Abgabe SE2 Übung 9

Furkan Hidayet Rafet Aydin, M1630039
Falk-Niklas Heinrich, M1630123

2022-12-03

Aufgabe 9.2: Qualitätssicherung und V-Modell XT (Pflichtaufgabe)

a) Erläutern Sie die Bestandteile dieses Vorgehensbausteins.

Der für obligate Vorgehensbaustein ist vom QS-Verantwortlichen sowie einem Prüfer zu bearbeiten.
Im Artefaktbereich Qualitätssicherung erstellt der QS-V. das QS-Handbuch sowie eine Nachweisakte, dem Prüfer obliegt das Anlegen und führen des Prüfprotokolls und der Prüfspezifikation.
Für das Berichtswesen soll der QS-Verantwortliche Qualitätssicherungsberichte anfertigen.

Im QS-Handbuch werden die Qualitätssicherungsziele sowie die QS-Maßnahmen definiert, auch der Umfang der QS-Berichte wird hier festgehalten. Die Nachweisakte enthält Protokolle für eventuell nachzuweisende Prüfungen. Wie der Prüfer das Projekt prüfen soll, wird in der Prüfspezifikation verankert, die Ergebnisse dieser Prüfungen werden im Prüfprotokoll notiert. Die nach Festlegung im Handbuch regelmäßig anzufertigen QS-Berichte beinhalten neben Angaben und Ergebnisse der durchgeführten Prüfungen, aufgetretene QS-Probleme und die ergriffenen Gegenmaßnahmen.

b) Bewerten Sie, welche der im Vorgehensbaustein aufgeführten Projektergebnisse (zu erstellenden Artefakte) in einem studentischen Bachelorprojekt sinnvoll sein können.

- **Nachweisakte**, wenig sinnvoll

  Da in der Nachweisakte Bestätigungen der Konformität zu externen Normen und Prüfungen abgelegt werden, ist ihr Einsatz im studentischen Projekt so gut wie nie notwendig.

- **Prüfprotokoll**, wenig sinnvoll (abhängig von Projekt)

  Wie die meisten Vorgehensweisen und Berichte im V-Modell XT sind auch die Prüfprotokolle zu bürokratisch. In kleinen Projekten erbringt die formalisierte Erfassung keinen Vorteil, zumal sich überhaupt die Frage nach der Person des Prüfers stellt.
  Sollte das Projekt hingegen eine gewisse Größe erreichen und ein QS-Prüfer explizit bestimmt sein, kann er die Prüfergebnisse und Empfehlungen hier festhalten und so mit der Projektgruppe klar kommunizieren.

- **Prüfspezifikation**, wenig sinnvoll

  Auch hier ist wieder die überbordende Bürokratie die Gegenanzeige.
  Nur für komplexe Projekte und Produkte lohnt es sich für QS-Prüfungen eigenen Vorgaben anzulegen.

- **QS-Handbuch**, (wenig) sinnvoll

  Unter der Bedingung der Beschränkung des Aufwandes zur Erstellung eines QS-Handbuchs kann es auch in kleineren Projekten sinnvoll sein.
  Schriftliches Festhalten der Qualitätsziele und Maßnahmen kann die QS im Projekt verankern und bedingt zuvor eine Diskussion über Qualität im Projektteam.

- **QS-Bericht**, sinnvoll (abhängig von Projekt)

  Regelmäßiges Überprüfen ist Kernbestandteil jedes QS-Systems, daher sind auch die Berichte zu diesen Überprüfungen wichtig. Im studentischen Projekt ist das Berichtswesen allerdings nur sinnvoll, falls die Berichte einen, für alle Projektmitglieder ersichtlichen, Vorteil bringen und auch über Autorität verfügen.

Aufgabe 9.3: Begriff “Rolle”

Wenn einem Mitarbeiter (oder auch externen Person) eine Rolle zugewiesen wird, wird eine Aufgabengruppe und die Verantwortung dafür dieser Person zugeordnet. Das erlaubt die Aufteilung von Arbeit und Verantwortung und ermöglicht auch die Auswahl einer passend qualifizierten Person, da die Anforderungen mit der Rolle verknüpft sind.

Daraus ergibt sich auch die Möglichkeit, einer tatsächlichen Person mehr als eine Rolle zuzuweisen.

Aufgabe 9.4: Reviews versus Audits

Reviews dienen der Feststellung der Qualität oder Qualitätsproblemen eines Projektergebnisses, z.B. von Codeteilen. Sie werden von projektinternen Personen ausgeführt und dauern normalerweise kürzer als einige Stunden.

