muss nicht zwingend eine Option das Speichermedium für alles sein. Die Entscheidung hat Einfluss auf die Zuverlässigkeit, Wartbarkeit, Effizienz und Portierbarkeit.

5. Wie werden Daten persistiert?

In Abhängigkeit vorheriger Entscheidungen ist unter Umständen noch offen, wie auf Speichermedien zugegriffen wird. Kommt eine Bibliothek oder ein Persistenzframework zum Einsatz, oder reichen die Bordmittel des Programmiermodells bzw. des Frameworks bereits aus? Unter Umständen können auch hier nicht mit einer Lösung alle Anforderungen zu Wartbarkeit und Effizienz befriedigend erfüllt werden.

6. Wie wird die Web-Applikation realisiert?

Viele Mitglieder greifen über einen Browser auf die Plattform zu. Zu kaum einem Thema gibt es in Java mehr Optionen als zu Webframeworks; auch die Auswahl der Oberflächentechnologien (HTML, JavaScript, AJAX, ...) steht aus. Benutzbarkeit spielt für immer-nur-schach.de eine große Rolle, die Entscheidung hat auch Einfluss auf die Wartbarkeit der Lösung. Ähnlich wie bei der Persistenz ist bei dieser Frage das Programmiermodell interessant. Welche Möglichkeiten bietet es selbst? Welche Zusatzbibliotheken werden unterstützt? ...

7. Wie interagieren Clients und Server bei Partien miteinander?

Eine besondere Herausforderung stellt die Interaktion zwischen den Gegnern einer Partie miteinander dar. Sie läuft über den Server. Wie bekommt zum Beispiel ein Web-Client mit, dass der Gegner gezogen hat? Fragt er regelmäßig (z. B. Polling alle 2 s) nach, oder wird eine völlig andere Lösung (z. B. anderes Protokoll, WebSocket) gewählt. Die Entscheidung wirkt auf Benutzbarkeit und Effizienz und hat Wechselwirkungen mit Ablaufumgebung und Randbedingungen bezüglich der Clients. Neben Browsern müssen auch die geforderten und ggf. zukünftigen mobilen Clients bedacht werden. Mit welcher Technologie soll eine entsprechende Remote-API zur Verfügung gestellt werden?

8. Wird für alle Benutzergruppen die gleiche Clienttechnologie verwendet?

Neben den Mitgliedern müssen auch andere (konkret Schachpolizisten, Administratoren) auf die Plattform zugreifen. Wird für die Oberflächen aller Anwendergruppen die gleiche Clienttechnologie verwendet, oder werden für bestimmte z. B. auch Rich Clients oder Kommandozeilenwerkzeuge realisiert?

9. Wie wird die Forenfunktionalität abgebildet?

Im Rahmen von immer-nur-schach.de ist ein Forum für die Mitglieder gefordert. Hier ist zu entscheiden, ob man diese Funktionalität selbst implementiert oder eine Lösung integriert („Make or buy?“). Im Falle eines Fremdproduktes ist dieses auszuwählen. Die Entscheidung wirkt sich vor allem auf die Benutzbarkeit, Wartbarkeit und Portierbarkeit aus.

10. Wie wird die Internet-Werbung realisiert?

Die Internet-Werbung („AdServer“) soll nicht selbst implementiert werden. Stattdessen ist entweder ein Fremdprodukt auszuwählen, das immer-nur-schach.de hostet, oder eine Plattform auszuwählen, welche die Funktionalität anbietet. In beiden Fällen muss die Werbung geeignet in die Oberflächen von immer-nur-schach.de eingebettet werden. Die Entscheidung hat Einfluss auf Wartbarkeit und Portierbarkeit.

