THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Seminar

Instabilität und Ungewissheit nicht-funktionaler Anforderungen

Eine Untersuchung von Herausforderungen und agilen Lösungsansätzen

Maria Kazim

24. Mai 2024

Zusammenfassung

Diese Arbeit untersucht die Bedeutung nicht-funktionaler Anforderungen (NFRs) in der Softwareentwicklung und die Herausforderungen ihrer Instabilität und Unsicherheit. NFRs beschreiben Qualitätsmerkmale wie Leistung, Sicherheit und Zuverlässigkeit, die oft vage und dynamisch sind. Empirische Daten zeigen, dass traditionelle Ansätze oft unzureichend sind. Agile Methoden wie Extreme Programming (XP) und das Scaled Agile Framework (SAFe) bieten flexible und iterative Lösungen zur Bewältigung dieser Herausforderungen. Sie fördern die frühzeitige Identifikation, kontinuierliche Überprüfung und Anpassung von NFRs durch Feedbackschleifen und Stakeholder-Einbindung. Die Arbeit zeigt, dass agile Methoden die Softwarequalität verbessern und NFRs effizienter verwalten.

Inhaltsverzeichnis

1 Einführung

"Non-functional requirements" not sounding cool isn't a reason to neglect them." - Simon Brown, *Software Architecture for Developers: Volume 1 - Technical leadership and the balance with agility* [3]

Mit diesem Zitat erkannte Simon Brown die oft unterschätzte Bedeutung nicht-funktionaler Anforderungen (im Folgenden mit NFR abgekürzt) in der modernen Softwareentwicklung und warnte ausdrücklich davor, sie aufgrund ihrer scheinbaren Unattraktivität zu vernachlässigen. Obwohl NFRs oft weniger Beachtung finden als funktionale Anforderungen (im Folgenden mit FR abgekürzt) [Abschn.1][12], sind sie für den langfristigen Erfolg von Software unerlässlich. Während FRs klar definieren, *was* ein System tun soll, beschreiben NFRs die qualitativen Aspekte, genauer *wie* das System diese Funktionen erfüllen soll [Abschn.2][12]. Diese Qualitätsfaktoren umfassen jedoch oft vage und schwer quantifizierbare Eigenschaften wie Leistung, Sicherheit und Zuverlässigkeit, die sich im Laufe der Entwicklung ändern oder konkretisieren können. Die daraus resultierende Notwendigkeit einer nachträglichen Anpassung und Optimierung von Software ist schließlich der Grund, warum NFRs zugleich instabil und unsicher sind. Diese Unsicherheit und Instabilität stellen Softwareentwicklungsprozesse vor bedeutende Herausforderungen. Um diesen Herausforderungen zu begegnen, ist es unerlässlich, den aktuellen Umgang mit NFRs und bisherige Versäumnisse genauer zu betrachten, um darauf aufbauend geeignete Maßnahmen zu ergreifen. In diesem Zusammenhang gewinnen insbesondere agile Methoden zunehmend an Bedeutung. Agile Methoden versprechen eine flexible und iterative Herangehensweise, die geeignet sein könnte, diese Herausforderungen - Instabilität und Unsicherheit - im Softwareentwicklungsprozess zu adressieren. Das Ziel dieser Ausarbeitung ist daher, die Herausforderungen, die im Umgang mit NFRs verbunden sind, genauer zu untersuchen und wie agile Methoden dazu beitragen können, der Instabilität und Unsicherheit im Umgang mit NFRs zu begegnen. Zu diesem Zweck wird eine Analyse der empirischen Daten von Viviani et al. durchgeführt, um zugleich Einblicke in die Praxis der Verwaltung von NFRs zu gewinnen und zu untersuchen. Der Aufbau der Seminararbeit gliedert sich wie folgt: Nach dieser Einleitung werden im zweiten Kapitel die Grundlagen von NFRs erläutert, auch in Abgrenzung zu FRs, gefolgt von einer Untersuchung der Herausforderungen im Umgang mit NFRs im dritten Kapitel. Kapitel vier erläutert die Studie von Viviani et al. Anschließend werden die Ergebnisse dieser Studie präsentiert und analysiert, um daraus Handlungsempfehlungen abzuleiten. Im fünften Kapitel werden agile Methoden und ihre Eignung zur Adressierung der Herausforderungen von NFRs betrachtet. Abschließend werden in Kapitel sechs die wesentlichen Erkenntnisse zusammengefasst.

2 Definition nicht-funktionaler Anforderungen in Abgrenzung zu funktionalen Anforderungen

Für die erfolgreiche Umsetzung von Softwarelösungen ist das Verständnis sowohl der NFRs als auch FRs unerlässlich [Abschn.1][12]. In diesem Abschnitt werden daher die

theoretischen Grundlagen für NFRs dargelegt, darunter ihre Definition und die Abgrenzung zu FRs. Diese Erkenntnisse bilden anschließend die Basis für die Betrachtung von NFRs im weiteren Verlauf dieser Arbeit.

2.1 Definition und Eigenschaften nicht-funktionaler Anforderungen

NFRs beschreiben, „wie ein System seine Funktionalität ausführt“ [Abschn.2][12], um die Anforderungen der Nutzer in Bezug auf Leistung, Sicherheit, Zuverlässigkeit und Benutzerfreundlichkeit zu erfüllen [Abschn.2][5]. Hierzu zeichnen sich NFRs durch mehrere charakteristische Eigenschaften aus. Erstens sind NFRs häufig weniger konkret und schwerer messbar [Abschn.1][5]. Diese Eigenschaft erschwert ihre Validierung, da es oft schwierig ist, klare und präzise Kriterien für ihre Erfüllung zu definieren. Die Anforderung an die Systemleistung, wie die schnelle Reaktionsfähigkeit ist vage und es ist unklar, wie *schnell* genau definiert und gemessen werden soll. Zweitens sind NFRs oft dynamisch und unterliegen regelmäßigen Veränderungen im Laufe der Zeit. Diese Veränderungen resultieren aus den sich ändernden Anforderungen der Nutzer und den technologischen Fortschritten [Abschn.1][12]. Die Skalierbarkeit einer Anwendung ist ein geeignetes Beispiel hierfür. Zu Beginn eines Projekts kann eine bestimmte Benutzeranzahl als ausreichend betrachtet werden. Mit der Zeit und wachsendem Erfolg des Produkts kann jedoch die Notwendigkeit entstehen, die Anwendung zu skalieren, um eine höhere Anzahl gleichzeitiger Benutzer zu unterstützen. Nicht zuletzt sind NFRs oft weniger eindeutig dokumentiert und erfordern zusätzliche Erläuterungen und Spezifikationen. Sie werden häufig in Form von Qualitätsattributen festgehalten [Abschn.2][5]. Diese Anforderungen müssen oft durch technische Kriterien spezifiziert werden, um ihre Erfüllung messbar zu machen [Abschn.2][12]. So kann eine Anforderung an die Sicherheit eines Systems, wie etwa die Sicherheit sensibler Daten, durch die Spezifikation technischer Kriterien, wie den Einsatz von konkreten Verschlüsselungsalgorithmen, spezifiziert und messbar gemacht werden.

