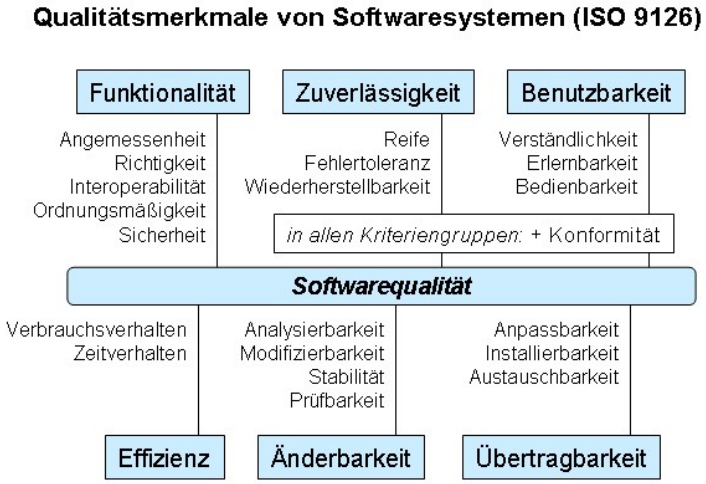
Warum Softwaretest?

Das Testen einer Software und die Dokumentation sind fester Bestandteil der Softwareentwicklung. Leider wird dies immer mal wieder vernachlässigt. Dabei sollte frühzeitig mit dem Testen begonnen werden, um schnell Fehler aufzudecken und diese zu beheben. Ein zu spätes Erkennen von Fehlern versucht einen erhöhten Aufwand bei deren Behebung und damit auch hohe Kosten. Damit besteht die Gefahr, dass ein Softwareprojekt aus finanzieller Perspektive scheitert.

Damit ist das Testen verantwortlich für die Softwarequalität.



Quelle: https://de.wikipedia.org/wiki/Softwarequalität#/media/Datei:ISO_9126_Grafik.png

Eine weitere Grundlage ist die Definition **IEEE 829 Standard for Software Test Documentation.** Hierbei handelt es sich von um einen Standard vom IEEE (Institute of Electrical and Electronics Engineers), der einen Satz von acht Basis-Dokumenten zur Dokumentation von Softwaretests beschreibt. Dieser Standard wurde dann 2013 durch die **ISO/IEC/IEEE 29119** ersetzt.

Optionale Vertiefung

Kurzüberblick zur Standardreihe ISO/IEC/IEEE 29119

https://t1p.de/8fcg6

Prozessschritte beim Testen

Unabhängig vom Vorgehensmodell – agil oder klassisch – sind die Grundlagen für das Testen von Software identisch. Folgende Punkte müssen beim Testen beachtet werden:

- Definition der Fehlerklassen und Bewertung der Schwere der Fehler
- Testkriterien und Abnahmekriterien
- Testumfang und Testabdeckung
- Umfang der Testobjekte
- Prioritäten im Test und Risikobewertung
- Testarten, Teststufen, Testdaten und Testumgebung
- Ressourcen- und Testteamplanung
- Teststeuerung und Überwachung
- Dokumentation des Testprozesses

Arbeitsauftrag

Führen Sie die Learning-App durch und halten Sie die Ergebnisse hier im Skript fest.

Learning-App: Der Testprozess

https://t1p.de/cg8iu

1. Testplanung: beschreibt den vollständigen Testprozess des jeweiligen Softwareprojekts
2. Testentwurf: Hierbei werden die Details der einzelnen Testobjekte betrachtet und dafür einzelne Test entworfen
3. Testvorbereitung: Aufbauend auf den Testspezifikationen werden die dort festgelegten Testszenarien vorbereitet und die benötigten Ressourcen zur Verfügung gestellt.
4. Testdurchführung: Alle Tests werden auf Grundlage der Testspezifikation durchgeführt und dokumentiert.
5. Testauswertung: Jedes Testobjekt wird einzeln bewertet, die Ergebnisse je Testfall überprüft und eine Liste aller Abweichungen erstellt.
6. Testabschluss: Nach Freigabe der gesamten Software kann der Testprozess abgeschlossen werden und alle Dokumente werden archiviert.

Unterschiedliche Testverfahren

Testverfahren können in statische und dynamische Testverfahren unterteilt werden. Statische Testverfahren dienen hauptsächlich dem Test von Software, ohne diese auf einem Rechner auszuführen. Bei den Prüfungsobjekten handelt es sich meistens um den Quellcode selbst. Bei den dynamischen Testverfahren wird versucht Fehler aufzuspüren, indem das Programm in einer Testumgebung ausgeführt wird.

Arbeitsauftrag

Ordnen Sie die unterschiedlichen Testverfahren den dynamischen oder statischen Testverfahren zu.

