QUALITÄT

Mehr als ein Werbeslogan

Qualität als primärer Treiber systematischer Architekturarbeit

Dieser Artikel erläutert, wie Qualität von der Anforderungsanalyse über den Entwurf bis hin zu Implementierung und Test methodisch herbeigeführt werden kann. Der Architekturdisziplin fällt hierbei eine Schlüsselrolle zu. Zwar sind nicht alle Disziplinen in der Verantwortung der Architekt:innen, dennoch sollte Architektur und damit das Thema Qualität in allen Entwicklungsphasen mit bedacht werden.

Anspruch und Wirklichkeit

Qualität ist ein Dauerbrenner. In der Welt der Software genauso wie überall sonst. Wer kennt nicht die unzähligen Werbeslogans wie „Bei Qualität machen wir keine Abstriche"? Der an sich selbst gestellte – zumindest der für die Außenwirkung formulierte – Anspruch ist hoch. Leider müssen wir aber immer wieder feststellen, dass Anspruch und Wirklichkeit meilenweit auseinanderliegen. Mit Floskeln und Marketing allein ist es nun einmal nicht getan.

Zwar ist der diffuse Wunsch nach Qualität mit Sicherheit vorhanden, dennoch hört es damit dann oftmals auch schon auf. Der Wille, sich mit dem Thema ernsthaft auseinanderzusetzen, ist in der Realität leider häufig begrenzt. Folglich ist das Wissen darüber, wie Qualität methodisch herbeigeführt werden kann, selten gegeben. „Qualität herbeiführen" wird von vielen fälschlicherweise mit Qualitätssicherung gleichgesetzt und damit ist das Thema erledigt. Es ist aber nicht möglich, Qualität nachträglich ins System hineinzutesten. Wenn das Kind schon in den Brunnen gefallen ist, kann auch die Q-Abteilung aus einem falschen System nicht das richtige zaubern. Stattdessen müssen wir Qualität von Anfang an ganzheitlich betrachten – über die verschiedenen Phasen des Entwicklungszyklus hinweg und darüber hinaus, nachdem die Software in den Verkehr gebracht wurde.

Qualität: Was ist das eigentlich?

Die methodische Auseinandersetzung mit dem Thema Qualität beginnt schon mit dem Qualitätsbegriff an sich. Was Softwarequalität nun wirklich genau heißt, ist oftmals nicht klar bekannt. Vielmehr scheint zu gelten: Viele Menschen, viele Meinungen. Ein Produktmanager hat häufig eine ganz andere Vorstellung von Qualität als eine Usability-Expertin oder die Entwicklungsabteilung.

Wer hat nun recht? Die Entwickler, die sich gut wartbaren Code wünschen? Oder geht es doch eher um Usability, vielleicht sogar primär um eine hohe Performance? Bild 1 soll die Heterogenität des Qualitätsbegriffs verdeutlichen.

Diese unterschiedlichen Aspekte sind tatsächlich mögliche Qualitäten, wie sie beispielsweise die ISO 25010 [1] beschreibt – mehr dazu weiter unten.

Doch natürlich kann Qualität nicht einfach eine Frage des Standpunkts sein: Statt subjektivem Bauchgefühl muss eine objektive methodische Herangehensweise die Frage beantworten, welche Qualitätsattribute für ein spezielles Softwaresystem wirklich relevant sind und wie wir diese herbeiführen können. Nur dann stellen wir sicher, dass der Satz „unser System hat eine hohe Qualität" keine Floskel bleibt, sondern zu einer belegbaren Aussage wird.

Hartnäckige Mythen

Auch wenn es mittlerweile bekannt sein sollte: Eine Eier legende Wollmilchsau bauen zu wollen ist nach wie vor ein unsinniges Unterfangen. Wir können kein System bauen, welches alle Qualitäten erfüllt, sondern müssen uns auf die wesentlichen Attribute konzentrieren.

Und diese sind bei jedem nichttrivialen System unterschiedlich: Es gibt keine Silver Bullet. Ein gut entworfenes System ist maßgeschneidert. Das heißt nicht, dass für schon gelöste Probleme das Rad neu erfunden werden soll, ganz im Gegenteil. Aber jedes Produkt ist individuell: Es verfolgt spezifische Geschäftsziele und muss sich gegenüber Konkurrenzprodukten etablieren. Eine 0815-Lösung, die sich für alles eignen soll, wird nur auf die wenigsten Systeme perfekt passen.

Es gibt also keine Abkürzung, um eine Antwort auf die Frage zu finden, welche Qualitätsattribute für ein spezielles Softwaresystem wirklich relevant sind. Es ist die Aufgabe der Architekten und Architektinnen, hierauf durch methodisches Vorgehen eine objektive Antwort zu finden.

Was ist Qualität? Viele Menschen, viele Meinungen (Bild 1)

Struktur hineinbringen durch Qualitätsmodelle

Nachdem wir schon Qualitätsattribute erwähnt haben, stellt sich die Frage, welche Attribute es überhaupt gibt und welche Attribute wir betrachten sollten. Hierbei helfen uns Qualitätsmodelle.