Beeinflussungen

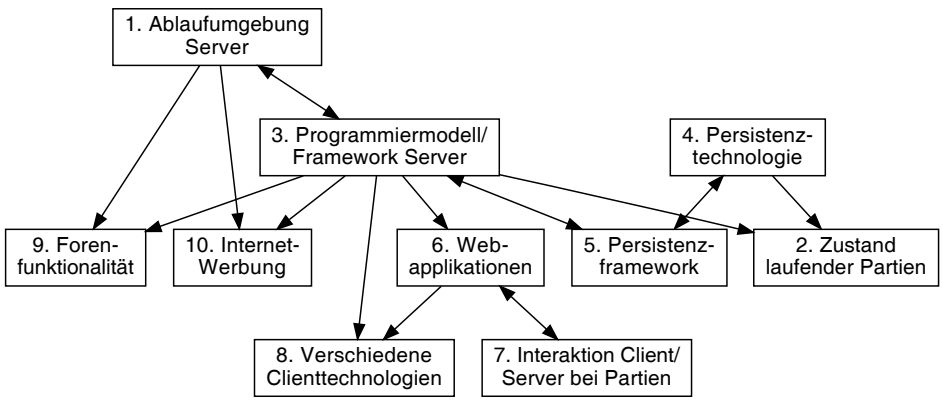

BILD 3.2 Thematische Beeinflussungen der Fragestellungen

Die Themen der Fragestellungen sind teilweise voneinander abhängig, sie beeinflussen sich. Bild 3.2 zeigt die Zusammenhänge. Sie können die Beziehungen bereits während des Entwurfes in dieser Form visualisieren, es hilft Ihnen bei der Arbeit. Wertvoll ist so eine Darstellung aber auch, um sich im Rahmen der Dokumentation einen Überblick zu verschaffen (siehe auch nächstes Unterkapitel). Die Lösung der Fragestellungen kann sich wechselseitig beeinflussen, wie in Bild 3.2 bei Frage 1 und 3. Wenn die Entscheidungen getroffen werden, bleibt typischerweise nur eine Richtung übrig, die oft die chronologische ist (Beispiel: wir haben uns für Java EE als Programmiermodell entschieden, und das beeinflusste die Entscheidung für den Applikationsserver JBoss als Ablaufumgebung). Die Dokumentation sollte dies berücksichtigen, und bei Verwendung einer Darstellung wie Bild 3.2 innerhalb eines Architekturüberblickes sollte bereits aus der Beschriftung klar hervorgehen, welchen Sachverhalt (z. B. Zeitpunkt) sie zeigt.

Steckbrief: Dokumentationsmittel „Architekturentscheidung“

Synonym: Entwurfsentscheidung

Zweck:

Nachvollziehbare Darstellung der zentralen Entscheidungen, im Architekturüberblick an prominenter Stelle versammelt zum schnellen Zugriff.

Form:

Textuell, je Entscheidung ein kurzes Dokument/Unterkapitel mit stets gleicher Struktur, zum Beispiel wie hier vorgeschlagen:

- Fragestellung
- Relevante Einflussfaktoren
- Getroffene Annahmen
- Betrachtete Alternativen
- Begründete Entscheidung

Um Zusammenhänge überblicksartig darzustellen, zum Beispiel ergänzt um Kreuztabellen und Beeinflussungsdiagramme.

Checkliste für den Inhalt (pro Architekturentscheidung):

- Wird plausibel dargestellt, warum die Fragestellung architekturrelevant ist?
- Ist die Fragestellung offen formuliert? Lässt sie Alternativen zu?
- Sind die Einflussfaktoren (vor allem die Qualitätsziele!) konsistent zur restlichen Architekturdokumentation?
- Ist die Auswahl der Alternativen nachvollziehbar?
- Werden die Alternativen ergebnisoffen gegeneinander abgewogen?
- Ist die Begründung für die Entscheidung schlüssig aus den Einflussfaktoren hergeleitet?
- Ist festgehalten, wer an der Entscheidung beteiligt war und wann sie getroffen wurde?

Checkliste für die ausgewählten Entscheidungen in einem Architekturüberblick:

- Begünstigt die Reihenfolge der dargestellten Entscheidungen ein sequenzielles Lesen?
- Haben die Titel (Überschriften) der Entscheidungen die gleiche Form?
- Ist die Anzahl der ausgewählten Entscheidungen angemessen (Praxistipp für den Überblick: 5 +/– 2)?

Ablageort arc42:

Abschnitt 9: „Entwurfsentscheidungen“, Abschnitt 4: „Lösungsstrategie“

Übungsaufgabe 3: Fragestellungen formulieren, Entscheidung festhalten

Sammeln Sie stichpunktartig Fragestellungen, deren Beantwortung Einfluss auf die Architektur des Squeezebox Servers hatte. Wählen Sie davon eine aus, die Sie besonders interessant finden und deren Antwort Sie rekonstruieren können. Bearbeiten Sie diese Fragestellung nach dem Schema der Mindmap (Bild 3.1). Bei den Lösungsalternativen können Sie sich von folgenden Fragen leiten lassen:

- Wie hätte man das anders lösen können?
- Wie hätten Sie das gemacht?