2.2 Abgrenzung von funktionalen und nicht-funktionalen Anforderungen

Ein ganzheitliches Verständnis von NFRs gelingt letztlich jedoch nur über eine Abgrenzung zu den FRs. FRs definieren die spezifischen Aufgaben, die ein Softwaresystem ausführen soll [Abschn.2][12]. Im Gegensatz zu NFRs zeichnen sich FRs durch Konkretheit und Messbarkeit aus. Sie sind spezifisch, oft leicht testbar und beziehen sich direkt auf die vom System zu erbringenden Leistungen [Abschn.2][12]. Sie dienen als direkte Anweisungen für die Entwicklung der Systemfunktionen und bilden zugleich die Basis für die Definition der Systemleistung [Abschn.2][11]. Eine FR in einem Online-Banking-System könnte lauten, dass Benutzer Transaktionen durchführen können. Diese Anforderung ist konkret und messbar, da sie genau beschreibt, welche Funktion das System bieten muss. Die Umsetzung kann getestet werden, indem überprüft wird, ob Benutzer erfolgreich Geld von einem Konto auf ein anderes überweisen können und ob die Transaktion korrekt verbucht wird. Gleichzeitig bilden FRs das stabile Fundament des Gesamtsystems. Sie werden in der Regel zu Beginn eines Projekts klar definiert und ändern sich im Ver-

lauf des Projekts kaum [Abschn.1][12]. So bleibt die Anforderung, dass ein Benutzer in einem E-Mail-System Nachrichten versenden und empfangen kann, normalerweise während des gesamten Projekts unverändert. Darüber hinaus erfolgt die Dokumentation von FRs häufig in Form von detaillierten Anforderungsspezifikationen, die klar definieren, was das System tun soll [Abschn.2][5]. Diese Dokumentationen sind leicht verständlich und können direkt in Testfälle übersetzt werden [Abschn.2][5]. Die Anforderung, dass ein Benutzer in einem E-Commerce-System eine Bestellung abschließen kann, stellt eine klare Beschreibung dar, welche Schritte der Benutzer durchlaufen muss und wie das System darauf reagieren soll. Während FRs also den unmittelbaren Nutzen und die Funktionsfähigkeit des Systems sicherstellen, gewährleisten NFRs langfristige Qualität, Sicherheit und Benutzerakzeptanz.

3 Herausforderungen im Umgang mit nicht-funktionalen Anforderungen

Ein fundiertes Verständnis für NFRs und ihre Abgrenzung zu FRs bildet die Grundlage für die erfolgreiche Bewältigung der damit verbundenen Herausforderungen. Die oben erörterten Eigenschaften von NFRs – nämlich ihre Unkonkretheit, Dynamik und die mangelnde Dokumentation – führen dazu, dass NFRs oft instabil und unsicher sind. Nur durch eine gezielte Betrachtung und ein tiefgehendes Verständnis dieser Herausforderungen, ihrer Ursache und Auswirkungen können Strategien entwickelt werden, die es ermöglichen, diese Herausforderungen zu adressieren.

3.1 Instabilität und Unsicherheit

Die Instabilität ist auf die Dynamik von NFRs zurückzuführen. Die Ursachen für die Instabilität sind daher häufige Änderungen und Anpassungen im Laufe eines Projekts. Diese Änderungen und Anpassungen sind wiederum das Ergebnis sich ändernder Rahmenbedingungen, neuer Erkenntnisse oder veränderter Prioritäten [Abschn.3.2][12]. Da NFRs sich überwiegend durch qualitative Merkmale auszeichnen und von unterschiedlichen Menschen unterschiedlich interpretiert werden können, besteht eine größere Wahrscheinlichkeit, dass die Anforderungen im Verlauf des Entwicklungsprozesses überarbeitet werden müssen. Neue gesetzliche Vorgaben für Datenschutz und Sicherheit können ebenfalls eine Anpassung bestehender Sicherheitsanforderungen erfordern. Ebenso kann die Verfügbarkeit neuer Technologien, wie etwa verbesserte Verschlüsselungsmethoden, eine Anpassung bestehender Sicherheitsanforderungen notwendig machen. Die Unsicherheit hingegen ist auf die Unkonkretheit von NFRs zurückzuführen. Unsicherheit beschreibt die Schwierigkeit, dass NFRs aufgrund ihrer ungenauen und oft vage formulierten Art nicht frühzeitig und vollumfänglich bestimmt werden können [Abschn.5.2][12]. Diese Schwierigkeit entsteht durch die Herausforderung, dass zukünftige Nutzungsszenarien und technische Rahmenbedingungen nicht vorhersagbar sind. NFRs wie Sicherheit, Benutzerfreundlichkeit oder Performance lassen immer Spielraum für Interpretation [Abschn.1][5]. Diese Unsicherheit kann zu Missverständnissen und Fehlkommunikation

zwischen den Stakeholdern führen und infolgedessen die Entwicklung erschweren. Verstärkt wird dieses Problem dadurch, dass NFRs aktuell erst dann vollständig identifiziert und spezifiziert werden, wenn das System bereits im Einsatz ist oder wenn sich externe Bedingungen ändern [Abschn.2][12].