Ein Qualitätsmodell ist eine strukturierte Sammlung von den – in unserem Fall für ein Softwaresystem – relevanten Qualitätskriterien. Ich verwende die Begriffe Attribut und Kriterium synonym, auch wenn einige Normen dies anders tun. Nach Meinung des Autors stiftet Letzteres aber mehr Verwirrung, als es hilft. Das International Software Architecture Qualification Board (iSAQB) [2] beispielsweise verwendet die Begriffe ebenfalls synonym. Die Qualitätskriterien sollten dabei klar und eindeutig definiert sein. Sie lassen sich mittels einer Baumstruktur abbilden, die Subkriterien unter verschiedene Kernkategorien einsortiert. Beispielweise könnten wir innerhalb der Kernkategorie Performance sowohl das Zeitverhalten als auch den Ressourcenverbrauch betrachten. Das Zeitverhalten wiederum könnten wir in feingranularere Attribute wie Latenz und Durchsatz aufdröseln und so weiter.

Product Quality Model der ISO 25010 (Bild 2)

Welches Qualitätsmodell?

Was ist nun ein geeignetes Modell, und unter welchen Modellen können wir auswählen? Die bekanntesten Qualitätsmodelle für die Softwareentwicklung sind wohl durch die ISO 25010 (als Nachfolgenorm der ISO 9126 [3]) gegeben. Das heißt nicht, dass diese Modelle für jeden Anwendungsfall (oder ganz allgemein) die optimale Wahl darstellen. Zum einen ist die Menge der aufgeführten Kriterien nicht vollständig – das will die Norm auch gar nicht erreichen, weswegen sie dies auch explizit erwähnt. Für bestimmte Systeme fehlen eventuell wichtige Attribute oder sie sind nicht ganz passgenau: Beispielsweise sucht ein Microservice-Entwickler unter Umständen verzweifelt nach Skalierbarkeit. Und Sustainability im Sinne des CO_2-Fußabdrucks und Green Software lässt sich leider auch nicht finden.

Die ISO 25010 wie auch jedes andere generische Qualitätsmodell (beispielsweise ein von einer Firma vorgegebenes Modell) stellt daher erst einmal eine sinnvolle Basis, einen guten Startpunkt dar und kann, wenn nötig und möglich, an eigene Anforderungen angepasst werden. Auch das sieht die Norm selbst so vor – Stichwort Tailoring.

ISO 25010

Werfen wir einen Blick auf das Product Quality Model der ISO 25010, um das abstrakte Konzept Qualitätsmodell etwas greifbarer werden zu lassen (Bild 2).

Zur Erläuterung: Beim Product Quality Model geht es um die interne (verborgene) und externe (von außen messbare) Qualität, allerdings ohne den Nutzungskontext mitzubetrachten. Letzterer ist Fokus im Quality-in-Use-Model. Ein Durchkauen der einzelnen Qualitätskriterien werden wir uns sparen – das ist Aufgabe der Norm.

Ein Hinweis aber noch: Die einzelnen Qualitätskriterien sollte man nicht nach eigenem Gutdünken interpretieren, sondern in der Norm nachschlagen. Auch wenn einige Definitionstexte im Hinblick auf ihre Verständlichkeit vielleicht noch Raum für Verbesserungen aufweisen: Das Nachschlagen hilft, Fehlinterpretationen und damit Missverständnisse zu vermeiden.

Allen, die als Softwareentwickler:in und -architekt:in tätig sind, möchte der Autor deshalb an dieser Stelle als Empfehlung mitgeben, zumindest von der ISO 25010 ein Exemplar auf dem Rechner oder im Regal stehen zu haben. Der Autor ist mitnichten ein Normenfetischist, aber gerade deswegen: Die Norm ist kurz und knapp und gehört nach Meinung des Autors in den Handwerkskasten all derer, die die Softwerkskunst ernst nehmen wollen.

Jetzt haben wir also ein Qualitätsmodell – und nun?

Wenn wir uns methodisch mit Qualität auseinandersetzen wollen, dann ist die Auswahl eines geeigneten Qualitätsmodells der erste sinnvolle Schritt. Das Qualitätsmodell dient uns zum einen als eine Art Checkliste („Haben wir an alles Nötige gedacht?“). Vor allem aber ist es die Basis für sämtliche weiteren Aktivitäten im Vorgehen, bei denen wir Qualität genauer unter die Lupe nehmen wollen, denn dort müssen wir nun nicht mehr nur diffus über Qualität reden.

Wir haben stattdessen klar definierte Qualitätskriterien an der Hand, die uns dabei helfen, präzise Qualitätsanforderungen zu erheben und für diese passende Entwurfsstrategien festzulegen.

Qualitätsanforderungen gehören in den Fokus

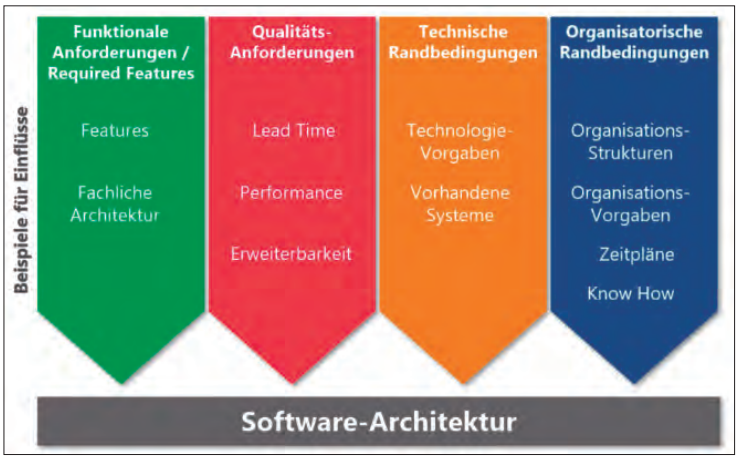
Welchen Stellenwert hat nun Qualität für den Systementwurf? Obwohl viel über Qualität geredet wird, so liegt im Entwicklungsalltag das Hauptaugenmerk vieler Beteiligter häufig auf den funktionalen Anforderungen – Qualitätsanforderungen stehen nicht im Fokus. Wenn wir das richtige System bauen wollen (und nicht irgendein System), ist das ein Problem.