Diskutieren Sie Stärken und Schwächen der Alternativen, zum Beispiel indem Sie sie gegen Qualitätsszenarien halten. Treffen Sie Annahmen, wo Ihnen Wissen fehlt, und dokumentieren Sie diese. Formulieren Sie Fragen, die Sie den Entwicklern stellen würden, um mehr Klarheit zu bekommen.

Insgesamt (Liste der Fragestellungen, ausgearbeitete Entscheidung) sollte Ihre Lösung 3 DIN-A4-Seiten nicht übersteigen.

3.3 Einflussfaktoren auf Entscheidungen

Kapitel 2 beschreibt Einflussfaktoren, die auf Entscheidungen wirken. Wäre das Ergebnis in der Dokumentation eine Reihe kurzer Kapitel je Entscheidung (oder einzelne Seiten im Wiki), ginge der Überblick schnell verloren. Wenn Sie effizient mit den Ergebnissen weiterarbeiten wollen, sollten Sie diesen aber behalten.

3.3.1 Den Überblick behalten

In der Praxis stellen sich im Zusammenhang mit der Nachvollziehbarkeit der Softwarearchitektur und auch ihrer Weiterentwicklung oft Fragen wie diese:

- Welche Entscheidungen beeinflussen die Erreichung eines bestimmten Qualitätsziels?
- Randbedingung XY hat sich geändert, welche Entscheidungen sollten wir daraufhin noch einmal überprüfen?
- Zu Qualitätsmerkmal YZ ist ein neues Szenario erarbeitet worden. Welche Lösungsalternativen halten wir dagegen?

Tatsächlich stehen die betreffenden Dinge in Beziehung zueinander, wie exemplarisch in Bild 3.3 gezeigt. In der Architekturdokumentation sind die Zusammenhänge zwar idealerweise in Prosa oder stichpunkthaft beschrieben (siehe Vorlage zu Entscheidungen). Die Informationen sind aber gut über das ganze Dokument verstreut. Deshalb kann man leicht den Überblick verlieren. Es besteht die Gefahr, Zusammenhänge zu übersehen. Daher bietet sich an, die betreffenden Beziehungen zu verdichten und näher zueinander zu bringen. Eine gute Möglichkeit, Verknüpfungen überblicksartig darzustellen, sind Kreuztabellen.

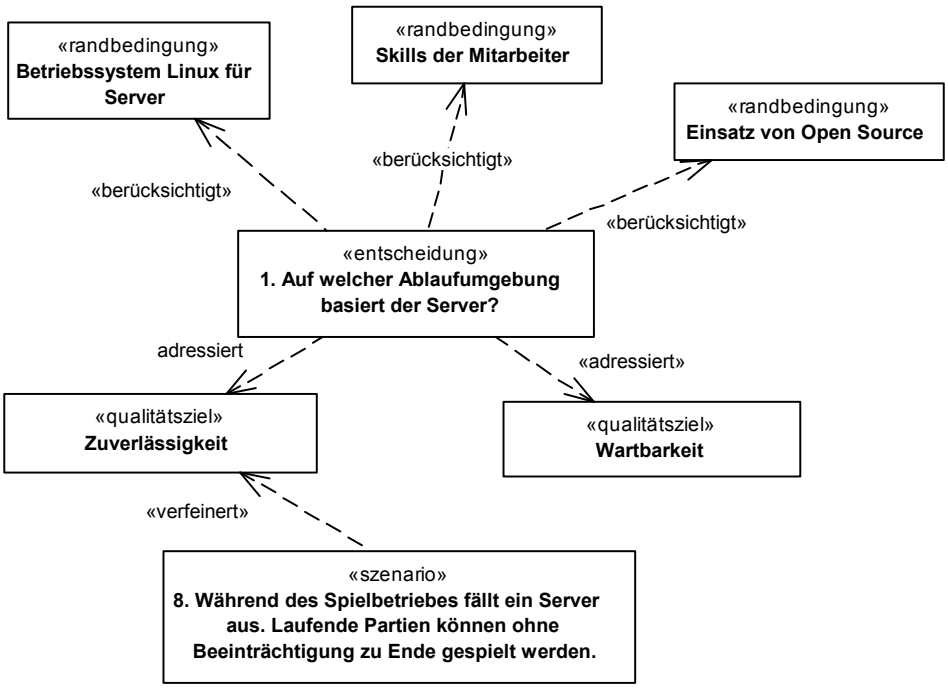

BILD 3.3 Ausschnittartige Zusammenhänge rund um eine Fragestellung

3.3.2 Kreuztabellen

Eine „normale“ Tabelle (wie z. B. in einer relationalen Datenbank) enthält Spaltenüberschriften und darunter Datensätze als Zeilen, deren Zellen jeweils einen Wert für das Merkmal der entsprechende Spalte angeben. Im Gegensatz dazu enthält eine Kreuztabelle sowohl Spalten- als auch Zeilenüberschriften, diese enthalten jeweils die Ausprägungen eines Merkmals (z. B. die verschiedenen Qualitätsziele). Die Tabellenzellen werden als Schnittpunkte (Kreuzpunkte) der beiden Merkmale gelesen.