3.2 Folgen von Instabilität und Unsicherheit

Die Folgen von Instabilität und Unsicherheit im Umgang mit NFRs sind vielfältig und wirken sich auf alle Phasen der Softwareentwicklung aus. Sie erschweren die *Planung*, *Umsetzung* und finale *Auslieferung* und stellen somit einen entscheidenden Faktor für den Projekterfolg dar.

3.2.1 Folgen der Instabilität

In der *Planungsphase* führt Instabilität dazu, dass präzise Anforderungen schwer zu definieren und klare Ziele schwer zu setzen sind. Änderungen an NFRs können die initiale Planung überflüssig machen und erfordern ständige Anpassungen. Diese Umstände erschweren die Ressourcenallokation und die Erstellung realistischer Zeitpläne [Abschn.5.1][12]. So könnte die geplante Leistung eines Systems angepasst werden müssen, wenn neue gesetzliche Vorgaben für Datenschutz und Sicherheit eingeführt werden. Infolgedessen muss das Team möglicherweise die Sicherheitsprotokolle des Systems überarbeiten, um den erhöhten Anforderungen an die Datensicherheit gerecht zu werden. In der *Implementierungsphase* können instabile NFRs umfangreiche Änderungen erfordern, die teure und zeitraubende Überarbeitungen zur Folge haben. Anpassungen können den Entwicklungsprozess verlangsamen und das Risiko erhöhen, sodass Termine nicht eingehalten werden [Abschn.2][12]. Die Einführung eines neuen Sicherheitsfeatures oder die Verbesserung der Performance kann beispielsweise unerwartete Änderungen erfordern, wie etwa das Umschreiben von Code, die Aktualisierung von Schnittstellen oder die Integration zusätzlicher Softwarekomponenten. In der *Ausführungsphase* kann die Instabilität von NFRs zu einer Erhöhung der Fehlerraten im fertigen Produkt führen [Abschn.5.2][12]. Wiederholte Änderungen der Anforderungen hinsichtlich der Benutzeroberfläche resultieren in einer Vielzahl an Anpassungen, die möglicherweise eine inkonsistente Benutzererfahrung zur Folge haben [Abschn.2][12]. Dies wiederum erhöht das Risiko von Fehlern und Unstimmigkeiten in der Software.

3.2.2 Folgen der Unsicherheit

In der *Planungsphase* führt die Unsicherheit dazu, dass NFRs oft nur vage oder unvollständig formuliert werden können. Dies erschwert die Festlegung konkreter Projektziele und die Erstellung detaillierter Projektpläne. Die unklare Definition von Anforderungen kann zu Missverständnissen zwischen den Stakeholdern führen und die Kommunikation erschweren. Dadurch werden Risiken und potenzielle Probleme möglicherweise nicht frühzeitig erkannt und adressiert [Abschn.5.2][12]. Unsicherheit in der *Implementierungsphase* führt dazu, dass Entwickler häufig mit unklaren Anforderungen arbeiten müssen. Dies kann zu Fehlentwicklungen und ineffizienten Lösungsansätzen führen, da

Entwickler möglicherweise verschiedene Interpretationen einer Anforderung verfolgen. Diese Unsicherheit kann den Entwicklungsprozess verlangsamen und die Qualität der Software negativ beeinflussen, da wichtige Details erst spät im Prozess geklärt werden [Abschn.2][12]. Die Unsicherheit in der *Ausführungsphase* kann zu einer inkonsistenten Implementierung der NFRs führen, da unklare Anforderungen zu unterschiedlich interpretierten und umgesetzten Features führen. Dies kann die Qualität und Zuverlässigkeit des Endprodukts beeinträchtigen. Unsicherheit bei der Umsetzung von NFRs wie Sicherheit und Leistung kann zu Sicherheitslücken oder unzureichender Performance führen, was schließlich das Vertrauen der Endnutzer in das Produkt mindert.

4 Empirische Untersuchung zum Umgang mit nicht-funktionalen Anforderungen in der Praxis

Die vorherigen Abschnitte verdeutlichen die unbestreitbare Relevanz und Komplexität von NFRs. Jedoch blieb bisher der tatsächliche Umgang mit den Herausforderungen von NFRs in der Praxis häufig hinter den theoretischen Erkenntnissen zurück. Dabei sind diese Herausforderungen weit mehr als theoretische Überlegungen; sie beeinflussen unmittelbar die tägliche Arbeit aller Personen, die sich mit der Entwicklung von Software befassen. Um diese Diskrepanz zwischen Theorie und Praxis zu überbrücken, haben Viviani et al. eine umfangreiche Studie durchgeführt, die darauf abzielt, ein tieferes Verständnis für den Umgang mit NFRs in der Praxis zu entwickeln. Im Nachfolgenden wird diese Studie daher genauer untersucht, einschließlich ihrer Motivation, Ziele und Ergebnisse.

4.1 Kontextuelle Einordnung der Studie

Zuvor wird jedoch eine kontextuelle Einordnung der Studie vorgenommen, um ein umfassendes Verständnis der Ausgangslage und beteiligten Akteure zu gewährleisten. Unter den 40 Fachleuten, die an der Umfrage teilgenommen haben, arbeiteten „mehr als 50% in den letzten fünf Jahren als Softwarearchitekten oder Softwareingenieure“ [Abschn.4.1][12], und „fast 40% als Entwickler“ [Abschn.4.1][12]. Diese Gruppen sind maßgeblich an der Definition und Implementierung von und dem Umgang mit NFRs in Softwareprojekten beteiligt [Abschn.4.1][12].
Ihre Erfahrungen und Einsichten sind daher entscheidend, um zu verstehen, wie NFRs in der Praxis gehandhabt werden. Die Verteilung der Teilnehmer nach Berufserfahrung und ihrer Selbsteinschätzung bezüglich ihrer Expertise im Umgang mit NFRs in Projekten wird zusätzlich in *Abbildung 1* veranschaulicht. Die Grafik verdeutlicht, dass die meisten Antworten von erfahrenen Fachleuten stammen, was die Tiefe und Breite des Wissens in dieser Studie widerspiegelt. Diese Expertise beeinflusst direkt die Qualität und Zuverlässigkeit der Erkenntnisse, insbesondere in Bezug auf die Bewertung und Anpassung von NFRs im Lebenszyklus der Softwareentwicklung.

