Woche 07
Produktqualität

Das Thema Qualitätssicherung kommt bei uns immer etwas zu kurz, deswegen fangen wir hier mit einem ersten Workshop in der Wachstumsphase an. Am Projektende wollen sie eine Software, Dokumente oder einen kommentierten Foliensatz liefern, der relativ frei von Funktions-, Qualitäts- oder juristischen Mängeln ist. Wir starten mit ihrer Vergangenheit, denn aus Fehlern und Problemen lernt man am besten:

1. QS-Checkliste erstellen

Atul Gawande beschreibt die Bedeutung von Checklisten in verschiedenen Berufen, besonders in der Medizin in seinem „Checklist-Manifesto“. Sie können im Laufe mehrerer Sprints das Testen und ihre Qualitätssicherung über eigene Checklisten verbessern. *Checklisten entstehen aus der Erfahrung.* Sie können beispielsweise auf der nächsten Retrospektive damit beginnen und alle bisher aufgetretenen Probleme als Punkte auf einer Checkliste aufführen. Nachfolgend ein Beispiel für eine Checkliste aus einem Projekt:

1. Wird der vorgeschriebene Styleguide durchgängig benutzt?
2. Sind die Dialoge entweder modal oder nicht modal? (nicht abwechseln)
3. Gibt es Abhängigkeiten von der Hardware (z.B. Bildschirmauflösung)?
4. Wann werden Eingaben geprüft? Sofort bei Verlassen des Feldes oder erst beim Bestätigen? Ist das einheitlich gelöst?
5. Wie werden Fehler angezeigt, als gesondertes Fenster oder direkt im Dialog? Ist das einheitlich gelöst?

Führen Sie als Moderator mit ihrem Team folgendes Brainstorming durch: Überlegen Sie, was in Ihrem SEP Projekt oder in anderen Projekten in FWPM bisher schief gegangen ist. Womit hatten sie am Ende des Projektes die größten Probleme? Was hat Sie am Ende noch mal richtig Zeit gekostet? Was ist ihnen mit einigem zeitlichen Abstand eventuell jetzt peinlich (z.B. im Bereich der Code-Qualität), was würden sie nie wieder so machen?

- Jedes Teammitglied sollte mehrere dieser Probleme in einem Brainstorming auf eine Haftnotiz schreiben. (Einzelarbeit).
- Sie konsolidieren die Beiträge der einzelnen Teammitglieder (Clustern, doppelte entfernen).
- Das Team und Sie erstellen aus den Beiträgen eine Checkliste mit dem Ziel, dass Sie diese wirklich bei der Lieferung an ihren Auftraggeber einsetzen können.

Die Checkliste halten sie in ihrem Wiki-Fest. Wir werden ihr Ergebnis am Semesterende dagegen prüfen, sie müssen ihre Checkliste jedoch nicht vollständig erfüllen (der Erfüllungsgrad fließt nicht in ihre Note ein).

2. Maßnahmen zur QS planen

Folgende Maßnahmen sollten Sie sowieso unabhängig von ihrer Checkliste ergreifen:

1. Beschäftigen sie sich mit den Unit-Testing Frameworks in ihrem Programmiersprachen-Umfeld, also beispielsweise Java und JUnit. Versuchen sie, wenigstens Smoke-Tests einzubauen. Möglicherweise ist sogar die Automatisierung der GUI-Tests möglich.
2. Automatisieren sie den Build-Prozess und bauen Sie eine CI oder sogar CD Pipeline auf. Die Build Automatisierung sollte auch die Unit-Tests mit ausführen. Denkbar sind auch Tests gegen Sicherheitslücken oder Lizenzprobleme (vgl. Gitlab-Dokumentation).
3. Versuchen sie in die Art und Weise, wie sie Gitlab bedienen, ebenfalls Qualitätssicherung über Code-Reviews zu etablieren. Verwenden Sie Merge Requests und führen sie beim Merge ein Code Review durch!
4. Alle Dokumente laufen bei ihnen Teamintern einen Review-Prozess und werden von mindestens einem Reviewer oder einer Reviewerin geprüft, bevor diese an den Coach bzw. den Auftraggeber herausgehen.

Überlegen Sie im Team nun abhängig von ihren Produktrisiken, welche weiteren QS-Maßnahmen sie ergreifen können, um die Qualität ihrer Ergebnisse zu Messen / zu Testen und ggf. auch zu verbessern.

