Reviews

180.764 Software-Qualitätssicherung

17.10.2024 - WS 2024

Christina Zoffi

Wolfgang Gruber

Research Group for Industrial Software (INSO)
https://www.inso.tuwien.ac.at

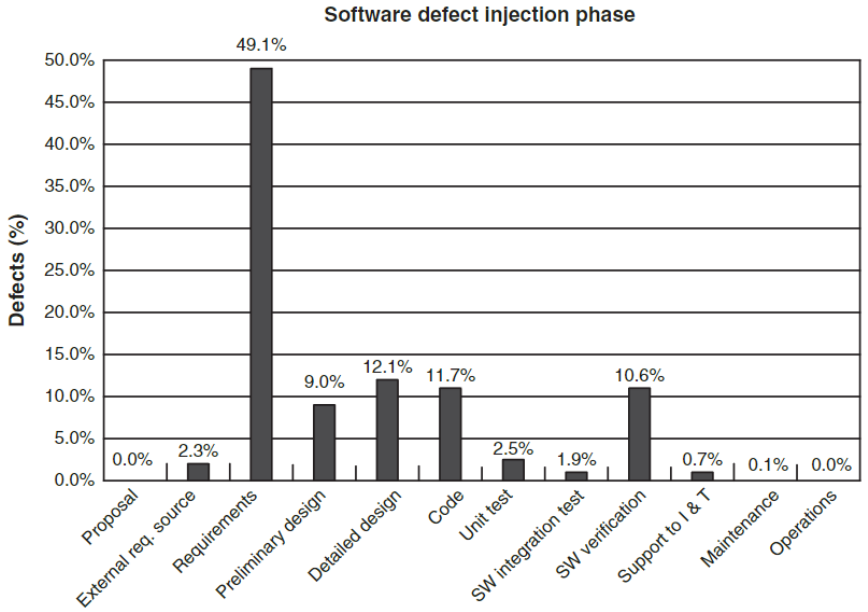
Agenda

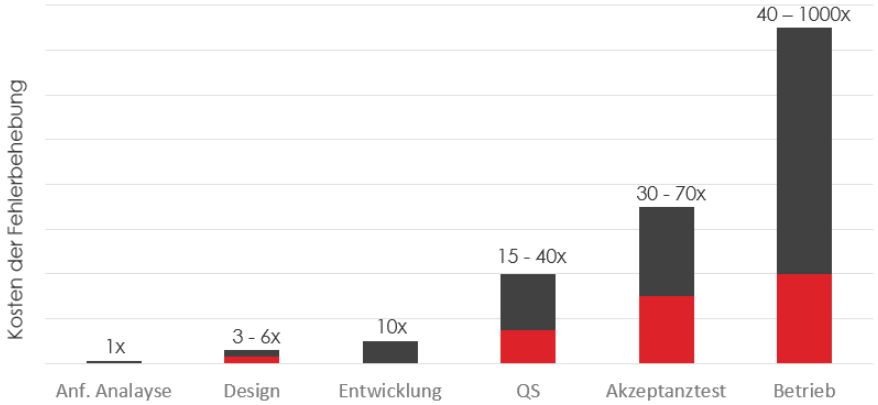
- Anforderungen und QS
- Reviews

Anforderungen und QS

Motivation QS in der Anforderungsanalyse

- Nahezu die Hälfte aller Fehler entstehen in der Phase der Anforderungsanalyse
- Fehlerkosten steigen, je später ein Fehler identifiziert/behoben wird (→ siehe auch 1. VO)





Begriffsdefinition

- „A requirement is an expression of one or more particular needs in a very specific, precise and unambiguous manner. “ (IEEE-29148)

- Anforderungen sollten den SMART-Kriterien entsprechen
  - S – Specific (präzise und eindeutig formuliert)
  - M – Measurable (messbar, überprüfbar, verifizierbar)
  - A – Achievable (erstrebenswert, erreichbar)
  - R – Realistic (realisierbar)
  - T – Time-Related (innerhalb definierter Zeitspanne erreichbar)