NFR Experience	1 to 3 years	4 to 6 years	7 to 9 years	10 to 12 years	13 to 15 years	15 years or more
Novice	0	1	0	0	0	0
Advanced Beginner	0	0	0	0	0	1
Intermediate	1	2	1	5	1	6
Proficient	0	0	2	1	4	6
Specialist	0	0	1	1	0	7

Abbildung 1: A Number of participants by years of experience and self-evaluated knowledge in handling NFRs [Abschn.4.1][12]

4.2 Motivation und Ziele der Studie

Die Autoren erkannten die Herausforderungen, die mit NFRs in der Softwareentwicklung verbunden sind. Ziel der Studie war daher, ein besseres Verständnis für die Instabilität und Unsicherheit von NFRs zu erlangen und aufzuzeigen, wie diese in realen Projekten verwaltet werden [Abschn.1][12]. Durch die Sammlung empirischer Daten von erfahrenen Fachleuten sollte die Studie schließlich Lücken in der bestehenden Literatur hinsichtlich der praktischen Herausforderungen und der Verwaltung von NFRs in der Praxis schließen [Abschn.1][12]. Ziel war auch aufzuzeigen, dass gegenwärtige Ansätze oft nicht ausreichen, um die Herausforderungen im Umgang mit NFRs zu bewältigen. Aus den gewonnenen Erkenntnissen werden anschließend konkrete Handlungsempfehlungen abgeleitet, die deutlich machen, dass diesen Herausforderungen nur durch den Einsatz agiler und flexibler Methoden begegnet werden kann [Abschn.6][12].

4.3 Gegenwärtiger Umgang mit nicht-funktionalen Anforderungen in der Praxis

Die Ergebnisse der Studie gewähren einen ersten Eindruck von den Strategien zur Bewältigung der Herausforderungen im Umgang mit NFRs. Diese Strategien beinhalten häufig eine Mischung aus reaktiven und proaktiven Maßnahmen. Einerseits werden oft reaktive Ansätze verfolgt, bei denen auf Änderungen reagiert wird, sobald sie evident werden [Abschn.5.2][12]. Diese Maßnahmen sind notwendig, um auf neue Anforderungen wie Sicherheitsstandards oder Leistungsoptimierungen zu reagieren und kurzfristige Anpassungen an Systemkomponenten vorzunehmen. Gleichwohl wird zunehmend die Notwendigkeit erkannt, proaktive Strategien zu entwickeln, um potenzielle Änderungen von NFRs frühzeitig zu identifizieren und gezielt darauf zu reagieren [Abschn.4.2][12]. Die regelmäßige Überprüfung sowie die technische Evaluation von NFRs stellen hierbei zentrale Maßnahmen dar. Diese Evaluierungen spielen eine entscheidende Rolle, da sie als wesentliche Treiber für potentielle Anpassungen dienen, wie in *Abbildung 2* gezeigt wird. Dort gaben 28% der Befragten an, dass technische Evaluationen maßgeblich für Änderungen verantwortlich sind, während 24% regelmäßige Überprüfungen als entscheidend erachteten [Abschn.4.2][12].
Die dargelegten Ansätze im Umgang mit NFRs stellen einen ersten Versuch dar, um den

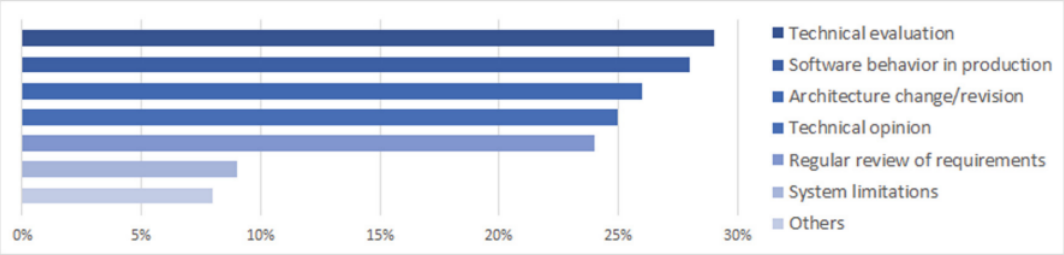

Abbildung 2: Driver of changes in NFRs [Abschn.4.2][12]

Herausforderungen von NFRs zu begegnen. Die Dynamik von NFRs verlangt jedoch, dass bestehende Strategien kontinuierlich hinterfragt werden. Ein kontinuierlicher Optimierungsprozess ermöglicht es, flexibel auf Änderungen zu reagieren und gleichzeitig die Qualität und Stabilität von Softwareprojekten zu gewährleisten. Dies fördert nicht nur die Reduktion von Risiken und Kosten [Abschn.2][12], sondern auch die Innovationsfähigkeit und Wettbewerbsstärke in einem sich ständig wandelnden Technologiemarkt.

4.4 Ergebnisse zur Bewältigung der Herausforderungen in der Praxis

Die Ergebnisse der empirischen Studie von Viviani et al. verdeutlichen nicht zuletzt auch eine Vielzahl von Versäumnissen im praktischen Umgang mit NFRs. Diese betreffen die Identifikation, Definition, Anpassung und Validierung von NFRs im Verlauf von Softwareprojekten [Abschn.2][12]. Im Folgenden werden basierend auf den wichtigsten Erkenntnissen der Studie konkrete Maßnahmen abgeleitet, die dazu beitragen sollen, diesen Versäumnissen wirksam entgegenzuwirken.