Die Architekturdisziplin hat insgesamt eine Schlüsselrolle inne. Computerwissenschaftler Ralph Johnson [4][5] ▶

schreibt: „Architecture is the decisions that you wish you could get right early in a project." Anforderungen können schnell geändert und das Detailed Design kann bei Bedarf lokal überarbeitet werden. Es ist dagegen extrem aufwendig, architekturelle Konzepte grundlegend zu überarbeiten, sobald die Architektur steht. In vielen Fällen wird dies das Projekt in Schieflage oder gar zum Scheitern bringen.

Sehen wir uns also an, durch welche Einflüsse eine Softwareearchitektur getrieben wird (siehe Bild 3). Neben den funktionalen Anforderungen sind das Qualitätsanforderungen und Randbedingungen. Dabei sind es genau die beiden Letzteren, die die Architektur häufig am stärksten prägen.

Es ist daher mehr als sinnvoll, Qualität explizit zu machen und als treibenden Faktor der Architekturarbeit – und weiter gefasst der gesamten Entwicklungsarbeit – in den Fokus zu rücken.



Einflüsse auf die Architektur eines Softwaresystems: Einflussfaktor-kategorien und einzelne beispielhafte Einflussfaktoren (Bild 3)

Qualität von der Requirement-Analyse bis zum Test

Die Frage ist, während welcher Phasen im Vorgehensmodell und bei welchen Aktivitäten wir Qualität weiter unter die Lupe nehmen wollen. Die Antwort darauf fällt ziemlich simpel aus. Wenn wir akzeptieren, dass Qualitätsanforderungen die Softwarearchitektur maßgeblich beeinflussen, und wenn wir es ernst damit meinen, Qualität methodisch herbeiführen zu wollen – dann müssen wir Qualität während aller Phasen im Vorgehensmodell auf dem Schirm haben.

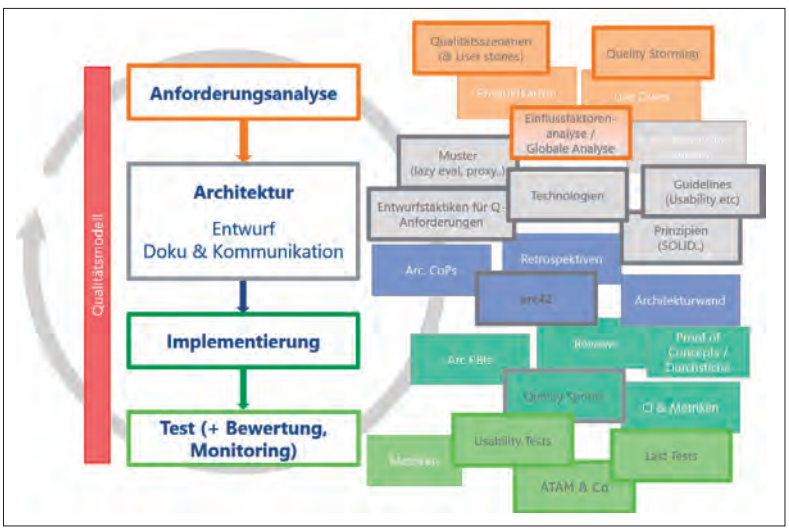
Somit begleitet uns das zu Beginn ausgewählte Qualitätsmodell von der Anforderungsanalyse bis zum Test. Nicht nur das: Konsequent zu Ende gedacht, müssen wir sie kontinuierlich während des gesamten Produktlebenszyklus betrachten und entsprechende Maßnahmen treffen.

Bild 4 zeigt beispielhaft einige Bausteine (Methoden, Aktivitäten, Konzepte et cetera), die man an den entsprechenden Stellen im Vorgehensmodell benutzen könnte. Bei einigen steht Qualität ohnehin im Zentrum, manchmal sogar im Namen (zum Beispiel Quality Storming [6]). Bei anderen spielen Qualität beziehungsweise gewisse Qualitätsaspekte zumindest eine Rolle. Oder sie bieten sich an, um das Thema Qualität transparent zu machen und kontinuierlich mitzuverfolgen – zum Beispiel während der Retrospektive.

Selbstverständlich liegen nicht alle Phasen und Bausteine in der Hauptverantwortung der Architekturdisziplin. Aber beispielweise auch während der Implementierung sollten Architekten und Architektinnen mit Rat und Tat zur Seite stehen. So könnten sie zum Beispiel Entwickler:innen coachen oder beim Entwurf von Review-Guidelines dafür verantwortlich sein, Qualitätsaspekte zu adressieren.

Systementwurf: eine Methodik

Nun wollen wir uns aber doch noch einmal die Kernaufgabe der Architektinnen und Architekten ansehen, den Systementwurf. Wie kommen wir zu einem Entwurf, der nicht nur die Umsetzung der funktionalen Anforderungen ermöglicht und Randbedingungen berücksichtigt, sondern auch die Qualitätsanforderungen ins Zentrum rückt? Dazu soll nun eine mögliche Methodik vorgestellt werden. Wir betrachten das Ganze erst einmal aus hoher Flughöhe. Bild 5 dient dabei als Diskussionsobjekt. Im Anschluss daran werden wir die einzelnen hier nur angerissenen Stationen genauer beleuchten.