Als Merkmale kommen in unserem Fall Fragestellungen, Entscheidungen, Anforderungen und Einflüsse aller Art in Frage, konkret etwa:

- Architekturentscheidungen
- User Stories
- Qualitätsmerkmale, Qualitätsziele, Qualitätsszenarien
- Randbedingungen
- Technische Risiken

Für eine konkrete Ausprägung einer Kreuztabelle entscheiden Sie sich für zwei Merkmale und stellen diese als Zeilen und Spalten einer Tabelle dar. Im einfachsten Fall verknüpft ein X in der bestimmten Zelle die betreffenden Ausprägungen der Merkmale miteinander. Korrekt gepflegt, verhilft die Tabelle sehr schnell zu Antworten auf Fragen wie diese:

- Welche Entscheidungen wurden durch die Skills unserer Mitarbeiter beeinflusst?
- Wie beeinflussen sich Architekturentscheidungen untereinander?
- Bei welchen Qualitätsmerkmalen mussten wir Kompromisse eingehen?
- Zu welchen Qualitätszielen haben wir (noch) keine Qualitätsszenarien?

3.3.3 Fallbeispiel: Einflüsse „immer-nur-schach.de“

Zur Illustration zeigt Tabelle 3.1, welche Qualitätsziele von „immer-nur-schach.de“ (vgl. Kapitel 2.5.3) durch die verschiedenen Fragestellungen (vgl. Kapitel 3.2.4) besonders adressiert werden.

TABELLE 3.1 Welche Fragestellungen adressieren vorrangig welche Qualitätsziele?

	Benutzbarkeit	Zuverlässigkeit	Wartbarkeit
1. Auf welcher Ablaufumgebung basiert der Server?		X	X
2. Wo wird der Zustand einer laufenden Online-Partie gehalten?	X	X	
3. Welches Programmiermodell/Framework wird für den Server verwendet?			X
4. Wo werden Daten persistiert?		X	
5. Wie werden Daten persistiert?		X	X
6. Wie wird die Web-Applikationen realisiert?	X		X
7. Wie interagieren Clients und Server bei Partien miteinander?	X	X	
8. Wird für alle Benutzergruppen die gleiche Clienttechnologie verwendet?	X		X
9. Wie wird die Forenfunktionalität abgebildet?	X		
10. Wie wird die Internet-Werbung realisiert?			X

3.3.4 Tipps zur Anfertigung von Kreuztabellen

Einfache Kreuztabellen wie Tabelle 3.1 stellen lediglich eine Beziehung zwischen zwei Merkmalen dar. Für verschiedene Fragestellungen sind dann auch verschiedene Tabellen zu erstellen und zu pflegen. Alternativ dazu können Kreuztabellen mehrere Merkmale als Zeilen oder Spalten aufnehmen, z. B. Qualitätsziele und Randbedingungen in zwei Blöcken nebeneinander oder untereinander. Auch mehrdimensionale Darstellungen sind durch geeignete Schachtelung der Zeilen und Spalten möglich.

Kreuztabellen mit mehr als zwei Merkmalen werden schnell sehr groß und somit für überblicksartige Dokumentation unhandlich. Generell sind sie zu allererst ein Arbeitsmittel, in den Architekturüberblick sollten Sie nur die aufnehmen, die ein besonders spannendes Zusammenwirken beschreiben, ggf. in vereinfachter Form. In Tabelle 3.1 habe ich entsprechend die geforderten Qualitätsmerkmale und nicht die Szenarien gewählt.

Weitere Informationen in einer Tabelle

Anstelle eines Kreuzes können Sie in die Zelle auch Werte aufnehmen, etwa eine Priorisierung oder Gewichtung. In der Statistik werden solche Tabellen benutzt, um Informationen zusammenzufassen und zu verdichten.

