KAPITEL 2

NAVIGATIONSHILFE: FUNKTIONALITÄT UND SOFTWARE-ARCHITEKTUR

Bevor eine Softwarearchitektur erstellt werden kann, müssen gemäss der Architektur-Definition die gewünschten Eigenschaften festgelegt werden. Diese Eigenschaften bilden die Inputs für die Architektur-

Erstellung.

Doch woraus bestehen diese Inputs? Handelt es sich dabei um die Funktionen, die das Software-System erfüllen muss?

Um diese Fragen zu klären, möchte ich mit einem kleinen Gedankenexperiment starten: Stellen Sie sich vor, sie müssten eine Software entwickeln, welche die einfache Aufgabe hat, zwei Zahlen zu addieren. Dazu würden Sie verschiedene Software-Entwickler beauftragen.

Basierend auf der Definition der Softwarearchitektur stellt sich diese Aufgabe folgendermassen dar:

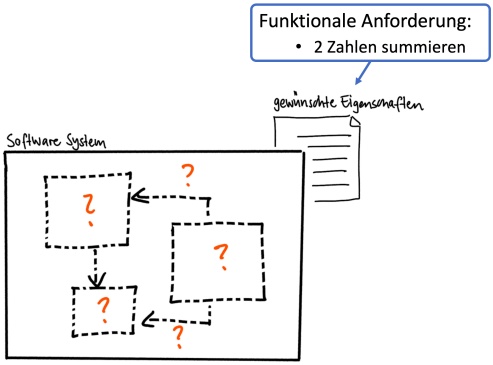


Abbildung 6: Gewünschte Eigenschaften des Software-Systems ist das Summieren von 2 Zahlen.

Trotz dieser einfachen Anforderung würden die resultierenden Lösungen der Software-Lieferanten erheblich variieren. Jeder Entwickler hätte eine andere Vorstellung davon, wie eine solche Software strukturiert sein sollte

und welche Technologien oder Plattformen dafür ideal wären.

Ein Entwickler würde vielleicht eine Webanwendung, ein anderer eine einfache Dialoganwendung und wieder ein anderer ein komplexes User Interface erstellen. Jede dieser Varianten würde die Grundanforderung – das Addieren zweier Zahlen – erfüllen, aber auf sehr unterschiedliche Weise und mit unterschiedlichen Strukturen.

Daraus lassen sich mehrere wichtige Schlüsse ziehen:

- **Funktionalität**: Die Funktionalität eines Systems bezieht sich auf die Fähigkeit, die Arbeit zu verrichten, für die es vorgesehen war. In unserem Beispiel ist das die Addition zweier Zahlen.
- **Vielfalt der Architekturen**: Diese Funktionalität kann durch viele verschiedene Architekturen erreicht werden. Die Wahl der Architektur kann von vielen Faktoren abhängen, einschliesslich der vorhandenen Technologien, der Präferenzen der Entwickler oder der Entwicklerteams oder spezifischer Anforderungen an Leistung und Skalierbarkeit.
- **Übereinstimmung mit Stakeholder-Erwartungen**: Selbst, wenn alle funktionalen Anforderungen erfüllt werden, bedeutet das nicht zwangsläufig, dass das resultierende System auch das ist, was der Kunde oder die Stakeholder erwarten oder benötigen.

Die Relevanz der Softwarearchitektur liegt also nicht nur darin, dass sie eine bestimmte Funktion ermöglicht, sondern auch darin, wie sie diese Funktion erfüllt und in welchem Masse sie die breiteren, oft impliziten Erwar-

tungen der Nutzer und Stakeholder erfüllen kann. Eine gut durchdachte Architektur ist entscheidend, um nicht nur die gegenwärtigen und offensichtlichen funktionalen Anforderungen effizient zu erfüllen, sondern auch zukünftige Erweiterungen und Modifikationen zu erleichtern, ohne dass eine Restrukturierung erforderlich ist.

Die nachfolgende Visualisierung veranschaulicht diese Aspekte. Die funktionalen Anforderungen spannen einen Anforderungsraum auf, in welchem die darin möglichen Lösungen unterschiedliche Eigenschaften haben können.