Klassifikation

Funktionale Anforderungen

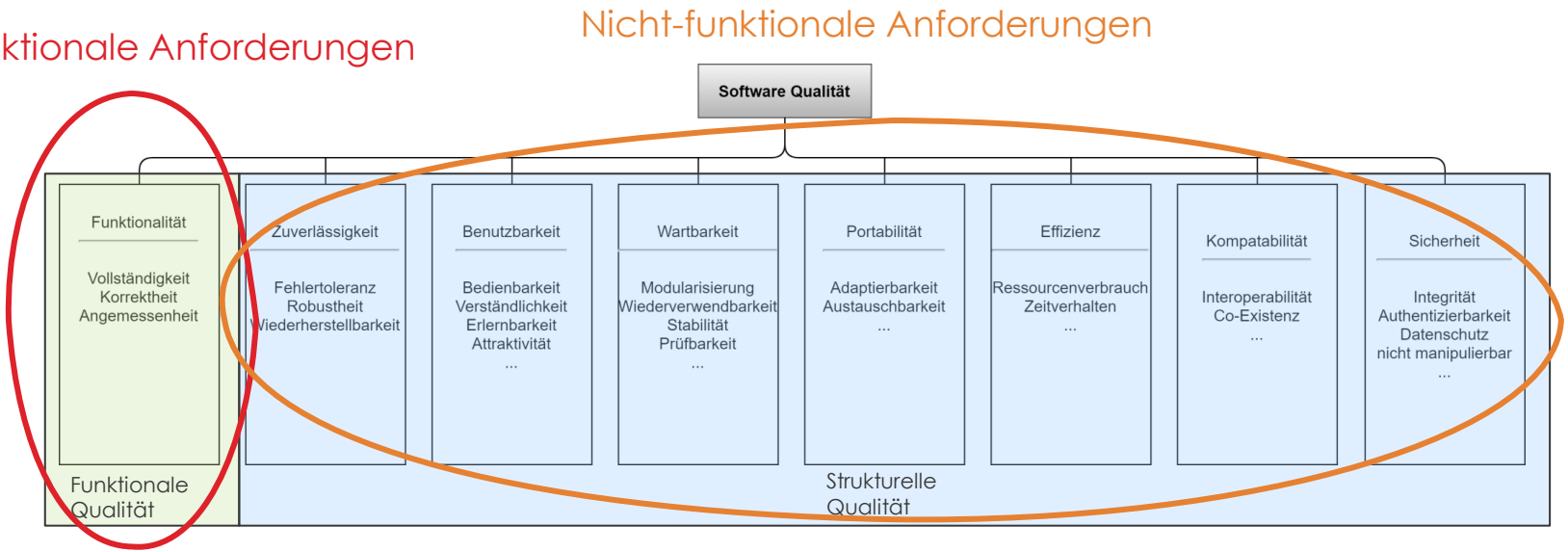
- Beschreiben das Verhalten des Systems
- Eingaben, Verarbeitungsschritte, Ausgaben des Systems
- Verhältnismäßig „einfach“ zu ermitteln
- Definieren das „was?“

Nicht-funktionale Anforderungen

- Beschreiben unter welchen Bedingungen etwas bereitgestellt werden muss, große Breite
- Erstrecken sich oftmals quer über die gesamte funktionale Anforderungsbasis
- Wesentlich schwieriger zu ermitteln (Stakeholder-Awareness, Messbarkeit)
- Definieren das „wie gut?“

Klassifikation

- Wiederholung – Bezug zu Qualitätsfaktoren aus 1. VO-Einheit



Diskussion

Warum sind diese Anforderungen problematisch?

- „Der Kreditantrag muss gespeichert werden können“
- „Das System muss schnell sein“
- „Die Verfügbarkeit muss immer gegeben sein“
- „Alle Formulare müssen benutzerfreundlich und leicht zu bedienen sein“

Was sind gute Anforderungen?

- Gute Anforderungen

Vollständigkeit	alle Aspekte sind definiert
Konsistenz	es liegen keine Widersprüche und Konflikte vor

- Beispiele für häufige Fehler
  - Fehlende Präzision (schwammig, nicht messbar, zu allgemein)
  - Vermischung (funktionaler und nicht-funktionaler Aspekte)
  - Zusammenführung (fehlende Abgrenzung, Beschreibung mehrerer Anforderungen in einem)
  - Mehrdeutigkeit (Unschärfe natürlicher Sprache)
  - Selbstverständlichkeit (implizite Annahmen, die nicht verschriftlicht sind)