Je nachdem, mit welchen Werkzeugen Sie die Kreuztabellen erstellen und pflegen, können solche Werte die Arbeit mit ihnen verbessern, z. B. durch die Möglichkeit, zu filtern oder zu sortieren. Auch das Aufnehmen eines detaillierteren oder weiteren Einflusses in die Tabellenzellen ist denkbar. Anstatt der Kreuze in Tabelle 3.1 können Sie auch besonders relevante Qualitätsszenarien zu dem Thema in die Zellen aufnehmen.

Werkzeugfrage

Im Grunde ein Vorgriff auf das Werkzeugkapitel, aber die Frage liegt natürlich auf der Hand: Wie erstellen Sie Kreuztabellen effizient, insbesondere wenn Sie mehrere Zusammenhänge darstellen oder die Zusammenhänge unterschiedlich detailliert oder gefiltert haben wollen?

Die naheliegende Werkzeugwahl für Kreuztabellen sind Tabellenkalkulationen wie z. B. Excel. Wenn die gleichen Informationstypen unterschiedlich miteinander verknüpft werden sollen, ist es attraktiv, die Elemente und Beziehungen untereinander als Graph in einem Modell zu speichern und die Kreuztabellen daraus zu generieren. Bild 3.4 zeigt als Beispiel das UML-Werkzeug Enterprise Architect[6], das eine solche Funktionalität bietet. Modelliert wurden die Beziehungen in Diagrammen wie z. B. Bild 3.3 auf der nächsten Seite.

[6] Sparx Systems, http://www.sparxsystems.com

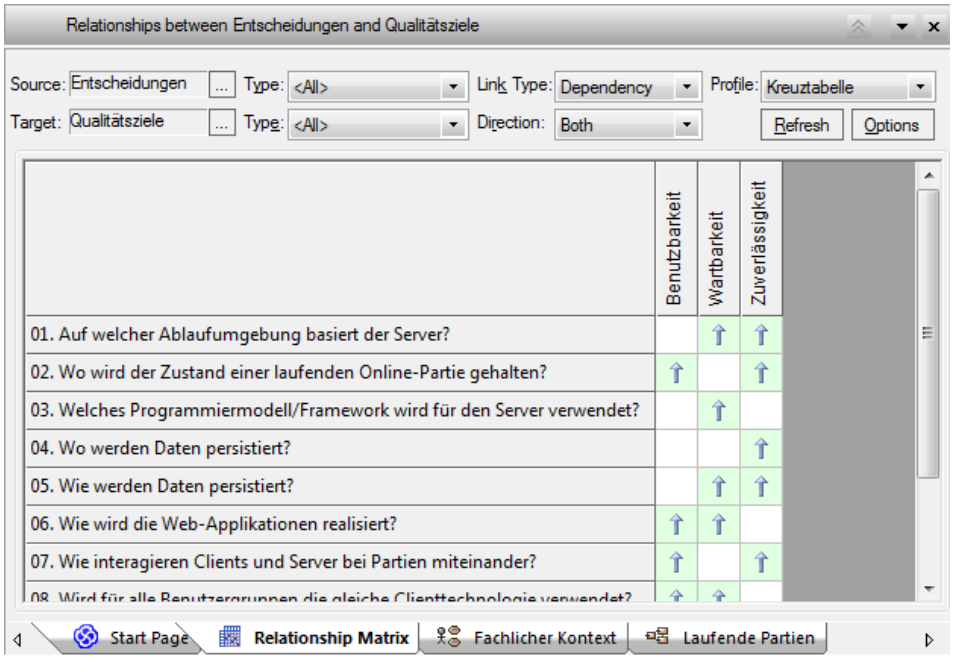

BILD 3.4 Beziehungen zwischen Modellelementen in einem UML-Werkzeug

Schematische Darstellungen aus Kreuztabellen

Auch der umgekehrte Weg, also das Ableiten einer graphischen Darstellung aus einer Kreuztabelle ist denkbar. Ein Beispiel haben Sie in Bild 3.2 gesehen, ich habe es mit Graphviz[7] erstellt.

Kernaussage dieses Kapitels

Beim Dokumentieren einer Architekturentscheidung ist der Lösungsweg mindestens so interessant wie die getroffene Entscheidung selbst. Ihre Annahmen gehören genauso dazu wie verworfene Alternativen. Sie lassen sich aus der realisierten Lösung später schwer ablesen, ebenso Ihre Begründung. Halten Sie zentrale Entscheidungen bereits im Entstehen fest, und wählen Sie eine Form, in der Sie und Ihre Leser den Überblick behalten!

[7] http://www.graphviz.org, siehe auch Kapitel 7