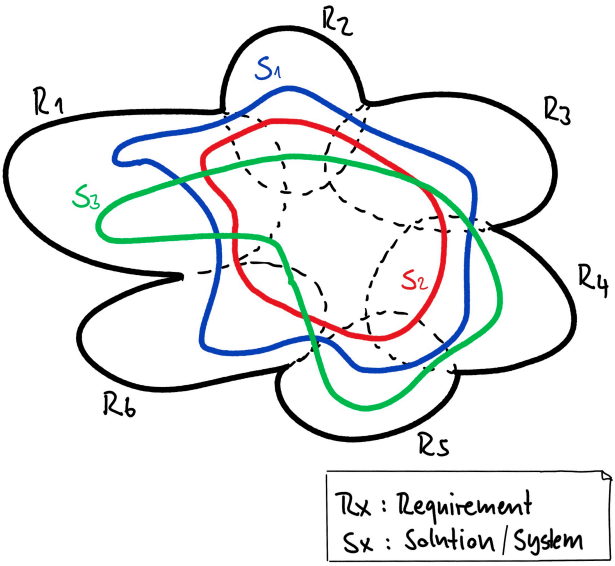


Abbildung 7: Visualisierung des Anforderungsraumes über die funktionalen Anforderungen - und die darin mögliche Lösungen.

Achten Sie stets auf eine klare und unmissverständliche Trennung zwischen Anforderungsraum und Lösungsraum. In Meetings, Workshops, Lösungsdiskussionen und auch in Dokumenten sollten diese beiden Bereiche strikt voneinander getrennt werden. Der Anforderungsraum definiert, was erreicht werden muss, während der Lösungsraum sich damit beschäftigt, wie diese Anforderungen erfüllt werden können. Diese klare Trennung hilft, Missverständnisse zu vermeiden, den Fokus zu bewahren und sicherzustellen, dass Lösungsdiskussionen erst nach Klärung der Anforderungen stattfinden.

Obwohl alle Lösungsvarianten die funktionalen Anforderungen komplett erfüllen, sind die Eigenschaften der Lösungen dennoch unterschiedlich. Die Unterschiede liegen in der Erfüllung der Qualitätsanforderungen wie nachfolgende Visualisierung zeigt.

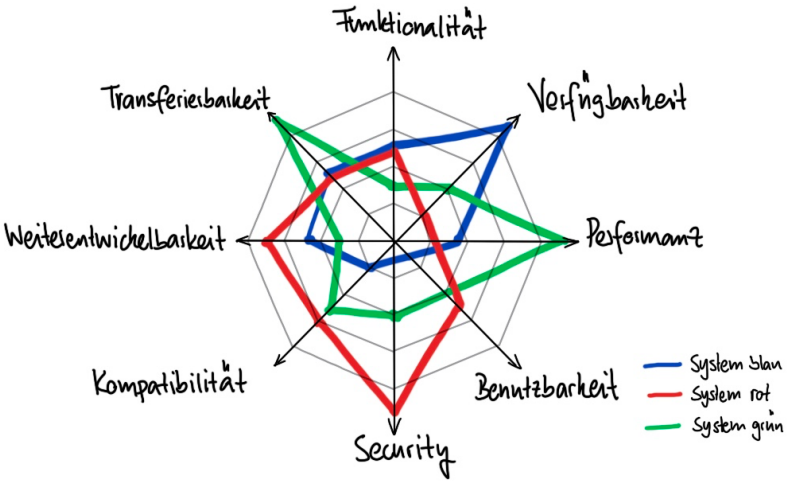


Abbildung 8: Unterschiedliche Lösungen haben unterschiedliche Qualitäts-Eigenschaften.

Das Ziel einer Softwareentwicklung muss sein, nicht eine beliebige Lösung, welche die funktionalen Anforderungen erfüllt, sondern geplant z.B. die «grüne» Lösung mit ihren Eigenschaften, zu erstellen.

Es reicht also nicht aus, die funktionalen Anforderungen als Input für die Softwarearchitektur vorzugeben. Die vollständigen Inputs nennt man «essenzielle Anforderungen». Dazu gleich mehr im nächsten Unterkapitel.

Die Brückenbau-Challenge:

Stellen Sie sich vor, Sie sind ein erfahrener Bau-Ingenieur. Eines Tages kommt eine spannende Herausforderung auf Ihren Schreibtisch: Eine Brücke soll zwei kleine Dörfer über einen Fluss verbinden. „Kein Problem!“, denken Sie und entwerfen eine robuste, schlichte Brücke. Funktional, stabil, perfekt.

Sie präsentieren Ihren Entwurf dem Projektteam und – oh Schreck! – die Reaktionen sind gemischt. Frau Meyer, die Bürgermeisterin, träumt von einer eleganten, ästhetischen Brücke, die ein Wahrzeichen der Region wird. Herr Schmidt, der Transportunternehmer, fordert eine breite Fahrbahn für schwere LKWs. Und die Umweltbehörde? Sie sorgt sich um die lokale Tierwelt und verlangt eine nachhaltige Bauweise. Ihr funktionaler Entwurf? Nicht schlecht, aber irgendwie auch nicht das, was alle wollen.

Was tun? Ein Workshop muss her! Gemeinsam mit dem Team setzen Sie sich zusammen, um die Anforderungen zu klären. Jeder darf seine Wünsche äussern, doch schnell wird klar: Viele Anforderungen sind gar nicht so klar, wie man dachte.