Bausteine der Qualität: Wo kann ich Qualität beachten? (Bild 4)

Anforderungen

Bevor etwas entworfen werden kann, muss sinnvollerweise zumindest in Teilen feststehen, was eigentlich gebaut werden soll. Das methodische Herausarbeiten und Dokumentieren dieser Anforderungen sollte im Wesentlichen Aufgabe des Requirement Engineerings sein. Dennoch haben wir Architekten und Architektinnen hier ein Wörtchen mitzureden. Es ist unsere Aufgabe, die aufgestellten Anforderungen zu hinterfragen und offene Punkte zu klären. Dazu gehört, Wechselwirkungen zwischen verschiedenen Quali-

tätsanforderungen sowie Randbedingungen zu berücksichtigen und daraus resultierende Konflikte und Risiken transparent zu machen. Denn nur dann kann man mit diesen bewusst umgehen. In der Realität sieht es leider manchmal so aus, dass Anforderungen nur lückenhaft oder sogar gar nicht erhoben wurden: Dann heißt es in den sauren Apfel beißen und uns selbst darum kümmern, dass die notwendige Basis für die eigentliche Architekturarbeit geschaffen wird.

Einflussfaktoren

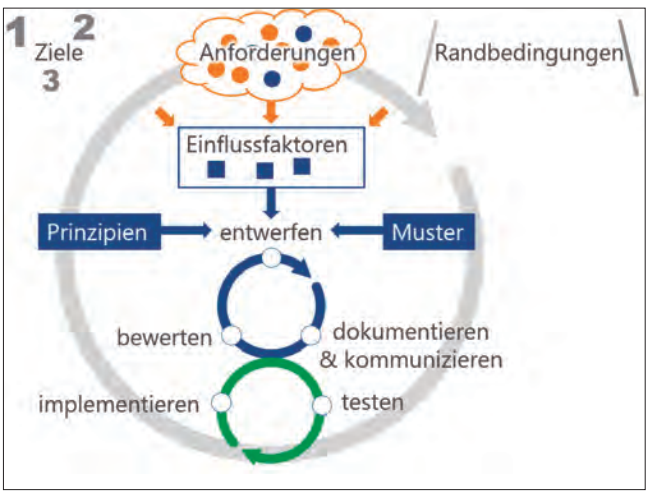
Die grobgranularen Geschäfts- und Architekturziele, detailliertere Anforderungen sowie Randbedingungen bilden zusammen mit dem Erfahrungsschatz der Architektinnen und Architekten den Pool an Eingangsdaten, aus dem die kritischen Einflussfaktoren herausgearbeitet werden. Bei der Einflussfaktorenanalyse geht es darum, die Faktoren zu spezifizieren, hinsichtlich gewisser Kriterien zu analysieren, um sie dann zu priorisieren und passende Entwurfsstrategien entwickeln zu können.

Entwurf, Dokumentation und Bewertung

Bei der Entwurfsarbeit helfen uns Taktiken, Prinzipien und Muster, auf die wir zurückgreifen können. Bevor es dann in Richtung Implementierung und Test geht, ist es enorm wichtig, die bisherigen Arbeitsergebnisse strukturiert zu dokumentieren. Damit Dokumentation ihren Zweck erfüllen und zudem effizient geschrieben werden kann, sollte dies immer zeitnah geschehen. Unter anderem gibt uns dies die Möglichkeit, erste qualitative Bewertungen fundamentaler Entscheidungen rechtzeitig durchzuführen. Das heißt, schon bevor Aufwände in Design oder Implementierung geflossen sind.

Nach der Iteration ist vor der Iteration

Aus der Implementierung schließlich gewinnen wir wichtiges Feedback, denn wir nutzen sie, um die bisher getroffenen Architekturentscheidungen auf den Prüfstand zu stellen. Die Erkenntnisse daraus bilden zusammen mit (eventuell) neu eintrudelnden beziehungsweise nachgeschärften Anforderungen die Basis für die nächste Iteration.



Eine zeitgemäße Methodik für den Systementwurf (Bild 5)

Die sogenannten „nichtfunktionalen Anforderungen“

Anforderungen werden gerne unterteilt in funktionale und nichtfunktionale Anforderungen. Zumindest letzteren Begriff findet der Autor unglücklich. Nichtfunktionale Anforderungen klingen irgendwie nach Anforderungen zweiter Klasse. Der Autor ist sich sicher, dass manches Unterbewusstsein damit etwas Unnützes, Negatives assoziiert, denn begrifflich liegen die Wörter funktional und funktionell ziemlich eng beieinander. Der Wichtigkeit und Bedeutung dieser nichtfunktionalen Anforderungen wird die Assoziationskette *nichtfunktional => nicht funktionell => nicht die eigentliche Funktion erfüllend (also nicht funktionierend)* leider alles andere als gerecht. Lasst uns also den Begriff nichtfunktionale Anforderungen ein für alle Mal streichen. Stattdessen sprechen wir ab sofort nur noch von Qualitätsanforderungen.