Hingegen prüfen Audits Projektstandards und -prozesse, werden von Projektexternen durchgeführt und sind von deutlich längerer Dauer.

Sowohl Review als auch Audit sind analytische QS-Maßnahmen.

Aufgabe 9.5: (Software-)Metriken und Qualität

a)

> Die bekannteste Metrik in der Softwareentwicklung ist LOC (lines of code bzw. SLOC = Source Lines of Code). Was sagt LOC über die Produktivität eines Softwareentwicklers aus? Warum ist LOC keine passende Metrik für objektorientierte Software?

SLOC sagt nichts über die Effizienz eines Programmes aus und kann daher auch nichts über die Produktivität des Programmierers aussagen.

Der Betrag SLOC hängt von vielen Einflüssen ab, unter anderem:

- Jede Programmiersprache hat eine unterschiedliche Expressivität, auch je nach Problemstellung.

- Die Architektur bedingt viele Unterschiede.
- Wahl der Algorithmen: Auslegung auf Performanz kann Komplexität stark steigern.
- Wird ein Framework verwendet? Wird es mitgezählt?
- Zählweise umstritten

b)

> Für die objektorientierte Softwareentwicklung wurden spezielle Metriken entwickelt. Recherchieren Sie nach den Metriken DIT, NOC, WMC, CBC und erklären Sie diese jeweils kurz. Welche Rückschlüsse lassen sich jeweils auf die Qualität der Software ableiten?

DIT

Depth of Inheritance Tree ist die maximale Länge von einem Knoten (z.B. Klasse) zur Projektwurzel.

Je größer die DIT-Zahl, desto komplexer ist die Klassenhierarchie. Damit wird auf der einen Seite die Möglichkeit zur Nutzung von ererbten Methoden wahrscheinlicher, aber auch die Analyse der Funktion einer Klasse schwieriger. Somit steigt auch der Aufwand zur Wartung.

(Chidamber und Kemerer 1994, 483ff.)

NOC

Number of Children: Anzahl der unmittelbaren Kinder einer Klasse.

Je größer NOC, desto mehr Code kann aus übergeordneten Klassen wiederverwendet werden, was die Effizienz steigert. Eine hohe Zahl kann aber auch falsche Vererbung hindeuten und macht Codeänderungen komplex.

(Chidamber und Kemerer 1994, 485f.)

WMC

Weighted Methods per Class: Summe der Komplexitätsfaktoren aller Methoden einer Klasse.

Je höher WMC, desto komplizierter und damit aufwändiger zu programmieren ist eine Klasse. Eine Klasse mit hohem WMC-Wert ist wahrscheinlich sehr spezifisch und lässt sich dadurch schlecht wiederverwenden.

(Chidamber und Kemerer 1994, 482f.)

CBC (CBO[1])

Coupling between object classes: Die Anzahl an Verbindungen einer Klasse zu anderen Klassen.

Je höher CBC, desto enger ist die Kopplung einer Klasse an andere. Ein hoher Wert zeigt also das genau Gegenteil von Modularität (lose Kopplung) umgesetzt wurde. Außerdem benötigt eine Klasse mit vielen Verbindungen eine engmaschige Testabdeckung.

(Chidamber und Kemerer 1994, 486f.)

Aufgabe 9.6: Code Review

> Nennen Sie drei Gründe, warum bei der Software-Entwicklung Code Reviews zusätzlich zu den obligatorischen Tests eingesetzt werden sollten.

- Softwaretest werden zumeist von selben Entwickler geschrieben, welcher auch die Funktion programmiert hat. Das führt dazu, dass evtl. Denkfehler sich auch in den Tests wiederfinden.
- Durch Reviews wird sichergestellt das auch andere Entwickler den Code (bzw. Gedankengang) verstehen.
- Codereviews führen zu mehr Beteiligung: Es bildet sich kein unersetzbarer Spezialist heraus.

[1] *Coupling Between Classes* wird häufig genannt, dann aber auf *Coupling Between Object Classes* aus (Chidamber und Kemerer 1994) verwiesen; z.B. Software Metrics Lecture, Univ. Kent, Dept. CS

Chidamber, S. R., und C. F. Kemerer. 1994. „A metrics suite for object oriented design“. *IEEE Transactions on Software Engineering* 20 (6): 476–93. https://doi.org/10.1109/32.295895.