Beispiele für gute Anforderungen

- Beispiele funktionale Anforderungen
  - *„Wenn der/die Antragssteller/in ein Girokonto bei der Bank besitzt, muss das System ihm oder ihr die Möglichkeit bieten, einen Kreditantrag zu erfassen“*
  - *„Der/Die eingeloggte Benutzer/in kann gespeicherte Befunde als PDF-Dokument exportieren“*

- Beispiele nicht-funktionale Anforderungen
  - *„Das System muss eine Spitzenlast von 3000 Logins/Minute verarbeiten können“ (Effizienz)*
  - *„Das System muss eine Verfügbarkeit von 95% innerhalb der Kernzeiten (06:00 – 18:00) aufweisen“ (Zuverlässigkeit)*
  - *„Formulare dürfen maximal 10 Eingabefelder pro Seite aufweisen“ (Benutzbarkeit)*

QS von Anforderungen

- Konstruktive Ansätze
  - Nutzung von Methoden und Prozessen, um Entstehung von Fehlern bereits während der Anforderungserhebung zu vermeiden, z.B.
  - → Einsatz von Checklisten, Schablonen, Templates, …
  - → Workshops, Prototyping

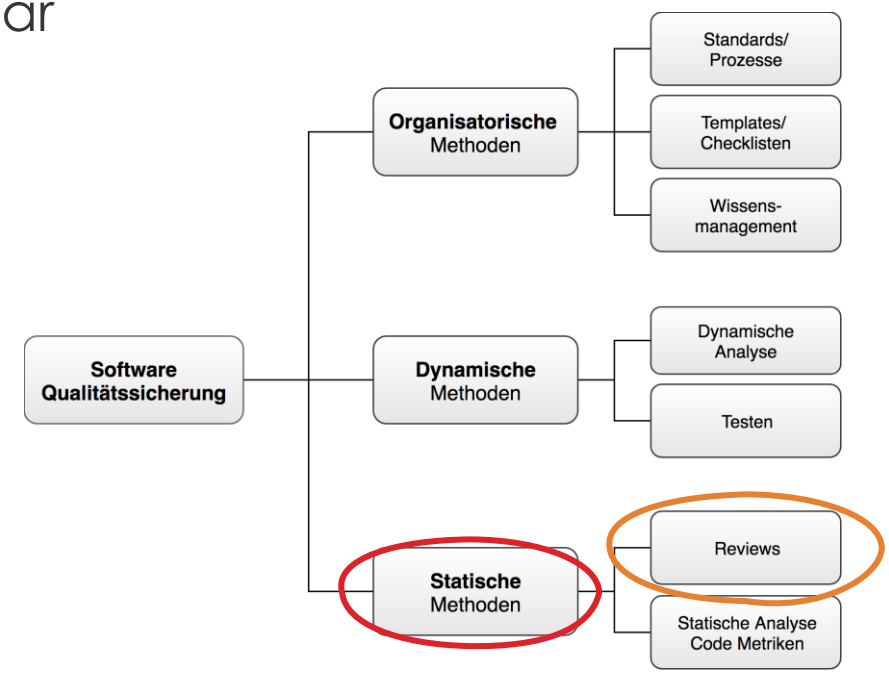
- Analytische Ansätze
  - Alle QS-Tätigkeiten, die auf bereits spezifizierte Anforderungen nachgelagert angewandt werden, z.B.
  - → Reviews
  - → Definition von Abnahmetestfällen

Reviews


Submitter
Reviewer
NO NEED TO DOUBLE CHECK THIS CHANGE LIST, IF SOME PROBLEMS REMAIN THE REVIEWER WILL CATCH THEM.
NO NEED TO LOOK AT THIS CHANGE LIST TOO CLOSELY, I'M SURE THE AUTHOR KNOWS WHAT HE'S DOING.