Die Anforderungsanalyse

Der Fokus dieses Artikels liegt zwar auf Architekturarbeit, wie schon erwähnt haben wir Architekten und Architektinnen bei der Anforderungsanalyse aber ein Wörtchen mitzureden, und manchmal müssen wir hier mehr tun, als uns lieb ist. Betrachten wir Qualität als treibenden Faktor, müssen wir folglich auch wissen, wie wir dies anforderungsseitig tun.

Qualitätsszenarien

Da stellt sich die Frage, wie wir Qualitätsanforderungen formulieren können. In der Realität sehen wir regelmäßig Aussagen wie „Der Code soll erweiterbar sein“, „Das System muss ausfallsicher sein“ und so weiter. Allerdings sind solche Anforderungen, die lediglich ein Qualitätsattribut nennen, maximal vage und unspezifisch. Sie erlauben es weder, geeignete Strategien abzuleiten, um die tatsächliche Anforderung effizient und möglichst effektiv zu erfüllen, noch ermöglichen sie es, die Zielerreichung zu überprüfen, da solche Aussagen nicht validierbar sind. Anforderungen dieser Art sind also völlig unbrauchbar.

Qualitätsszenarien sind stattdessen das Mittel der Wahl. Sie sind zentrales Element szenariobasierter Architekturevaluierungsmethoden wie ATAM (Architecture Tradeoff Analysis Method) [7]. Ebenso sollten sie in einer Architekturdokumentation eine wichtige Rolle spielen, wie beispielsweise im arc42-Template [8] der Fall.

Qualitätsszenarien sind kurze Beschreibungen, die Qualitätsanforderungen konkretisieren und dem EVA-Prinzip folgen. Bild 6 veranschaulicht dies schematisch. Das Schema definiert eine Quelle, die einen Stimulus generiert. Dieser wirkt auf einen Systembestandteil und hat eine Antwort des Systems zur Folge. Mittels einer Antwortmetrik kann eine Vorgabe an das System überprüft werden.

Das Ganze wird greifbarer, wenn wir uns ein konkretes Szenario anschauen (Bild 7). Das konkrete Änderungsszenario aus dem Beispiel könnte also etwas verknappt folgendermaßen formuliert werden: „Eine Änderung der Beitrags- ▶

berechnung ist mit < 1 PT Aufwand realisierbar." Die Praxis zeigt, dass das Schreiben guter Szenarien meist nicht auf Anhieb gelingt, sondern Übung verlangt. Es sollte uns am Herzen liegen, in die Qualität von Qualitätsanforderungen zu investieren, da diese letztendlich den Kern qualitätsgetriebener Architekturarbeit bilden.

Quality Storming

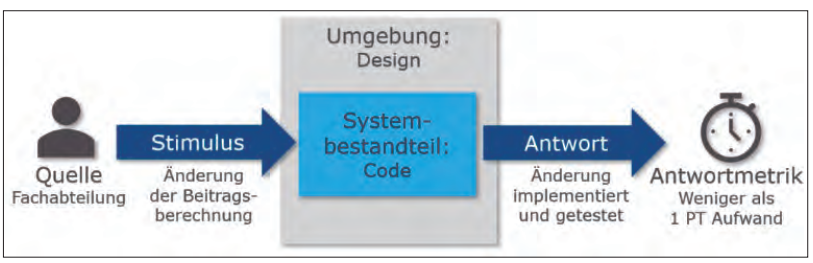
Quality Storming ist eine Workshop-basierte Methode, um Qualitätsanforderungen zu erheben, und gehört zu den kollaborativen Modellierungstechniken. Der Begriff ist angelehnt an verwandte Storming-Methoden (Event Storming [9], Gamestorming [10] und so weiter), die vor allem im Bereich des Domain Driven Design sehr beliebt sind. In einem mehrstündigen Workshop erarbeitet eine heterogene Gruppe von Stakeholdern eine Liste konsolidierter, priorisierter Qualitätsanforderungen, idealerweise in Form von Qualitätsszenarien. Grundlage ist auch hierfür wieder ein vernünftiges Qualitätsmodell. Ganz im Sinne von Domain Driven Design dreht es sich bei Quality Storming [5] darum, ein gemeinsames Verständnis des zu bauenden Systems zu entwickeln – in diesem Fall mit besonderem Fokus auf den Qualitätsanforderungen.

Die Einflussfaktorenanalyse

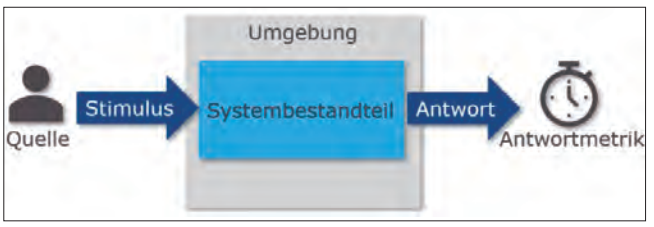
Wir erinnern uns, dass die funktionalen Anforderungen üblicherweise nicht den stärksten Einfluss auf die Architektur eines Systems haben. Randbedingungen und ganz besonders die Qualitätsanforderungen an das System spielen hier eine größere Rolle. Für Letztere haben wir Qualitätsszenarien als passendes Werkzeug vorgestellt. Bei der Einflussfaktorenanalyse nutzen wir Qualitätsszenarien, um kritische Faktoren zu präzisieren und messbar zu machen. Weiterhin werden diese Faktoren hinsichtlich Flexibilität, Veränderlichkeit und Einfluss analysiert, um sie anschließend zu priorisieren. Diese Methodik erläutern Christine Hofmeister et al. als Teil der von ihnen beschriebenen Globalen Analyse (Bild 8) [11].