Also, ran an die Arbeit! Was bedeutet „ästhetisch ansprechend“? Was heisst „nachhaltig“? Und „robust“? Gemeinsam konkretisieren Sie die Anforderungen: Die Brücke muss mindestens 30 Tonnen tragen, um schwere LKWs sicher zu transportieren. Sie braucht Korridore für Wildtiere, um deren Lebensräume zu schützen. Und

sie soll wirklich ein Hingucker sein, ein echtes Wahrzeichen!

Mit diesen klaren, messbaren (Qualitäts-) Anforderungen machen Sie sich wieder an den Entwurf. Diesmal wissen Sie genau, was alle wollen und wie Sie es umsetzen können.

Was lernen wir daraus? Qualitätsanforderungen sind entscheidend. Sie sorgen dafür, dass alle Beteiligten dasselbe Bild im Kopf haben und dass das Endprodukt nicht nur funktional, sondern auch qualitativ den Erwartungen entspricht.

Die Geschichte zeigt: Missverständnisse lassen sich vermeiden, wenn Qualitätsanforderungen explizit und messbar definiert werden. So wird aus einem funktionalen Entwurf ein echtes Meisterwerk, das alle Erwartungen erfüllt.

DIE ESSENZIELLEN ANFORDERUNGEN

Im vorhergehenden Kapitel ist klar geworden, dass neben der Funktionalität, d.h. neben den funktionalen Anforderungen, die die spezifischen Aufgaben eines Systems beschreiben, weitere Aspekte berücksichtigt werden müssen. Die sogenannten essenziellen Anforderungen für die Erstellung einer Softwarearchitektur bestehen aus **funktionalen Anforderungen**, **Qualitätsanforderungen** und **Rahmenbedingungen**. Alle diese Inputs zu berücksichtigen ist entscheidend, um sicherzustellen, dass die entwickelte Software nicht nur funktionsfähig ist, sondern alle gewünschten Eigenschaften erfüllt.

Die nachfolgende Darstellung zeigt die bereits weiter oben eingeführten Visualisierungen zu Anforderungsraum und mögliche Lösungen. Zu den essenziellen Anforderungen gehören jedoch auch noch die Rahmenbedingungen, welche den Lösungsraum einschränken.

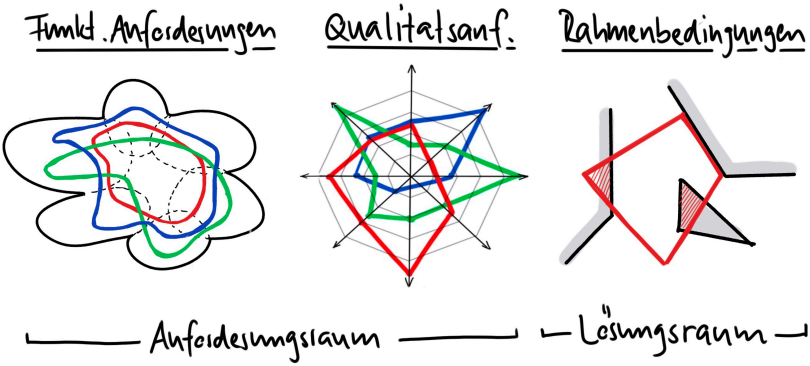


Abbildung 9: Die essenziellen Anforderungen bestehen aus den funktionalen Anforderungen, den Qualitätsanforderungen und den Rahmenbedingungen.

Nachfolgend die 3 Teile der essenziellen Anforderungen im Detail:

FUNKTIONALE ANFORDERUNGEN

Die funktionalen Anforderungen legen fest, welche spezifischen Aufgaben und Funktionen ein System erfüllen muss und wie es auf bestimmte Ereignisse oder Daten reagieren soll. Sie definieren klar, was das System leisten soll, und sind so formuliert, dass sie mit Tests überprüft werden können – eine funktionale Anforderung ist entweder erfüllt oder nicht erfüllt.

Interessanterweise haben funktionale Anforderungen oft weniger Einfluss auf die Gestaltung der Architektur, als man zunächst vermuten könnte. In vielen Fällen lassen sich fundierte Architekturentscheidungen treffen, selbst wenn noch nicht alle Details der funktionalen Anforderungen bekannt sind. Dies liegt daran, dass die Architektur häufig auf wiederkehrenden Mustern ba-

siert, die sich an den übergeordneten Zielen des Systems orientieren – konkreter: an den Qualitätsanforderungen, aber dazu gleich mehr. So kann der Architekturprozess auch parallel zur Verfeinerung der funktionalen Anforderungen fortgesetzt werden.