Wiederholung – Statische Qualitätssicherung

- Software muss nicht ausführbar bzw. integriert sein
- Fokus auf die innere Struktur („innere Qualität“)
- Methoden können bereits in frühen Phasen des SW-Entwicklungsprozesses angewandt werden
- Manuelle Prüfung von Arbeitsergebnissen sowie werkzeuggestützte Analysen



Begriffsdefinition

- „*Ein statischer Test, bei dem die Qualität eines Arbeitsergebnisses oder Prozesses von Personen bewertet wird*“ (ISTQB)
- „*A process or meeting during which a software product is presented to project personnel, management, users, customers, user representatives or other interested parties for comment or approval.” (ISO/IEC/IEEE 24765:2017)*

- Systematische Überpüfung von Artefakten in einem SW-Projekt
- Verschiedene Ausprägungen, variieren hinsichtlich
  - Formalität (von informell bis formal)
  - Zweck (z.B. Fehlerfindung, Wissensverteilung, Prozessverbesserung)
  - Flexibilität
  - Rollen

Zielsetzungen

- Frühzeitige Identifikation von Fehlern, Widersprüchen, Redundanzen und Abweichungen
- Anerkennen von guten Lösungen
- Erhöhung der Produktivität
- Reduktion von Kosten
- Bessere Wissensverteilung im Team
  - Autor(en) erhalten Feedback zur Verbesserung
  - Reviewer erhalten Einsicht und Wissen
- Gemeinsame Verantwortlichkeit (Collective Ownership)
- Herstellung von Einheitlichkeit und Konformität innerhalb eines Projekts/Unternehmens
- Kontinuierlicher Verbesserungsprozess

Anwendungsbereiche

- Reviews können grundsätzlich für alle Arbeitsergebnisse in einem Projekt angewandt werden
- Setzen auf die Fähigkeit des Menschen, komplexe Situationen zu erfassen und zu bewerten
- Eignen sich besonders, wenn keine Überprüfung durch Werkzeuge/Tools erfolgen kann oder als Ergänzung dazu

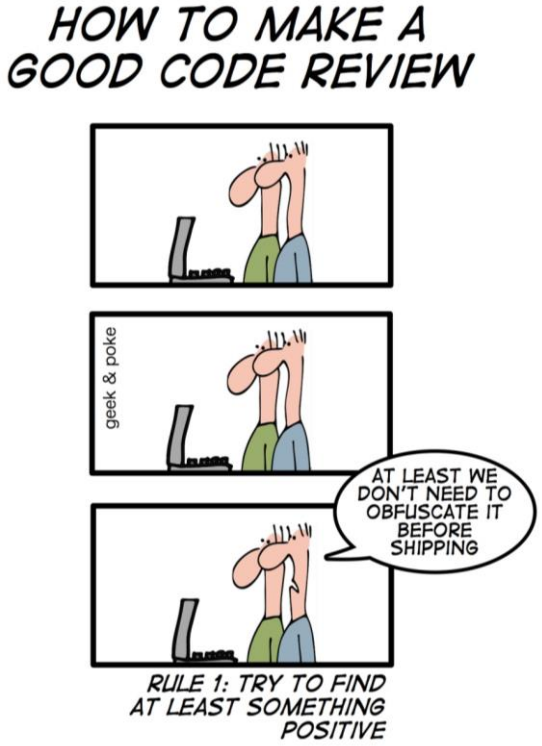
- Beispiele für Anwendungsbereiche:
  - Anforderungen und User Stories
  - Technische Spezifikationen und Entwürfe
  - Code
  - Testdokumente (z.B. Testkonzepte, Testfallspezifikationen)

Häufige Fehlertypen

- Fehler in Anforderungen
  - Abweichungen
  - Mehrdeutigkeiten
  - Widersprüche
  - Redundanzen
  - Lücken
  - …
- Fehler im Entwurf/Design
  - Abweichung von Anforderungen
  - Ineffiziente techn. Annahmen (z.B. Algorithmen)
  - Fehler in Diagrammen und Abläufen
  - Fehlerhafte Schnittstellenbeschreibungen
  - Sicherheitslücken
  - Schlechte Modularisierung
  - …
- Abweichungen von Standards, Normen und Konventionen

Erfolgsfaktoren

- Objektivität und sachliche Kommunikation
  - → Review des Objekts, nicht des Autors
  - → vertrauliche Atmosphäre
  - → keine Anschuldigungen
- Festlegung klarer Ziele
- Definition messbarer Bewertungskriterien
- Festhaltung und Nachvollziehbarkeit der Ergebnisse
- Auswahl des passenden Review-Typs
- Unterstützung und Akzeptanz in der Unternehmenskultur



Review Typen

Traditionell

- Resultierend aus Forschung der 1980/1990er
- Formalisiert, schwergewichtig, meetingzentriert
- Definierte Rollen und Prozesse
- Lesetechniken als Performance-Booster
- In Einklang mit sequenziellen Vorgehensmodellen