4.4.1 Frühe und kontinuierliche Identifikation von nicht-funktionalen Anforderungen

Ein signifikanter Anteil von fast einem Drittel der NFRs wird nicht zu Beginn des Projekts identifiziert [Abschn.5.1][12]. Die in *Abbildung 3* dargestellten Daten stützen die Annahme, dass NFRs auch in späteren Phasen des Projekts definiert werden.
Eine späte Identifikation birgt das Risiko, dass wichtige Anforderungen zu Beginn des Prozesses nicht ausreichend berücksichtigt werden und dies in der Folge zu erheblichen Problemen führen kann [Abschn.2][12]. Um dieser Herausforderung zu begegnen, müssen Projektteams frühzeitig Ansätze zur Identifikation von NFRs implementieren. Dazu gehört die Einbindung aller relevanten Stakeholder bereits in den frühen Phasen der Projektplanung. Frühzeitiges Feedback von Stakeholdern und regelmäßige Feedbackschleifen können helfen, NFRs schon zu Beginn umfassend zu identifizieren und kontinuierlich anzupassen [Abschn.5.1][12].

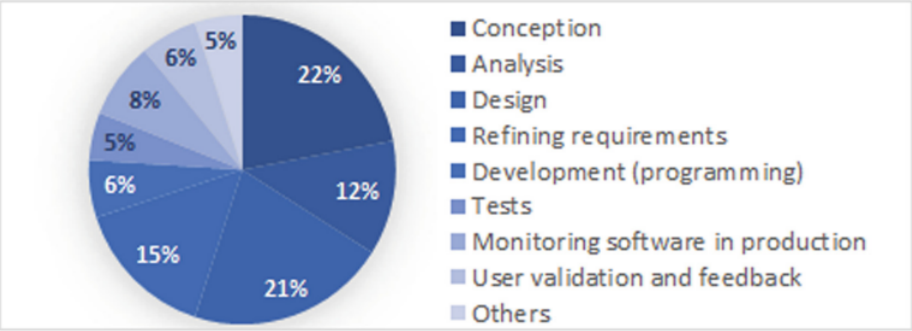

Abbildung 3: Activities in which NFRs were defined [Abschn.4.2][12]

4.4.2 Integration einer umfassenden Teststrategie

Das Fehlen einer Teststrategie erschwert es, Änderungen zu berücksichtigen und zu validieren, die sich potentiell auf NFRs und das Gesamtsystem auswirken [Abschn.5.2][12]. Ohne eine adäquate Teststrategie können unvorhergesehene Änderungen zu ernsthaften Problemen in der Systemstabilität führen [Abschn.5.2][12]. Eine umfassende Teststrategie, die sowohl FRs als auch NFRs abdeckt, erweist sich in der Praxis als entscheidend. Diese Strategie sollte, neben automatisierten Tests auch Sicherheitstests umfassen [Abschn.5.2][12], um sicherzustellen, dass Änderungen von Softwarekomponenten oder im Systemverhalten frühzeitig erkannt und validiert werden können.

4.4.3 Gezielte Ermittlung und Klassifizierung nicht-funktionaler Anforderungen nach Typ

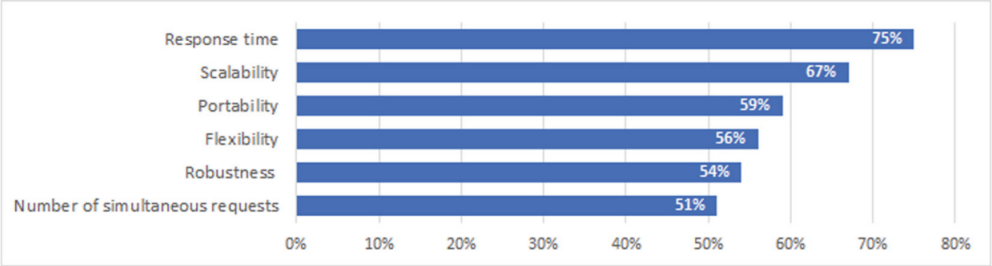

Abbildung 4: Proportion of the projects in which the type of NFR was declared [Abschn.4.2][12]

Die Identifikation von NFRs kann je nach Typ erfolgen [Abschn.5.3][12]. Unterschiedliche NFR-Typen erfordern spezifische Ansätze zur Identifikation und Anpassung. Da verschiedene Arten von NFRs unterschiedliche Ansätze erfordern, sollten die Methoden zur Identifikation und Anpassung der NFRs spezifisch auf den jeweiligen Typ abgestimmt werden. Dies beinhaltet beispielsweise unterschiedliche Techniken für die Ermittlung von

Performance-Anforderungen im Vergleich zu Sicherheitsanforderungen. Zusätzlich können NFRs je nach ihrer Relevanz für das Projekt unterschiedlich priorisiert und mit Ressourcen versehen werden. So bezog sich beispielsweise ein Großteil der ermittelten NFRs (75%) in einer Vielzahl von Projekten auf die Reaktionszeit, wie auch in *Abbildung 4* gezeigt wird. Empirische Daten können bei der Priorisierung helfen, indem sie aufzeigen, welche Anforderungen häufiger ermittelt werden und somit eine höhere Relevanz für viele Projekte haben [Abschn.5.3][12].

4.4.4 Stakeholder-Integration und kontinuierliche Kommunikation

Eine signifikante Anzahl von Änderungen an NFRs tritt nach der Analyse des Verhaltens des Systems in der Produktion auf [Abschn.5.3][12]. Späte Änderungen sind teuer und schwierig in der Umsetzung. Stakeholder spielen daher eine zentrale Rolle bei der Ermittlung von NFRs und sind mit 63% die häufigste Quelle für deren Identifizierung, wie auch in *Abbildung 5* veranschaulicht wird [Abschn.4.2][12]. Durch den regelmäßigen Austausch mit Stakeholdern können potenzielle Anforderungen frühzeitig identifiziert und kontinuierlich angepasst werden, um den sich ändernden Bedürfnissen und Erwartungen gerecht zu werden [Abschn.4.2][12].