QUALITÄTSANFORDERUNGEN

Qualitätsanforderungen sind der zweite Teil der essenziellen Anforderungen. Diese werden oft fälschlicherweise mit den nicht-funktionalen Anforderungen gleichgesetzt. Nicht-funktionale Anforderungen sind jedoch ein umfassenderer Begriff, der alle Anforderungen umfasst, die sich auf die Betriebsweise des Systems beziehen und nicht auf spezifische Funktionen.

Qualitätsanforderungen sind hingegen spezifischer und messbar. Sie beziehen sich auf die Qualitätsattribute des Systems und legen fest, wie diese erfüllt werden sollen. Daher können Qualitätsanforderungen als eine Untergruppe der nicht-funktionalen Anforderungen betrachtet werden, die konkrete Kriterien vorgeben, um die Qualität des Endprodukts sicherzustellen.

Qualitätsanforderungen spezifizieren, wie die Funktionen ausgeführt werden sollen, um die Erwartungen an Qualität und Leistung zu erfüllen. Sie betreffen unter anderem Aspekte wie Antwortzeiten (Performance), Ausfallsicherheit (Zuverlässigkeit) oder Benutzerfreundlichkeit. Zu den Qualitätsanforderungen gehören aber Themen wie Weiterentwickelbarkeit, Änderbarkeit oder Skalierbarkeit. Im Gegensatz zu den funktionalen Anforderungen können nicht alle Qualitätsanforderungen

im klassischen Sinne getestet werden, um die Erfüllung oder Nicht-Erfüllung zu prüfen. Trotzdem sollten die Qualitätsanforderungen so definiert sein, dass es möglich ist zu prüfen, wie gut diese erfüllt werden.

Qualitätsszenarien

Um Qualitätsanforderungen messbar und verifizierbar zu machen, sollten Qualitätsszenarien (Quality Attribute Scenarios) verwendet werden. Ein Qualitätsszenario beschreibt eine spezifische Situation, in der die Qualität des Systems geprüft werden kann, und umfasst typischerweise ein Stimulus (eine Eingabe oder ein Ereignis), eine Antwort des Systems und messbare Kriterien, anhand derer die Antwort bewertet wird.

Es genügt nicht, einfach Begriffe wie "Erweiterbarkeit" oder "schnelles, performantes System" als Qualitätsanforderungen zu definieren. Solche allgemeinen Aussagen sind zu vage und führen nicht zu konkreten, überprüfbaren Zielen. Deshalb sind überprüfbare Qualitätsszenarien notwendig. Diese Szenarien helfen dabei, die genauen Erwartungen an die Qualität des Systems festzulegen und sicherzustellen, dass alle Beteiligten ein gemeinsames Verständnis der Anforderungen haben.

Ein Qualitätsszenario für Erweiterbarkeit könnte beispielsweise so aussehen:

- **Stimulus**: Ein Entwickler möchte eine neue Funktionalität in das bestehende System integrieren.
- **Antwort**: Das System ermöglicht die Integration der neuen Funktionalität ohne grössere Änderungen an der bestehenden Architektur.
- **Messbare Kriterien**: Die Integration der neuen Funktionalität dauert nicht länger als 4 Stunden und erfordert keine Änderungen an bestehenden Modulen.

Ein Qualitätsszenario für Performance könnte so aussehen:

- **Stimulus**: 1000 gleichzeitige Benutzer greifen auf die Webanwendung zu.
- **Antwort**: Das System verarbeitet die Anfragen innerhalb akzeptabler Antwortzeiten.
- **Messbare Kriterien**: 95% der Anfragen werden in weniger als 2 Sekunden beantwortet.

Diese klar definierten Szenarien stellen sicher, dass die Qualitätsanforderungen konkret und überprüfbar sind. Sie bieten eine Grundlage für Review und Validierung der Architektur und helfen, die Entwicklung in die richtige Richtung zu lenken, um die gewünschten Qualitätseigenschaften zu erreichen. Durch die Verwendung von Qualitätsszenarien wird gewährleistet, dass die Qualitätsanforderungen nicht nur theoretisch festgelegt, sondern auch praktisch überprüfbar und erfüllbar sind.

Ein komplett definiertes Qualitätsszenarium enthält im Idealfall die folgenden 6 Teile:

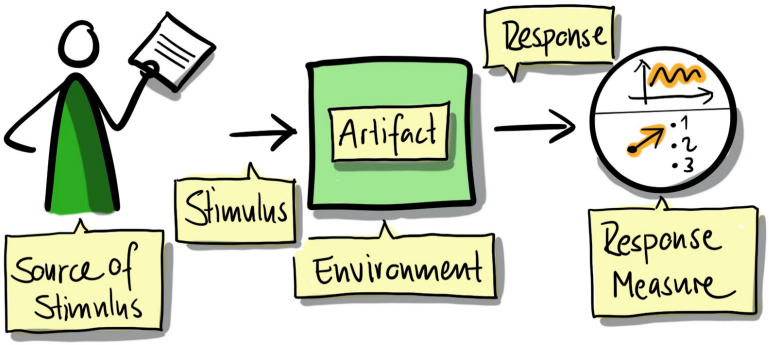


Abbildung 10: Die 6 Teile eines gut definierten Qualitätsszenarium.