Whitebox-Test, Audit, Inspektion, Walkthrough, Blackbox-Test, Review

Testverfahren	
Statische Testverfahren	**Dynamische testverfahren**
Inspektion Review Walkthrough Audit	Blackbox-Test Whitebox-Test

Erklärvideo zu den Testverfahren **https://t1p.de/7eqq6**	

Was sind Whitebox-Test und Blackbox-Test? Beitrag Dev-Insider.de **https://t1p.de/yrt0u**	

Learning-App: Testverfahren **https://t1p.de/01z69**	

Hinweis: Black-Box- und White-Box-Tests werden im weiteren Verlauf des Moduls noch genauer betrachtet und praktisch durchgeführt.

Die Test-Pyramide

Die Testpyramide von Mike Cohn beschreibt ein Konzept zum Einsatz der genannten automatisierten Softwaretests. Die Pyramide besteht aus drei Leveln, aufgebaut nach Einsatzhäufigkeit und Relevanz. Sie ordnet also nicht nur ein, sondern bewertet Unit-Tests, Integrationstests und Systemtests auch.

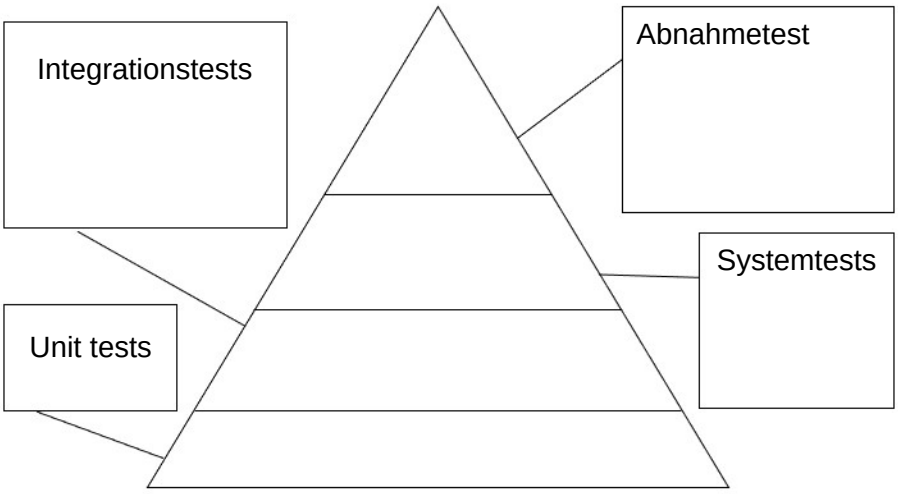
Die breite Basis der Testpyramide bilden im Idealfall viele schnelle und einfach zu wartende Unit-Tests. So können die meisten Fehler schon in frühen Phasen der Entwicklung entdeckt werden. Auf dem mittleren Level befinden sich die Integrationstests. Sie leisten wertvolle Dienste bei der zielgerichteten Prüfung von kritischen Schnittstellen. Die Ausführungszeiten von Integrationstests sind länger und auch ihre Pflege ist aufwändiger als die von Unit-Tests.

Die Spitze der Pyramide besteht aus dem Abnahmetest. Dieser beinhaltet den Test des Gesamtsystems in der Kundeumgebung. Er wird i.d.R. durch den Kunden selbst oder dessen Beteiligung durchgeführt und ist die Grundlage für die Abnahme. Als nächstes folgen die langsamen Systemtests, die mitunter sehr wartungsintensiv sind. Systemtests sind sehr hilfreich, um die Funktion der Anwendung als Ganzes zu prüfen. Zum Testen von Verzweigungen im Code sind sie jedoch ungeeignet. Da die Tests des oberen Levels zeitintensiv und somit teuer sind, muss der Einsatz von aufwendigen Systemtests sorgfältig abgewogen werden. Grundsätzlich sollten so viele Tests wie möglich auf den untersten Level (Unit-Tests) verlagert werden. Tritt bei Integrations- oder Systemtests ein Fehler auf, muss dieser nicht nur behoben, sondern auch ein entsprechender Unit-Test geschrieben werden, um ein erneutes Auftreten zu verhindern.

Quelle: https://www.openknowledge.de/blog/die-testpyramide#:~:text=Die%20Testpyramide%20von%20Mike%20Cohn%20beschreibt%20ein%20Konzept%20zum%20Einsatz,Tests%2C%20Integrationstests%20und%20Systemtests%20auch.

Arbeitsauftrag

Vervollständigen Sie die Test-Pyramide mit den jeweiligen Tests und deren Definition.



LearningApp: Test-Pyramide **https://t1p.de/qyu29**	