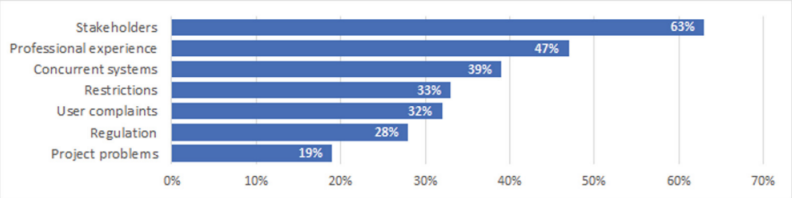

Abbildung 5: NFRs elicitation approach [Abschn.4.2][12]

Insgesamt zeigt die Studie deutlich, dass der Umgang mit NFRs komplex und vielschichtig ist. Die hier abgeleiteten Maßnahmen, wie die frühe und kontinuierliche Identifikation von NFRs, die Integration einer umfassenden Teststrategie, die gezielte Ermittlung und Differenzierung nach Typ sowie die intensive Stakeholder-Integration, bieten wertvolle Ansätze zur Bewältigung der identifizierten Herausforderungen. Der Fokus liegt letztlich darauf, diese Lösungsansätze in die Praxis umzusetzen und anzuwenden. Agile Arbeitsweisen, die durch ihre Flexibilität und iterative Herangehensweise eine kontinuierliche Identifikation und Anpassung von NFRs ermöglichen, können dabei besonders geeignet sein [Abschn.6][12]. Sie fördern eine dynamische und reaktionsfähige Projektumgebung, in der NFRs effektiv verwaltet und in die Entwicklungsprozesse integriert werden können.

5 Agile Lösungsansätze im Umgang mit nicht-funktionalen Anforderungen

Eine frühzeitige Identifikation, regelmäßige Überprüfung sowie kontinuierliche Anpassung von NFRs ermöglicht es Teams, den Herausforderungen im Umgang mit NFRs

flexibel zu begegnen. Agile Lösungsansätze können diesen Prozess unterstützen, indem sie den organisatorischen Rahmen schaffen, der Unsicherheiten und Instabilität gezielt adressiert. Sie ermöglichen Teams, auf Veränderungen schnell zu reagieren und Anforderungen kontinuierlich zu evaluieren und zu optimieren [Abschn.4][10]. In diesem Kapitel werden zwei agile Ansätze präsentiert, die sich besonders zur Bewältigung der Herausforderungen im Umgang mit NFRs eignen und somit den Erfolg von Softwareprojekten gewährleisten. Hierbei handelt es sich um Extreme Programming (im Folgenden mit XP abgekürzt) und Scaled Agile Framework (im Folgenden mit SAFe abgekürzt).

5.1 Extreme Programming

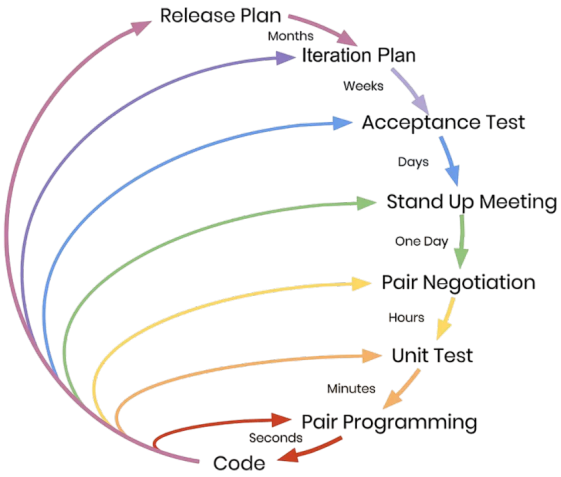

Abbildung 6: The XP Process [4]

XP bezeichnet einen agilen Ansatz, der sich durch seinen Fokus auf Softwarequalität auszeichnet. Dies wird durch Praktiken wie testgetriebene Entwicklung (im Folgenden mit TDD abgekürzt), Pair Programming, kontinuierliche Integration und regelmäßiges Feedback gewährleistet [Abschn.1][9]. TDD beschreibt eine Technik, bei der Tests vor dem eigentlichen Code geschrieben werden [Abschn.1][9]. Dies fördert eine frühzeitige und kontinuierliche Identifikation von NFRs [Abschn.5.2][12], da durch die Erstellung von Tests vor der Codeimplementierung Anforderungen klar definiert und überprüfbar gemacht werden müssen. Dies zwingt das Entwicklungsteam dazu, sowohl funktionale als auch nicht-funktionale Faktoren der Software von Beginn an zu berücksichtigen. Insbesondere bei NFRs führt dies dazu, dass erforderliche Leistungs-, Sicherheits-, Zuverlässigkeits- und andere Qualitätskriterien früh im Entwicklungsprozess definiert und in die Testfälle integriert werden. Dadurch werden etwaige Lücken oder Missverständnisse in den Anforderungen sofort sichtbar und können angepasst werden. Pair Programming beschreibt eine Methode, bei der zwei Entwickler gemeinsam am Code arbeiten [Abschn.1][9]. Einer schreibt den Code, der andere denkt über die Implementierung nach, überprüft Fehler und schlägt Verbesserungen vor. Durch den Wissensaustausch werden Fehler minimiert und die Codequalität gesteigert [Abschn.1][9]. Der Wis-

sensaustausch kann aber auch schon im Vorfeld genutzt werden, um potenzielle NFRs gezielt zu identifizieren, dokumentieren und NFRs nach Typ zu differenzieren. Durch den intensiven Austausch bringen beide Entwickler ihre unterschiedlichen Erfahrungen und Expertise ein und hinterfragen kritisch und unterscheiden, welche Anforderungen wichtig sind und wie sie umgesetzt werden können [Abschn.5][8]. Die kontinuierliche Integration gewährleistet, dass Änderungen regelmäßig in das Hauptrepository eingeführt und getestet werden, was die Stabilität des Projekts fördert [Abschn.2][9] und hilft, NFRs kontinuierlich zu überprüfen und zu validieren. Schließlich ermöglichen regelmäßige Feedbackschleifen [Abschn.1][9], dass die Entwicklung eng an den Bedürfnissen der Nutzer ausgerichtet bleibt und dass alle NFRs effektiv kommuniziert und integriert werden. Der starke iterative Charakter von XP auf allen Ebenen wird in *Abbildung 6* zusätzlich veranschaulicht. Jede dieser Aktivitäten trägt zur schnellen Identifikation und Anpassung an sich ändernde Anforderungen bei, wodurch die Stabilität und Qualität des Endprodukts gesichert wird.