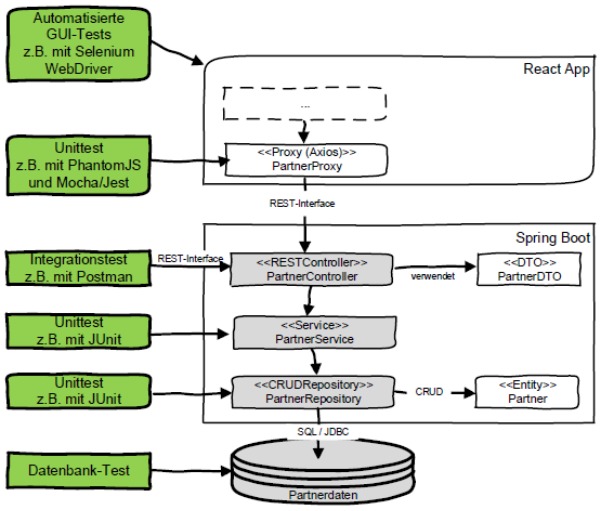
- Erstellen Sie hierzu **eine Wiki-Seite** mit den von ihnen vorgeschlagenen Maßnahmen. Stellen Sie diese ihrem Coach vor. Verwenden sie etwa folgende Tabelle

	Werkzeug	Erfolgsmaß	Wo/Wann
Funktionsumfang	JUnit-Tests am Backend, Datenbank gemockt, mit JaCoCo	80% Zweigüberdeckung	Jeder Push auf jeden Branch, in Pipeline
	Cypress UI-Tests am Frontend	Smoketest, jeder Dialog einmal durchlaufen	Bei Push auf Develop Branch in Pipeline
	User Storys mit Postman	Jede User-Story über Postman-Collection darstellen	Bei Push auf Develop Branch, in Pipeline
	Nutzerakzeptanztest, manuell Explorativ, auf basis Test Charters für die wichtigsten Funktionsbereiche	Nutzerakzeptanztest, manuell explorativ	Vor Inbetriebnahme Von STAGE auf PROD
Wartbarkeit	SonarQube verfolgt Testcoverage, Technische Schulden, Code Duplikate (Rosenheim-Profil)	0 Technische Schulden Maximal 10% Duplikate Keine Vulnerabities / Bugs	Jeder Push auf jeden Branch, in Pipeline
	Arbeit mit Feature Branches, dort Merge-Request stellen, manuelles Code-Review	Anmerkungen zum Merge Request	Bei Push auf Develop Branch, manuell (Merge Request)

- Erstellen Sie zu jeder Maßnahme ein Ticket in Gitlab und weisen sie das einem Verantwortlichen zu.

3. Testautomatisierung prüfen

Agile Methoden enthalten in der Regel keine expliziten Testphasen zur Konsolidierung des Produktes. Dies geschieht entweder außerhalb der Entwicklung (vgl. z.B. Water-Scrum-Fall) oder muss während der laufenden Entwicklung vom Team selbst geleistet werden. Damit dieses Konzept sicher funktioniert, ist eine Automatisierung der Tests sowie die Kontinuierliche Integration bzw. kontinuierliches Liefern erforderlich (CI/CD). Die Grafik zeigt einige Stellen, an denen sie mit automatisierten Tests etwas erreichen können.



Überlegen Sie sich in ihrem Team eine sinnvolle Dosis an Testautomatisierung. Welche Tests sind für ihr Team sinnvoll automatisierbar, unter der Randbedingung, dass diese Tests in der CI/CD-Pipeline ausgeführt werden können.

Hier die Aufgabe:

Zeichnen Sie als Moderator (wenn möglich) die T-Architektur (vgl. Skript SEP) ihres Produktes auf. Diese Architektur zeigt, wie ein Request vom Benutzer aus über verschiedene Teile (Klassen/Module) ihres Produktes verarbeitet wird, bis z.B. der Datenbankzugriff erfolgt. In technischen Systemen wäre das der Weg von einem erfassten Wert über einen Sensor, bis dann ein Aktor angesprochen wird.

1. Gehen sie mit ihrem Team jede Verarbeitungsstufe durch und überlegen, welche Möglichkeiten sie dort für die Testautomatisierung hätten. Z.B. an der GUI wären das beispielsweise Selenium, Appium oder Cypress.

2. Schreiben sie für jede Möglichkeit eine Haftnotiz und überlegen auch in welcher Stufe ihrer CI-Pipeline dieser Test möglich wäre. Einige Pipelines haben eine Akzeptanztest-Stage. Hier könnten sie beispielsweise den GUI-Test einbauen. Bei der Build-Stage finden sich beispielsweise die Unit-Tests.
3. Gehen sie nun ein zweites Mal die Haftnotizen durch und überlegen im Team, ob sich der Aufwand für die Testautomatisierung für sie lohnen würde. Wägen sie also gewonnene Sicherheit oder gesparten Testaufwand gegen den notwendigen Aufwand ab.

Wenn sich eine Automatisierungstechnik als für sie sinnvoll erweist, planen sie die Umsetzung bis zum Projektende im Januar. Mir ist mit der Aufgabenstellung wichtig, dass sie einmal bewusst durchgehen, was sie tun könnten und sich dann bewusst für oder gegen entsprechende Maßnahmen entscheiden. Zum Zeitpunkt des Workshops haben sie vermutlich pro Person weniger als 6 Personentage an Aufwand noch zur Verfügung.