1. **Source of Stimulus** → Quelle der Interaktion auf das System.
2. **Stimulus** → Die Interaktion auf das System.
3. **Artifact** → Das Artefakt, das angeregt wurde, d.h. auf welches die Interaktion (der Stimulus) trifft.
4. **Environment** → Die Bedingung, der Zustand in welchem sich das System, der Artifact befindet, wenn das die Interaktion auftritt.
5. **Response** → Das erwartete Resultat.
6. **Response Measure** → Das Mass, das Kriterium, mit dem die Reaktion des Systems bewertet werden kann.

Qualitätsanforderungen spielen eine zentrale Rolle in der Softwareentwicklung – insbesondere in sicherheitskritischen Bereichen wie der Medizintechnik, wo ich in meiner beruflichen Laufbahn viele Jahre tätig war.

Wie bereits in der Einleitung erwähnt, habe ich Software für lebenserhaltende Medizinprodukte entwickelt, bei denen die Qualitätsanforderungen besonders hoch waren. Sie können sich sicher vorstellen, dass in solchen Fällen die Anforderungen an die Ausfallsicherheit und Zuverlässigkeit streng sind. Schliesslich geht es darum, sicherzustellen, dass die Software auch in Ausnahmesituationen zuverlässig funktioniert und die Sicherheit der Patienten gewährleistet bleibt.

In diesem Zusammenhang gibt es Normen, die Massnahmen für die Ausfallsicherheit, Zuverlässigkeit und Robustheit der Software einfordern. Allerdings fordern diese Normen in der Regel nicht was genau zu tun ist. Die entsprechenden Qualitätsanforderungen müssen produktspezifisch klar definiert und messbar gemacht werden, um sie in den Entwicklungsprozess und die Softwarearchitektur einplanen zu können. Auch hier haben sich Qualitätsszenarien als unverzichtbares Werkzeug erwiesen, da sie eine eindeutige und überprüfbare Definition bieten, um sicherzustellen, dass die festgelegten Qualitätsanforderungen auch tatsächlich erfüllt werden.

Ein konkretes Beispiel aus meiner Arbeit im Bereich der „Robustheit“ veranschaulicht dies: Ein Qualitätsszenario für die Software eines Medizingeräts sah vor, dass ungewöhnliche Ereignisraten von Sensor-Schnittstellen (wie z.B. I2C, SPI, Bluetooth, RS232 etc.) innerhalb von weniger als 2000 Millisekunden erkannt werden mussten. Die Software musste in der Lage sein, eine angemessene Reaktion auf solche Anomalien zu zeigen. Das entsprechende Szenario wurde also so definiert:

- **Stimulus:** Ein ungewöhnliches Ereignis tritt auf, z.B. eine abnormale Datenrate eines Sensors.
- **Antwort:** Das System erkennt das ungewöhnliche Verhalten innerhalb von 2 Sekunden.
- **Messbare Kriterien:** Die Software reagiert durch eine Sicherheitsmassnahme sowie eine Protokollierung und stellt sicher, dass keine falschen Daten weiterverarbeitet werden.

Solche Qualitätsszenarien sind in sicherheitskritischen Systemen unerlässlich, um sicherzustellen, dass die Software korrekt reagiert und keine Risiken für Patienten oder Anwender entstehen.

Es muss nicht immer gleich ein lebenserhaltendes Medizingerät sein, aber die präzise Definition und Überprüfung von Qualitätsanforderungen sind in jedem Softwareprojekt von entscheidender Bedeutung.

RAHMENBEDINGUNGEN

Der dritte essenzielle Input für die Architekturgestaltung sind die Rahmenbedingungen (Constraints). Diese definieren den Lösungsraum, in dem sich die Architektur bewegen kann. Beispielsweise können technische Rahmenbedingungen die Wahl bestimmter Technologien vorschreiben oder ausschliessen, während andere Rahmenbedingungen die Integration mit bestehenden Systemen erfordern können. Rechtliche Rahmenbedingungen könnten sich aus Datenschutzbestimmungen ergeben, die bestimmte Sicherheitsmassnahmen erzwingen.

Der Einfluss dieser Rahmenbedingungen auf die Architektur ist bedeutend, da sie oft Grenzen setzen, die kreative Lösungen erfordern können, um die funktionalen und qualitativen Anforderungen innerhalb des vorgegebenen Rahmens zu erfüllen.