5.2 Scaled Agile Framework

SAFe ergänzt die Praktiken von XP. SAFe bietet zusätzliche Mechanismen, die geeignet sind, agile Praktiken - beispielsweise von XP - auf Organisationsebene zu skalieren [Abschn.2.2][6]. Damit eignet sich SAFe auch für komplexere Projekte, die mehrere Teams involvieren [1]. Während XP sich also stark auf die technische Exzellenz und teaminterne Praktiken konzentriert, ermöglicht SAFe die koordinierte Umsetzung agiler Praktiken über verschiedene Teams hinweg und integriert strategische Planungselemente [Abschn.2][12], die in XP weniger stark betont werden. Die systematische Planungs- und Koordinationsmechanismen ermöglichen die frühzeitige und kontinuierliche Identifikation von NFRs auch in komplexen Projekten [Abschn.2][12]. Hierzu gehört beispielsweise der Einsatz von Program Increment (PI) Planning, bei dem alle agilen Teams und ausgewählte Stakeholder regelmäßig zusammenkommen, um die Prioritäten und Anforderungen für das nächste Inkrement festzulegen. Diese kollaborativen Planungssitzungen fördern eine umfassende Diskussion und zugleich Identifikation von NFRs, die sonst möglicherweise übersehen würden, und stellen sicher, dass Stakeholder-Anforderungen und Erwartungen kontinuierlich berücksichtigt und angepasst werden [Abschn.2.2.1][6]. Darüber hinaus sieht SAFe die kontinuierliche Integration von Code auf Programmebene vor [Abschn.2][6]. Dies hat zur Folge, dass mehrere agile Teams, die an verschiedenen Teilen eines größeren Produkts oder Systems arbeiten, ihre Arbeit regelmäßig zusammenführen [1]. Durch diese systematische Zusammenführung werden NFRs nicht nur regelmäßig angepasst, sondern auch im umfassenden organisatorischen Kontext validiert. Dies stellt sicher, dass die NFRs in allen Systemen und Komponenten berücksichtigt und aufeinander abgestimmt werden, wodurch eine ganzheitliche und konsistente Erfüllung der Anforderungen sichergestellt wird.

5.3 Ziele agiler Methoden

Die Hauptziele agiler Methoden im Kontext von NFRs sind schließlich die Steigerung der Transparenz, Flexibilität und Reaktionsfähigkeit [Abschn.2.1][2]. Durch regelmäßige Iterationen und Feedback-Schleifen wird sichergestellt, dass NFRs kontinuierlich bewertet und an die sich ändernden Bedingungen und Erwartungen der Stakeholder angepasst werden können [Abschn.6][12]. Dies fördert nicht nur eine bessere Produktqualität, sondern auch eine stärkere Einbindung und Zufriedenheit der Endnutzer.

5.4 Vorteile agiler Methoden im Umgang mit nicht-funktionalen Anforderungen

Agile Methoden bieten zahlreiche Vorteile im Umgang mit NFRs. Durch den iterativen Ansatz können NFRs regelmäßig überprüft werden [Abschn.1][7], was zu einer präziseren und umfassenderen Erfassung von NFRs führt. Der iterative Ansatz von agilen Methoden trägt dazu bei, dass das Produkt kontinuierlich an sich verändernde Anforderungen und Marktbedingungen angepasst werden kann. Durch diesen iterativen Ansatz können ebenso Probleme frühzeitig erkannt und behoben werden [Abschn.1][7], bevor sie sich zu einem großen Hindernis entwickeln. Die Integration einer umfassenden Teststrategie, wie sie von Methoden wie XP unterstützt wird, ermöglicht außerdem eine frühzeitige Identifizierung und laufende Validierung der NFRs [Abschn.5.2][12]. Zudem erleichtert die gezielte Ermittlung und Differenzierung von NFRs nach Typ, unterstützt durch den regelmäßigen und intensiven Wissensaustausch, eine effiziente Steuerung und Priorisierung dieser Anforderungen im Entwicklungsprozess. Die kontinuierliche Kommunikation und Einbindung von Stakeholdern [Abschn.1][7] schließlich verbessert das Verständnis und die ganzheitliche Identifizierung von NFRs und trägt dazu bei, dass das Endprodukt mit den Erwartungen der Nutzer übereinstimmt.

5.5 Mögliche Limitierung agiler Methoden im Umgang mit nicht-funktionalen Anforderungen

Trotz der vielen Vorteile gibt es auch Herausforderungen und Limitationen beim Einsatz agiler Methoden für den Umgang mit NFRs. Ein Mangel an klarer Definition und Priorisierung der NFRs zu Beginn des Projekts kann dazu führen, dass diese Anforderungen im Laufe der Entwicklung untergehen. Es ist entscheidend, dass agile Teams sowohl FRs als auch NFRs verstehen [Abschn.1][12]. Dies erfordert oft zusätzliche Anstrengungen und Ressourcen, die eingeplant werden müssen. Das Verständnis für NFRs kann durch unklare Anforderungen und mangelnde Abgrenzungen seitens der Stakeholder zusätzlich erschwert werden. Agile Teams müssen daher sicherstellen, dass die Anforderungen der Stakeholder sorgfältig evaluiert werden, um festzustellen, welche Anforderungen funktional und nicht-funktional sind und welche Anforderungen gänzlich irrelevant für die Implementierung sind.