Modern

- Getrieben aus Praxis und Industrie
- Leichtgewichtig, toolgestützt, kontinuierlich stattfindend
- Erlauben Asynchronität und verteilte Teams
- In Einklang mit agilen Praktiken
- Insb. für Source-Code-Reviews

Review Typen

Grundlage für moderne Reviews

Inspektion	Technisches Review	Walk-Through	Peer Review
sehr formal	formal	wenig formal	informell
Identifikation von Fehlern und Anomalien Prüfung auf Erfüllung der Spezifikation Prüfung von Anforderungen	Identifikation von Fehlern und Anomalien Prüfung techn. Artefakte auf Konformität mit der Spezifikation Prüfung der Einhaltung von Standards, Regulationen, Guidelines	Identifikation von Fehlern und Anomalien Verbesserung der Qualität Erwägen von Alternativen Wissensverteilung (Ausbildung, Einschulung)	Identifikation von Fehlern und Anomalien Verbesserung der Qualität Erwägen von Alternativen Verschiedene Ausprägungen möglich
Formaler Prozess Definierte Rollen Formaler Bericht Einsatz fachlicher Experten	Formaler Prozess Definierte Rollen Einsatz technisch qualifizierter Experten	Keine generellen Vorgaben hinsichtl. Prozess und Rollen, meist nur Autor und Gutachter Oft in kleineren Teams etabliert	Keine generellen Vorgaben hinsichtl. Prozess und Rollen, meist nur Autor und Gutachter Ad-hoc Selbstorganisiert

Formale Reviews – Rollen

- Rollen und Verteilung variieren je nach Review Art
- Einsatz von Werkzeugen kann Rollen obsolet machen

Autor	Urheber (oder stv. für mehrere Urheber) des Review-Objekts Unterstützt bei Unklarheiten und führt Korrekturen und Verbesserungen durch
Moderator	Organisiert und koordiniert das Review Kommunikations- und Vermittlungsinstanz zw. den beteiligten Personen Meist Qualitätsmanager oder QS-Experte
Gutachter	Führt das Review durch und präsentiert die Ergebnisse Meist Fachexperten (techn. / fachl.), z.B. andere Entwickler, Tester, Analysten, Betreiber
Manager	Obliegt die Freigabe des Review-Objekts Meist leitende Position, Projektleiter, Testleiter, Entwicklungsleiter
Protokollant	Dokumentiert und protokolliert das Review-Meeting Hält alle wichtigen Erkenntnisse und Entscheidungen fest

Formale Reviews – Prozess

Planung	Definition der Arbeitspakete und Review Inhalte Erarbeitung von Checklisten, Richtlinien, Ziele etc. Festlegung des Teams und der benötigten Rollen

Formale Reviews – Prozess

Planung	Definition der Arbeitspakete und Review Inhalte Erarbeitung von Checklisten, Richtlinien, Ziele etc. Festlegung des Teams und der benötigten Rollen
Kick-Off	Initiierung des Reviews nach Erfüllung der Eingangskriterien (z.B. Code kompiliert) Verteilung an das Team, Festlegung der Ziele und Aufgaben

Formale Reviews – Prozess

Planung	Definition der Arbeitspakete und Review Inhalte Erarbeitung von Checklisten, Richtlinien, Ziele etc. Festlegung des Teams und der benötigten Rollen
Kick-Off	Initiierung des Reviews nach Erfüllung der Eingangskriterien (z.B. Code kompiliert) Verteilung an das Team Festlegung der Ziele und Aufgaben
Vorbereitung	Durchführung des Reviews (asynchron), intensive Durcharbeitung Dokumentation von Fehlern, Rückfragen und Kommentaren Nutzung der bereitgestellten Checklisten etc.