In meinen Beratungen bezeichne ich Rahmenbedingungen oft als unsere Freunde, denn sie sind bereits entschiedene Lösungsaspekte, über die nicht mehr diskutiert werden muss. Dadurch vereinfacht sich die Lösungsfindung erheblich und ermöglicht es, den Fokus auf die wirklich entscheidenden Gestaltungsfragen zu legen.

Stellen Sie sicher, dass Sie wissen, welche Inputs für die Architektur Ihres Systems verwendet wurden. Hinterfragen Sie die Qualitätsanforderungen und stellen Sie sicher, dass diese klar und messbar resp. überprüfbar sind. Überprüfen Sie die Rahmenbedingungen kritisch und vergewissern Sie sich, dass es sich dabei nicht um Wünsche, sondern um nicht zu diskutierende Bedingungen handelt.

Um die im Kapitel beschriebenen theoretischen Konzepte in die Praxis zu überführen, habe ich das weiter oben erwähnte Gedankenexperiment mit der einfachen Rechenanwendung aufgegriffen. Diese Anwendung verwende ich als durchgängiges Beispiel, um die Definition der essenziellen Anforderungen bei der Erstellung einer Softwarearchitektur sowie dann auch die weiteren Schritte zu veranschaulichen.

Ich beginne immer mit den funktionalen Anforderungen. Diese müssen so weit wie möglich bekannt sein und identifizierbar vorliegen. In den meisten Projekten sind diese bereits definiert und dokumentiert und müssen in der Architekturerstellung nur noch referenziert werden. In unserem Fall habe ich die funktionalen Anforderungen in einer einfachen Tabelle aufgelistet:

#	Anforderung	Beschreibung
FR1	Eingabemöglichkeit für zwei Zahlen	Bereitstellung von zwei Textfeldern zur Eingabe von Zahlen, die nur numerische Eingaben akzeptieren
FR2	Button zur Addition	Bereitstellung eines "Addieren"-Buttons, der die Addition der Zahlen aus den Textfeldern auslöst.
FR3	Ausgabe des Ergebnisses	Bereitstellung eines Textfelds oder Labels zur Anzeige des Additionsresultats.
FR4	Fehlerbehandlung bei ungültigen Eingaben	Anzeige einer Fehlermeldung beim Versuch ungültigen/nicht numerische Eingaben in die Textfelder einzugeben.
FR5	Benutzerfreundliche Oberfläche	Intuitive, leicht verständliche und logisch angeordnete Benutzeroberfläche.
FR6	Korrekte Berechnung und Anzeige	Sicherstellung der korrekten Addition der eingegebenen Zahlen und leserliche Darstellung des Ergebnisses.

Dann müssen die Qualitätsanforderungen definiert werden. Dazu habe ich einen sogenannten Quality-Attribute-Tree erstellt, in welchem die Qualitätsszenari-

en in die Struktur der Qualitäts-Attribute gemäss ISO/IEC 25010 (siehe Hintergrund-Box) einfügt sind. Für die Ermittlung der Qualitätsanforderungen sollten Sie möglichst umfassend die relevanten Stakeholder wie z.B. Produktmanager, Product-Owner oder Schlüsselkunden miteinbeziehen.

Hier exemplarisch einige der Qualitätsszenarien, welche für das Rechenprogramm berücksichtigt werden sollten (Zugriff auf alle Qualitätsszenarien über untenstehenden Link):

Reliability:

QAR1	Sinnvoller Umgang mit Fehlern, die während einer Berechnung auftreten könnten. ▪ **Source of Stimulus**: System-interner Fehler ▪ **Stimulus**: Ein Fehler tritt während der Berechnung auf, zum Beispiel durch einen Hardwarefehler oder ein Software-Fehler. ▪ **Environment**: Das System befindet sich im normalen Betrieb. ▪ **Artifact**: Gesamtsystem ▪ **Response**: Das System erkennt den Fehler, führt eine Fehlerbehandlung durch und stellt sicher, dass der Benutzer informiert wird und die Systemintegrität gewahrt bleibt. ▪ **Response Measure**: o Der Fehler wird innerhalb von 2 Sekunden erkannt. o Das System protokolliert den Fehler und benachrichtigt den Benutzer mit einer verständlichen Fehlermeldung, ohne dass das System abstürzt.

Performance:

QAP1	Eine Addition muss innerhalb von 1ms durchgeführt werden können. ▪ **Source of Stimulus**: Benutzer ▪ **Stimulus**: Der Benutzer gibt zwei Zahlen zur Addition ein. ▪ **Environment**: Das System ist im normalen Betriebsmodus. ▪ **Artifact**: Die Rechenkomponente des Systems. ▪ **Response**: Das System berechnet die Summe der beiden Zahlen. ▪ **Response Measure**: Die Berechnung erfolgt innerhalb von 1 Millisekunde.