Kritische Einflussfaktoren identifizieren

Zu Beginn werden die kritischen Einflussfaktoren identifiziert. Das sind diejenigen, die einen bedeutenden globalen Einfluss haben, schwierig zu erfüllen sind (eventuell aufgrund mangelnder Erfahrung) und für die zukünftige Änderungen wahrscheinlich sind. Als Quelle dienen Geschäfts- und Qualitätsziele, Anforderungen, Randbedingungen – sowie des Weiteren die Erfahrung des Entwicklungsteams.



Schematische Beschreibung von Qualitätsszenarien am Beispiel eines Änderungsszenarios (Bild 7)



Schematische Beschreibung von Qualitätsszenarien (Bild 6)

Präzisieren und messbar machen

Im Anschluss daran werden die kritischen Faktoren spezifiziert: Sie werden präzisiert und messbar gemacht, und zwar mithilfe von Qualitätsszenarien. Weiterhin schlägt die Globale Analyse eine Kategorisierung der Faktoren in technologische, organisatorische und Produktfaktoren vor.

Flexibilität, Veränderlichkeit und Einfluss analysieren

In der nächsten Phase geht es darum, die Faktoren hinsichtlich Flexibilität, Veränderlichkeit und Einfluss zu analysieren: Ist der Faktor verhandelbar? Welche Spielräume gibt es bei der Umsetzung? Muss mit Änderungen bei dem Faktor gerechnet werden, zum Beispiel durch neue Anforderungen oder Technologien? Wirkt der Faktor auf einen oder mehrere Bausteine oder auf andere Faktoren?

Betrachtet man die verschiedenen Einflussfaktoren, so sind diese nämlich nicht alle komplett unabhängig voneinander: Manche dieser Faktoren unterstützen sich, andere stehen im Widerspruch zueinander. Beim Entwurf müssen wir dies mitberücksichtigen.

Priorisieren mittels Utility Tree

Zuvor ist es allerdings sinnvoll, die Einflussfaktoren zu priorisieren, selbst wenn wir bei der obigen Identifikation nur kritische Einflussfaktoren in die Betrachtung mit einbezogen haben. Denn natürlich gibt es auch unter diesen Faktoren Unterschiede hinsichtlich ihrer Priorität.

ATAM (Architecture Tradeoff Analysis Method) [7] schlägt die Priorisierung auf Basis eines Quality- beziehungsweise Utility Trees [7] vor. Für jedes Blatt des Baums (hier stehen die Qualitätsszenarien) werden zwei Dimensionen betrachtet: die Wichtigkeit der Qualitätsanforderung für den Erfolg des Systems und die Höhe des Risikos für deren Umsetzung. Eine Anforderung, die enorm wichtig für den Erfolg ist und deren Erfüllung gleichzeitig mit einem hohen Risiko verbunden ist, wird dementsprechend hoch priorisiert.

Obwohl ATAM eine Methode zur Architekturbewertung ist, können wir dieses Vorgehen natürlich wunderbar auf eine Einflussfaktorenanalyse übertragen. Für weitere Details möchte der Autor noch mal auf den frei verfügbaren Technical Report zu ATAM verweisen [7]. Dieser ist trotz seines Erscheinungsdatums im Jahr 2000 in großen Teilen noch immer relevant. Zudem ist er sehr gut zu lesen, weshalb es sich für alle Architekturinteressierten lohnt, hier einmal einen Blick hineinzuwerfen.

Entwurfsarbeit

Die Ergebnisse der Einflussfaktorenanalyse sind Input für die nachfolgende Entwurfsphase. Aus der Gesamtbetrachtung der hoch priorisierten Einflussfaktoren kristallisieren sich zentrale Architekturthemen heraus, für welche wir Entwurfsstrategien entwickeln müssen. Diese Aspekte sind ebenfalls in [11] als Teil der Globalen Analyse beschrieben.

Architekturthemen identifizieren

Architekturthemen sind die Themen, die für das zu entwerfende System zentral beziehungsweise risikobehaftet sind.

Häufig gehören zu solchen Themen mehrere kritische Einflussfaktoren, die zueinander in starker Wechselwirkung stehen. Sie können sich ergänzen oder einander widersprechen. Ein klassisches Beispiel für Einflussfaktoren, die im Widerspruch zueinander stehen, wäre die gleichzeitige Forderung nach hoher Performanz und guter Portierbarkeit. Letztere lässt sich über zusätzliche Abstraktionsschichten erreichen, welche allerdings einen negativen Einfluss auf Speicherverbrauch und Zeitverhalten haben. Bestimmte Technologievorgaben als Randbedingung wiederum können gut oder schlecht zu anderen Einflussfaktoren passen: Man denke zum Beispiel an den Einfluss der Programmiersprache auf Attribute wie Performanz, Wartbarkeit oder Robustheit.