6 Fazit und Aussicht für die Zukunft

Die vorliegende Arbeit hat die Instabilität und Unsicherheit von NFRs in der Softwareentwicklung untersucht und aufgezeigt, wie agile Methoden zur Bewältigung dieser Herausforderungen beitragen können. Die theoretische Grundlage und die empirischen Daten der Studie von Viviani et al. verdeutlichen, dass NFRs aufgrund ihrer Dynamik oft spät im Entwicklungsprozess umfassend spezifiziert werden, was zu kostspieligen Anpassungen und Projektverzögerungen führt. Agile Methoden wie XP und SAFe bieten durch ihre iterative und flexible Struktur effiziente Lösungsansätze, um NFRs möglichst frühzeitig zu identifizieren, regelmäßig zu evaluieren und anzupassen. Feedbackschleifen und die Einbindung von Stakeholdern ermöglichen die kontinuierliche Überprüfung und Validierung von NFRs im Lebenszyklus der Softwareentwicklung. Die Vorteile agiler Methoden im Umgang mit NFRs, wie die Reduktion von Unsicherheiten und Instabilität und die Verbesserung der Softwarequalität, sind klar erkennbar.
Die Zukunft des Umgangs mit NFRs in der Softwareentwicklung liegt auch weiterhin in der verstärkten Integration agiler Methoden und der systematischen Verbesserung der Spezifikations- und Validierungsprozesse. Ein zentraler Aspekt könnte die Entwicklung und Implementierung von Best Practices sein, die eine kontinuierliche Berücksichtigung von NFRs sicherstellen. Dies beinhaltet die Verbesserung der Schulung und Sensibilisierung aller Projektbeteiligten für die Bedeutung von NFRs sowie die Einführung von standardisierten Methoden zur Erfassung und Bewertung dieser Anforderungen. Darüber hinaus könnten neue technologische Ansätze zur Überwachung und Analyse von NFRs dazu beitragen, deren Verwaltung zu automatisieren. Die Integration von künstlicher Intelligenz und maschinellem Lernen bietet vielversprechende Möglichkeiten, um Veränderungen in den Anforderungen frühzeitig zu erkennen und geeignete Maßnahmen zu ergreifen. Durch künstliche Intelligenz können außerdem Muster in der Anforderungsspezifikation erkannt und evaluiert werden, um die Ermittlung von NFRs zu automatisieren. Dies würde einerseits die Genauigkeit der Spezifikationen und die Konsistenz der Anforderungen verbessern und andererseits die Differenzierung zwischen FRs und NFRs weitestgehend erleichtern. Abschließend sei nochmals auf das eingangs erwähnte Zitat von Simon Brown verwiesen: „Non-functional requirements not sounding cool isn't a reason to neglect them." Diese Aussage unterstreicht die zentrale Erkenntnis dieser Arbeit: NFRs sind für die Qualität und den Erfolg von Softwareprojekten unerlässlich und verdienen die gleiche Aufmerksamkeit wie FRs. Wenn man erst einmal ein solides Verständnis für NFRs erworben hat, erscheinen diese auch keineswegs mehr unattraktiv. Vielmehr zeigt sich, dass durch die kontinuierliche Optimierung der Methoden im Umgang mit NFRs die Softwareentwicklung zukünftig noch flexibler, reaktionsfähiger und effizienter gestaltet werden kann. Dies wird nicht nur zur Steigerung der Softwarequalität, sondern auch zur Erhöhung der Kundenzufriedenheit und Wettbewerbsfähigkeit beitragen. Und das klingt schließlich mehr als attraktiv.

Literatur

[1] *Agility at Scale*. https://agility-at-scale.com/safe/requirements-program/f [Accessed: 23.05.2024].

[2] Mehmet Biçer. „The Advantage of Being an Agile Organization in the Pandemic Crisis“. In: *Stratejik Yönetim Araştırmaları Dergisi* 4.2 (2021), S. 123–141.

[3] S Brown. *Software Architecture for Developers-Volume 1: Technical leadership and the balance with agility*. 2018.

[4] Parita Dey. *Medium*. https://medium.com/@paritasampa95/agile-software-development-extreme-programming-3101c989228f [Accessed: 23.05.2024]. 2023.

[5] Jonas Eckhardt, Andreas Vogelsang und Daniel Méndez Fernández. „Are "non-functional"requirements really non-functional? an investigation of non-functional requirements in practice“. In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. Austin, Texas: Association for Computing Machinery, 2016, S. 832–842. ISBN: 9781450339001. DOI: 10.1145/2884781.2884788. URL: https://doi.org/10.1145/2884781.2884788.

[6] Marimuthu Gopal und Abdulrahman Omar Yacoob. *Requirement Engineering using ScaledAgile Framework®(SAFe) in AutomotiveIndustry: Practices and Challenges*. 2022.

[7] Aleksander Jarzębowicz und Paweł Weichbroth. „A Qualitative Study on Non-Functional Requirements in Agile Software Development“. In: *IEEE Access* 9 (2021), S. 40458–40475. DOI: 10.1109/ACCESS.2021.3064424.

[8] Nosheen Qamar u. a. „Evaluating the Impact of Pair Documentation on Requirements Quality and Team Productivity“. In: *arXiv preprint arXiv:2304.14255* (2023). DOI: https://doi.org/10.48550/arXiv.2304.14255.

[9] M. Rizwan Qureshi und Jacob Ikram. „Proposal of Enhanced Extreme Programming Model“. In: *International Journal of Information Engineering and Electronic Business* 7 (Jan. 2015), S. 37–42. DOI: 10.5815/ijieeb.2015.01.05.

[10] Samuel Gbli Tetteh. „Empirical Study of Agile Software Development Methodologies: A Comparative Analysis“. In: *Asian Journal of Research in Computer Science* 17.5 (2024), S. 30–42. DOI: https://doi.org/10.9734/ajrcos/2024/v17i5436.

[11] Saeed Ullah, Muzaffar Iqbal und Aamir Mehmood Khan. „A survey on issues in non-functional requirements elicitation“. In: *International Conference on Computer Networks and Information Technology*. 2011, S. 333–340. DOI: 10.1109/ICCNIT.2011.6020890.

[12] Luiz Viviani u. a. „An Empirical Study About the Instability and Uncertainty of Non-functional Requirements“. In: *Agile Processes in Software Engineering and Extreme Programming - 24th International Conference on Agile Software Development, XP 2023, Amsterdam, The Netherlands, June 13-16, 2023, Proceedings*. Hrsg. von Christoph J. Stettina, Juan Garbajosa und Philippe Kruchten. Bd. 475.

Lecture Notes in Business Information Processing. Springer, 2023, S. 77–93. DOI: 10.1007/978-3-031-33976-9_6. URL: https://doi.org/10.1007/978-3-031-33976-9%5C_6.