Formale Reviews – Prozess

Meeting	Präsentation der Review Ergebnisse Besprechung und Bewertung der identifizierten Fehler Erarbeitung von Empfehlungen und Verbesserungsvorschlägen

Formale Reviews – Prozess

Meeting	Präsentation der Review Ergebnisse Besprechung und Bewertung der identifizierten Fehler Erarbeitung von Empfehlungen und Verbesserungsvorschlägen
Nacharbeit	Behebung der identifizierten Mängel Umsetzung von Verbesserungsvorschlägen

Formale Reviews – Prozess

Meeting	Präsentation der Review Ergebnisse Besprechung und Bewertung der identifizierten Fehler Erarbeitung von Empfehlungen und Verbesserungsvorschlägen
Nacharbeit	Behebung der identifizierten Mängel Umsetzung von Verbesserungsvorschlägen
Follow-Up	Nachkontrolle der durchgeführten Arbeiten Freigabe des Reviewobjekts (oder Einberufung neues Reviews) Sammlung von Metriken/Lessons-Learned (kontinuierlicher Verbesserungsprozess)

Lesetechniken

- Hilfestellung, wie beim Review vorgegangen werden soll
- Sollen die Effizienz steigern (Fehlerrate, Fokus, Dauer, …)
- Beispiele:

Ad Hoc Ansatz

- Wenig bis keine Anleitung
- Erfordert wenig Vorbereitung
- Outcome stark vom Skillset und Sichtweise des Reviewer abhängig

Checklisten-basierter Ansatz

- Review basierend auf vorgegebenen Checklisten
- Set an Ja/Nein Fragen, die auf potenzielle Fehlerkategorien abzielen

Perspektiven-basierter Ansatz

- Betrachten des Review Objekts aus der Sicht unterschiedlicher Stakeholder (z.B. Kunde, Entwickler, Tester)
- Ermöglicht Auffinden verschiedener Fehlertypen

Lesetechniken – Beispiel

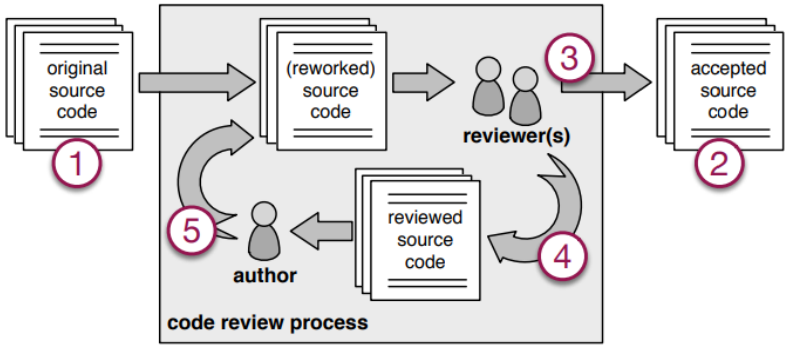
„#121: Der/Die eingeloggte Benutzer/in muss gespeicherte Befunde als PDF-Dokument exportieren können“

- Checklisten-basiert, z.B.
  - ✓ Ist die Anforderung eindeutig identifiziert?
  - ✓ Ist die Anforderung aus dem Projektauftrag entnehmbar?
  - ✓ Steht die Anforderung in Widerspruch mit anderen Anforderungen?
  - ✓ Ist die Anforderung prüfbar?

- Perspektiven-basiert, z.B.
  - Sicht „Kunde“:
    - ✓ Entspricht die Anforderung den Erwartungen an das System?
  - Sicht „Architekt/Entwickler“:
    - ✓ Lässt sich daraus ein technisches Design erstellen?
    - ✓ Lässt sich eine präzise Schnittstellendefinition ableiten?
  - Sicht „QS/Test“:
    - ✓ Ist die Anforderung testbar?
    - ✓ Lässt sich daraus eine präzise Testfallspezifikation ableiten?

Moderne Reviews

- Weiterentwicklung bzw. Ausprägung von Peer Reviews
- Einsatzgebiet besonders bei Source Code Reviews
- Zentrale Eigenschaften
  - Werkzeuggestützt
  - Informell
  - Asynchron
  - Kontinuierlich
  - Änderungsbasiert
- Nutzen Funktionen von Versionskontrollsystemen
  - Diff-View
  - Merge/Pull Requests



Moderne Reviews – Evolution

- Vergleich zw. traditionellen, formalen und modernen, leichtgewichtigen Ansätzen
  - Durchlaufzeit
    - Mehrere Wochen vs. Tage/Stunden
  - Anzahl Reviews
    - Wenige zu fixen Meilensteinen vs. häufig und kontinuierlich
  - Anzahl involvierte Personen
    - 5-30 (!) Personen vs. 1-2 Reviewer
  - Charakter
    - Prüfung eines gesamten Artefakts vs. änderungsbasierte Reviews mit kleinem Umfang

Moderne Reviews – Ausprägungen

Pre-Commit („review-then-commit“)