Entwurfsstrategien entwickeln

Für diese Architekturthemen gilt es nun, Lösungsstrategien zu erarbeiten. Zu den Strategien gehört die Anwendung von Taktiken, Patterns und Prinzipien sowie die Auswahl und Kombination passender Technologien. Eine mögliche Strategie, um eine kurze Lead Time zu erreichen, wäre beispielsweise die Wahl einer Microservice-Architektur. Je nach Domäne und Systemart sind Qualitätsanforderungen und daraus resultierende Lösungsstrategien aber sehr unterschiedlich. Diesen Aspekt können wir nicht weiter vertiefen, denn allein Muster für spezielle Einsatzzwecke füllen viele Bücher, und Technologien sind im steten Wandel. Wir alle müssen die Konzepte studieren und die Erfahrung aufbauen, die wir im eigenen Umfeld benötigen. Dafür gibt es keine Abkürzung.

Konflikte transparent machen

Das Entwickeln tragfähiger Strategien – als wesentlichem Teil der eigentlichen Entwurfsarbeit – erfordert, dass wir Architekt:innen Konflikte rechtzeitig transparent machen und mit den Stakeholdern thematisieren. Dies ist eine unserer Hauptverantwortlichkeiten. So mancher scheinbar kaum lösbare Konflikt kann sich auf diese Art in Luft auflösen oder deutlich entschärft werden. Damit vermeiden wir häufig noch rechtzeitig das Setzen falscher Prioritäten oder verhindern die Verschwendung von Ressourcen durch das Implementieren aufwendiger Lösungen.

Globale Analyse nach Hofmeister (Bild 8)

Dokumentation

Wenn wir Qualität ins Zentrum unserer Arbeit rücken, so gilt dies auch für die Dokumentation. Nicht aus Selbstzweck: Gute Dokumentation ist die Basis dafür, dass Architektur nachhaltig ist und langfristig methodisch erarbeitet werden kann. Zu guter Architekturarbeit gehört eine gute Dokumentation.

Qualitätsattribute von Dokumentation

Letztlich sollten wir bezüglich der Qualität von Dokumentation ähnliche Betrachtungen vornehmen wie für das zu entwerfende System: Einflussfaktoren und Qualitätsattribute dienen hier als Schlagworte. Je nach System und Projekt müssen wir entscheiden, ob die Dokumentation ausführlich oder knapp sein soll, für welche Stakeholder wir sie schreiben und welches Format und Tooling wir für das Schreiben verwenden. Für ein kleines, unkritisches System sollten wir nicht dieselben Maßstäbe anlegen wie für eine komplexe Software, die über Jahre hinweg weiterentwickelt wird.

Rechtzeitig dokumentieren!

Es hilft jedoch nichts, wenn Dokumentation im Nachgang entsteht, mit heißer Nadel gestrickt. Solche reine Prozessbefriedigung verfehlt das eigentliche Ziel komplett, die Entwickler:innen und Architektinnen und Architekten bei ihrer Arbeit zu unterstützen. Darüber hinaus ist das Schreiben von Dokumentation auf den letzten Drücker auch noch extrem ineffizient und demotivierend: ein Patentrezept, um Projekte in Schieflage zu bringen und gute Mitarbeiter zu vergraulen.

Stattdessen sollte Dokumentation frühzeitig und inkrementell erfolgen, begleitend zu den Analyse-, Entwurfs- und Implementierungstätigkeiten. Bevor wir auch nur angefangen haben, die ersten Zeilen Code zu schreiben, sollten wir also schon viele Arbeitsergebnisse dokumentiert haben.

Dazu zählen mindestens die Resultate der Anforderungsanalyse, der Einflussfaktorenanalyse und entwickelte Strategien und Konzepte. Dies erlaubt es uns außerdem, eine erste, frühe (das heißt vor der Implementierung stattfindende) Architekturbewertung vorzunehmen. Wer einmal gespürt hat, wie es sich anfühlt, wenn gute Dokumentation ernsthaft gelebt wird, wird Dokumentieren hoffentlich nicht mehr als lästige Pflicht empfinden, sondern als befriedigenden, anspruchsvollen Teil einer zeitgemäßen Entwicklungsarbeit.

Qualität im arc42-Template

Was die Architekturdokumentation angeht, so hält der Autor die Struktur nach dem arc42-Template [8] in den meisten Fällen für die beste Wahl.

Es ist fundamental wichtig, Qualität methodisch zu betrachten. Denn wir wollen nicht irgendein System bauen, sondern das richtige System. Als Quasistandard für Softwarearchitekturdokumentation im deutschsprachigen Raum beleuchtet deswegen arc42 Qualität explizit:

- Qualitätsziele sind essenzieller Bestandteil von Kapitel 1 „Einführung und Ziele".
- Kapitel 10 heißt „Qualitätsanforderungen" und beinhaltet den Qualitätsbaum sowie Qualitätsszenarien. ▶

- Kapitel 4 „Lösungsstrategie" und vor allem Kapitel 8 „Querschnittliche Konzepte" betrachten die Lösungsseite und damit zu großen Teilen die Konzepte, mit denen Qualitätsziele und -anforderungen unter gegebenen Randbedingungen umgesetzt werden.
- Da Qualitätsanforderungen die Architektur maßgeblich beeinflussen, nehmen auf diese auch Kapitel 9 „Entwurfsentscheidungen" und Kapitel 11 „Risiken" maßgeblich Bezug.

Implementierung

Die Implementierung liegt zwar außerhalb der Hauptverantwortlichkeit der Architekturdisziplin. Dennoch ist es unsere Aufgabe als Architekten und Architektinnen, sicherzustellen, dass die Implementierung die Entwurfskonzepte umsetzt. Neben dem nachträglichen Abgleich von Code und Entwurf mittels Reviews und Analysetools sind wir dafür verantwortlich, das Team in Architekturfragen zu unterstützen. Und wir tragen Sorge dafür, dass ein gemeinsames Verständnis bezüglich Konzepten, Strategien und Entscheidungen besteht.