Operability:

QAO1	Einfache intuitive Benutzeroberfläche. ▪ **Source of Stimulus**: Benutzer ▪ **Stimulus**: Der Benutzer möchte die zwei Zahlen zur Addition eingeben. ▪ **Environment**: Das System ist betriebsbereit und wird auf einem Desktop-Computer verwendet. ▪ **Artifact**: Gesamtsystem. ▪ **Response**: Das System zeigt eine benutzerfreundliche Oberfläche zur Eingabe der Zahlen und zur Anzeige des Ergebnisses an. ▪ **Response Measure**: Der Benutzer kann die Aufgabe in weniger als 10 Sekunden ohne Fehler abschließen.

Maintainability:

QAM1	Integration neuer Rechenoperationen durch Neustart der Anwendung ▪ **Source of Stimulus**: Entwickler ▪ **Stimulus**: Der Entwickler möchte eine neue Rechenoperation (z.B. Division) zur Software hinzufügen (Einschränkung: Rechenoperation basierend auf 2 Zahlen). ▪ **Environment**: Laufzeit ▪ **Artifact**: Software. ▪ **Response**: Die neue Rechenoperation wird erfolgreich integriert und kann nach dem Neustart der Software sofort genutzt werden. ▪ **Response Measure**: Die Integration dauert maximal 5 Minuten. Die neue Operation steht sofort nach dem Neustart der Software zur Verfügung – ohne dass die Software neu gebaut werden muss.

...

Transferability:

QAT1	Nutzung von Rechenoperationen ohne User Interface. ▪ **Source of Stimulus**: Entwickler ▪ **Stimulus**: Der Entwickler möchte die Logik der Rechenoperationen als API nutzen. ▪ **Environment**: Entwicklungszeit ▪ **Artifact**: Der Quellcode und das API der Software. ▪ **Response**: Die Rechenoperation(en) sind über ein API verfügbar, sodass sie direkt von anderen Systemen oder Skripten aufgerufen werden können. ▪ **Response Measure**: Die API ist vollständig dokumentiert und getestet und kann ohne die Verwendung des User Interfaces von anderen Systemen oder Skripten verwendet werden.

Schliesslich müssen die Rahmenbedingungen definiert werden, da diese den Lösungsraum für die Architektur abstecken. Wie bereits weiter oben erwähnt, ist es essenziell, die Rahmenbedingungen explizit zu machen. Dazu liste ich die Rahmenbedingungen üblicherweise tabellarisch auf:

#	Rahmenbedingung	Beschreibung
C1	Windows Anwendung	Die Software muss als Windows-Anwendung ausgeführt werden können.
C2	.NET Umsetzung	Die Software muss unter Verwendung des .NET Frameworks oder .NET Core/.NET 5+ entwickelt werden.
C3	C# Umsetzung	Die Software muss in der Programmiersprache C# umgesetzt werden.
C4	Windows Forms	Die grafische Benutzeroberfläche der Software muss mit Windows Forms (WinForms) realisiert werden.

Damit sind die essenziellen Anforderungen vollständig definiert. Diese werden uns im weiteren Verlauf als Basis dienen, um die nächsten Schritte der Architekturentwicklung für unser Rechenprogramm durchzuführen.

Um einen weiteren wichtigen Aspekt zu veranschaulichen, möchte ich eine Geschichte aus meiner Praxis erzählen, als ich bei einem Kunden einen Qualitäts-Attribute-Workshop durchgeführt habe. Solche Workshops haben das Ziel, die Qualitätsanforderungen eines Systems zu identifizieren und in messbare Qualitätsszenarien zu überführen. Der Workshop bringt verschiedene Stakeholder zusammen, um ihre Perspektiven und Anforderungen zu diskutieren und zu dokumentieren.

Bei diesem speziellen Workshop war ein wichtiger Stakeholder anwesend, der schon sehr lange im Unternehmen tätig war und als Entwicklungsleiter eine zentrale Position innehatte. Während wir durch die verschiedenen Funktionen des Systems gingen und ich die Teilnehmer in Bezug auf die Qualitätsanforderungen befragte, brachte der Entwicklungsleiter wiederholt Aussagen wie: „Das ist doch selbstverständlich. Es ist doch klar, dass dieses Qualitätsniveau hier gefordert ist."