- Review erfolgt, bevor Änderungen in die gemeinsame Code Basis (z.B. develop/main-Branch) kommen
- Forciert einen Review-Prozess
- Kurze Zeitspanne zw. Entwicklung und Review
- Hohe Motivation zur Behebung, da es Abschluss blockiert
- Nachteil: kann entwicklungsverzögernd wirken

Post-Commit („commit-then-review“)

- Review erfolgt, nachdem Änderungen in der gemeinsamen Code Basis sind
- Ermöglicht schnellere Integration
- Nachteil: schlechte Qualität muss nachträglich bereinigt werden, Reviews können bei Zeitdruck „wegdiskutiert“ werden

Diskussion

Diskussion – Moderne Code Reviews

Welche Möglichkeiten fallen Ihnen ein, Pre-Commit-Reviews mithilfe moderner Tools (z.B. Gitlab) umzusetzen?

Welche Erfahrungen haben Sie damit gemacht?

Moderne Reviews – Pre-Commit

- Verschiedene Konzepte zur Ausgestaltung
  - Ohne Toolsupport
    - Gemeinsames Walkthrough der Änderungen
    - Export und Übermittlung eines Änderungs-Patches
    - Sonderform: Pair-Programming
  - Mit Toolsupport
    - Nutzung von Source-Code-Management-Plattform (z.B. GitLab …)
      - Verwendung von Branching-Konzepten für die Entwicklung (z.B. Feature-Branches)
      - Schreibschutz (protected branches) auf Public-Branches (z.B. main, develop)
      - Einsatz von Merge-Requests / Pull-Requests
      - Nutzung der Kommentierungs- und Approval-Funktion
    - Spezielle Pre-Commit-Reviewtools, z.B. Gerrit Code Review

Diskussion

Beispiel Review Prozess in einem SEPM-Projekt

In welchen Phasen bzw. zu welchen Meilensteinen im Projekt sollten Reviews eingesetzt werden? Welche Reviews eignen sich dafür?

Zusammenfassung

- Die Qualität von Anforderungen trägt maßgeblich zum Projekterfolg bei, ihre Qualitätssicherung ist daher besonders wichtig, um hohe Folgekosten zu vermeiden
- Anforderungen können in funktional und nicht-funktional unterteilt werden
- Reviews sind eine statische QS-Maßnahme, die eine systematische Überprüfung div. Artefakte in einem SW-Projekt ermöglichen
- Je nach Einsatzzweck stehen verschiedene Review-Typen zur Verfügung, die sich in Formalität, Rollen, Prozess und Flexibilität unterscheiden
- Im Kontext von Code Reviews haben sich in den letzten Jahren leichtgewichtige, toolgestützte Ansätze durchgesetzt, die sich gut in den Arbeitsalltag der Entwicklung integrieren lassen

Referenzen

- Thomas Grechenig, Mario Bernhart, Roland Breiteneder, Karin Kappel: „Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten“, Pearson Studium, 2009
- ISO/IEC 25010., ISO/IEC 25010:2011, Systems and software engineering, Systems and software Quality Requirements and Evaluation (SQuaRE), System and software quality models, 2011
- IEEE Standard for Software Quality Assurance Processes, in IEEE Std 730-2014 (Revision of IEEE Std 730-2002), 2014
- ISO/IEC/IEEEE 24765:2017, Systems and software engineering Vocabulary, 2017
- Neil Walkinshaw, „Software Inspections, Code Reviews, and Safety Arguments“, 2017
- Daniel Galin, “Software Quality: Concepts and Practice”, IEEE Computer Society, Inc., 2018
- Gernot Starke, “Kolumne: Quality-driven Software Architecture: Herausforderungen mit Qualität”, Javamagazin, 1/2023, S.50-52
- Claude Laporte, Alain April, “Software Quality Assurance”, Wiley, 2018
- ISTQB, Certified Tester Lehrplan Foundation Level, V. 4.0.1a, 2023
- A. Bacchelli, C. Bird; Expectations, outcomes and challanges of modern code review, ICSE’13 Proceedings of the 2013 International Conference on Software Engineering, 2013
- M. Beller, A. Bacchelli, A. Zaidman, E. Juergens; Modern Code Reviews in Open-Source Projects: Which Problems Do They Fix?, MSR‘ 14, 2014