Wir sollten die Möglichkeit nutzen, Qualitätsziele in der täglichen Entwicklungsarbeit zu verankern.

Über Review Guidelines können wir gezielt den Blick auf Qualitätsattribute wie Sicherheit, Benutzerfreundlichkeit oder Wartbarkeit lenken. Experten für diese Themen können als zusätzliche Reviewer eingesetzt werden. Natürlich können uns auch sinnvoll eingesetzte Metriken, die durch die CI-Pipeline berechnet werden, unterstützen.

Stand-ups und vor allem Retrospektiven als regelmäßige Events sind eine wunderbare Gelegenheit, um Qualitätsthemen ständig sichtbar im Fokus zu halten. Es ist die Aufgabe von uns Softwarearchitektinnen und -architekten, uns darum zu kümmern, dass unsere Themen präsent sind. Und natürlich sollte auch das Backlog Qualitätsaspekte und architekturelle Tätigkeiten explizit widerspiegeln [12].

Test, Bewertung, Monitoring

Die schönsten Konzepte und Lösungsstrategien auf dem Papier taugen am Ende nichts, wenn sie sich nicht in der Praxis bewähren. Anhand der Implementierung können wir überprüfen, ob die definierten Qualitätsziele erreicht wurden.

Feedback sinnvoll nutzen

Durch geschicktes Auswählen dessen, was wir früh implementieren, können wir architektonische Risiken rechtzeitig unter die Lupe nehmen. Das Feedback aus der Implementierung zeigt uns somit früh, ob wir auf dem richtigen Weg sind oder ob sich die Risiken als Probleme manifestieren und wir Lösungskonzepte überarbeiten müssen. Tests, zum Beispiel Lasttests, als quantitative Bewertungsverfahren und szenariobasierte Methoden gehen dabei Hand in Hand, und vernünftig ausgewählte Metriken können zusätzlich unterstützen.

Qualität ist nie fertig

Das Thema Qualität ist mit dem Inverkehrbringen der Software mitnichten abgeschlossen. Ein System ist nicht ein für alle Mal sicher oder performant – Sicherheitslücken werden aufgedeckt, Nutzungskontext und Lastprofile ändern sich über die Zeit. Und was die funktionelle Seite der Anforderungen betrifft, wird Software um neue oder verbesserte Features erweitert, was wiederum Auswirkungen auf die Qualitätsattribute des Systems haben kann („Werden die Qualitätsszenarien für Wartbarkeit, Performanz et cetera noch erfüllt?", „Braucht es neue Qualitätsszenarien?").

Qualität betrifft also den gesamten Software-Lebenszyklus. Je nach System müssen wir uns deshalb zum Beispiel auch Gedanken dazu machen, wie wir das Monitoring von Qualitätskennzahlen bewerkstelligen, um frühzeitig potenziellen Qualitätsproblemen im Produktivbetrieb entgegenzuwirken.

Fazit

Auf dem hart umkämpften Softwaremarkt ist Qualität ein erfolgsentscheidender Faktor. Deswegen verwundert es, dass Qualität zwar vollmundig beworben, in der Praxis aber häufig das Bauchgefühl konsultiert wird, wenn es darum geht, ein komplexes System zu bauen. Eine hohe Qualität bedeutet, dass wir das richtige System gebaut haben. Wenn wir uns nicht nur auf unser Glück verlassen wollen, ist hierfür eine methodische, gesamtheitliche Vorgehensweise unabdingbar. Diese umfasst den gesamten Software-Lebenszyklus.

Dabei ändern sich auch Qualitätsziele und Qualitätsanforderungen, sodass Qualität nie fertig ist, sondern verlangt, dass wir sie kontinuierlich betrachten. Am Ende ist so viel sicher: Qualität steht für uns Softwarearchitekten und -architektinnen als treibender Faktor im Zentrum unserer Arbeit. Und wenn wir das ernst nehmen, wird uns so schnell sicher nicht langweilig. ■

[1] ISO 25010, www.dotnetpro.de/SL2301ArchQuality1
[2] ISAQB, www.isaqb.org/de/
[3] ISO 9126, www.dotnetpro.de/SL2301ArchQuality2
[4] Ralph E. Johnson, www.dotnetpro.de/SL2301ArchQuality3
[5] Mailinglist, www.dotnetpro.de/SL2301ArchQuality4
[6] Quality Storming, www.dotnetpro.de/SL2301ArchQuality5
[7] ATAM, www.dotnetpro.de/SL2301ArchQuality6
[8] Arc42, https://arc42.de
[9] Event Storming, www.dotnetpro.de/SL2301ArchQuality7
[10] Gamestorming, www.dotnetpro.de/SL2301ArchQuality8
[11] Globale Analyse, www.dotnetpro.de/SL2301ArchQuality9
[12] Stefan Toth, Vorgehensmuster für Softwarearchitektur, Hanser Verlag, ISBN-13: 9783446460041

Lutz Marquardt
ist Principal Software Engineer und zertifizierter Architekturtrainer bei Method Park by UL Solutions. Er ist überzeugter Documentarian und Clean Coder mit Themen rund um Architektur, DevOps und Microservices.
lutz.marquardt@ul.com

dnpCode A2301ArchQuality