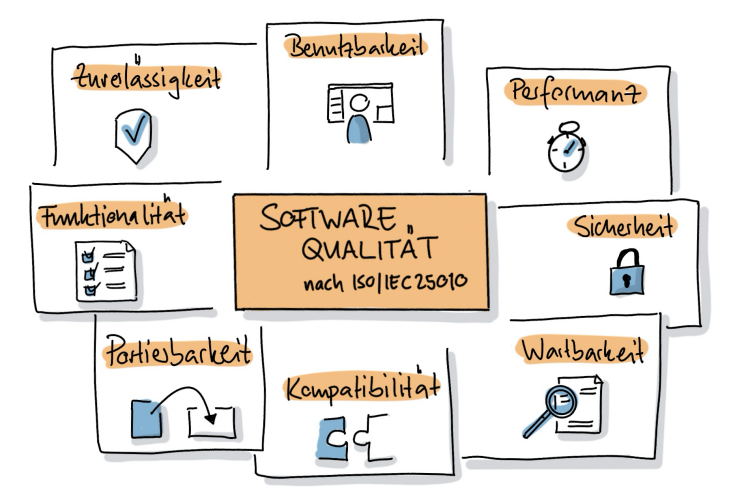
Diese Haltung verdeutlicht ein weit verbreitetes Problem: Viele Qualitätsanforderungen werden implizit angenommen und nicht explizit kommuniziert. Was für den erfahrenen Entwicklungsleiter klar und selbstverständlich erscheint, ist für andere Stakeholder möglicherweise nicht so offensichtlich. Die fehlende Dokumentation und nicht explizite Festlegung dieser Anforderungen können zu Missverständnissen und Fehlinterpretationen führen, was letztendlich die Qualität des Endprodukts beeinträchtigt.

Durch den Workshop konnten wir diese impliziten Annahmen sichtbar machen und in klare, messbare Qualitätsszenarien überführen.

ISO/IEC 25010

Die ISO/IEC 25010 ist eine internationale Norm, die ein Modell für die Bewertung der Qualität von Softwareprodukten bereitstellt. Sie wurde von der International Organization for Standardization (ISO) und der International Electrotechnical Commission (IEC) entwickelt und ist Teil der umfassenden ISO/IEC 25000-Familie, die auch als SQuaRE (Software Product Quality Requirements and Evaluation) bekannt ist.

Die Norm definiert acht Qualitätsmerkmale, die als Rahmen zur Bewertung der Qualität von Softwareprodukten dienen:



- Funktionalität: Die Fähigkeit der Software, spezifizierte Aufgaben zu erfüllen.
- Performanz: Das Verhältnis zwischen der Leistungsfähigkeit des Systems und den genutzten Ressourcen.

- Kompatibilität: Die Fähigkeit der Software, in einer gemeinsamen Umgebung mit anderen Produkten zu funktionieren.
- Benutzbarkeit: Wie leicht und effizient die Software von bestimmten Benutzern in einem bestimmten Kontext genutzt werden kann.
- Zuverlässigkeit: Die Fähigkeit der Software, ihre Leistung unter bestimmten Bedingungen über eine bestimmte Zeitspanne hinweg aufrechtzuerhalten.
- Sicherheit: Der Schutz der Informationen und Daten, die das System verarbeitet.
- Wartbarkeit: Die Fähigkeit der Software, effizient geändert zu werden.
- Protierbarkeit: Die Fähigkeit der Software, von einer Umgebung in eine andere übertragen zu werden.

Die ISO/IEC 25010 ist besonders relevant für **Qualitäts-Attribute-Workshops**, da sie einen strukturierten Rahmen für die Identifizierung und Bewertung von Qualitätsanforderungen bietet. Während eines solchen Workshops wird der Fokus auf die Definition und Priorisierung der Qualitätsmerkmale gelegt, die für das jeweilige Projekt am wichtigsten sind. Die Norm bietet eine umfassende und anerkannte Grundlage, auf der die Teilnehmer ihre Anforderungen diskutieren und formulieren können.

Ein wesentlicher Bestandteil der Arbeit in Qualitäts-Attribute-Workshops ist die Erstellung von Qualitätsszenarien. Diese Szenarien machen die Qualitätsanforderungen messbar und überprüfbar, in-

dem sie spezifische Situationen beschreiben, in denen die Qualität des Systems getestet werden kann. Die acht Qualitätsmerkmale der ISO/IEC 25010 bieten dabei eine wertvolle Orientierungshilfe. Jedes Szenario kann einem oder mehreren dieser Merkmale zugeordnet werden, wodurch sichergestellt wird, dass alle relevanten Aspekte der Softwarequalität berücksichtigt werden.

Durch die Verwendung der ISO/IEC 25010 kann sichergestellt werden, dass die Qualitätsanforderungen umfassend definiert wurden. Dies fördert nicht nur die Klarheit und Präzision in der Planung, sondern erhöht auch die Wahrscheinlichkeit, dass die entwickelten Systeme den langfristigen Anforderungen und Erwartungen entsprechen.