Software testen und dokumentieren

Sie haben erfolgreich eine Anwendung zur Verwaltung von Daten erstellt. Nun wollen Sie diese Anwendung abschließend testen und dokumentieren.

Das Testen von SW und die Erstellung von Dokumentationen gehören genauso zum Alltag der SW-Entwicklung wie das Programmieren selbst. Auch wenn das Testen und die Dokumentation von SW oftmals nicht die gebührende Aufmerksamkeit erfahren, kann ihre Bedeutung für die Entwicklung und Erweiterung von SW nicht hoch genug eingeschätzt werden.

Das Testen und Dokumentieren von SW sollte so früh wie möglich begonnen werden, denn die Korrektur von Fehlern verursacht umso weniger Kosten, je früher diese entdeckt werden. Fehler, welche sich in der Entwicklungsphase in der Regel leicht korrigieren lassen, sind im Live-Betrieb einer Software nur noch mit erheblichem Mehraufwand zu beseitigen.

Das Testen von SW verfolgt das Ziel, mit vertretbarem Aufwand an Zeit und Personal möglichst verlässliche Aussagen über das Verhalten von Software zu machen und darüber, ob die gestellten Anforderungen erfüllt werden.

In vielen Fällen wird das Testen auf die **funktionalen** Anforderungen reduziert, dabei sind die **nicht funktionalen Anforderungen** genauso wichtig. Allerdings sind diese schwerer zu testen, weil sie stärker subjektiven Einflüssen unterliegen.

Das Testen wird heutzutage nicht mehr als isolierte Phase im SW-Entwicklungsprozess angesehen, welche erst nach der Implementierung einsetzt, sondern als eine wichtige Maßnahme zur Qualitätssicherung, welche sich durch alle Phasen der SW Entwicklung zieht.

Arbeitsauftrag 1:

1. *Lesen Sie sich den Infotext auf Seite 2 zu funktionalen und nicht funktionalen Anforderungen aufmerksam durch.*
2. *Bearbeiten Sie im Anschluss die Aufgaben 1 – 3.*
3. *Präsentieren Sie Ihre Ergebnisse vor der Klasse.*

Funktionale und nicht funktionale Anforderungen

Infotext:

Kriterien bei der der Formulierung der Anforderungen sind:

- **Korrektheit**: Die Anforderung muss sehr genau den Bedürfnissen des Auftraggebers entsprechen. Außerdem muss die Gesamtheit der Anforderungen vollständig sein.
- **Eindeutigkeit**: Die Formulierung der Anforderung erlaubt nur eine gültige Interpretation. Es darf zu keinen Überschneidungen kommen und Anforderungen dürfen sich nicht widersprechen.
- **Prüfbarkeit**: Es muss Möglichkeiten geben, um nachzuweisen, ob die Anforderung erfüllt ist oder nicht.
- **Nachverfolgbarkeit**: Der Umsetzungsprozess der Anforderungen muss nachvollziehbar sein.

Frage: Welche **Methode** haben Sie bereits kennengelernt, um Anforderungen nach den oben genannten Kriterien zu definieren?

S.M.A.R.T.

Die Anforderungen als solche lassen sich in unterschiedliche Kategorien einteilen:

Funktionale Anforderungen:

Funktionale Anforderungen beziehen sich darauf, was das System leisten soll. Diese sind von Software zu Software unterschiedlich und sehr spezifisch. In den funktionalen Anforderungen wird beschrieben, welche einzelnen Funktionen und Funktionskomplexe die zu entwickelnde Software haben soll. Die dazu benötigten Eingabedaten, zwingend erforderliche Verarbeitungsschritte und die erwarteten Ausgaben werden dazu genau spezifiziert.

Nicht funktionale Anforderungen:

Nicht funktionale Anforderungen beziehen sich darauf, wie das System seine Leistungen erbringen soll. Sie werden in der Anforderungsspezifikation genau wie die funktionalen Anforderungen beschrieben. Allerdings unterliegen manche dieser Anforderungen einer gewissen Subjektivität und deren Umsetzung und die Abnahmekriterien lassen sich schwerer beschreiben. Bei den nicht funktionalen Anforderungen wird zwischen Qualitätsanforderungen und Rahmenbedingungen unterschieden. Die Qualitätsanforderungen werden dabei meistens an den **Softwarequalitätskriterien** ausgerichtet. Leider wird die Bedeutung der nicht funktionalen Qualitätsanforderungen meistens unterschätzt. Dabei spielen sie eine entscheidende Rolle hinsichtlich der Kundenzufriedenheit. Wenn diese Anforderungen nicht spezifiziert wurden, dann wird meistens zu spät bemerkt, dass das System zu umständlich zu bedienen ist oder die Performance nicht ausreicht.

Weitere wesentliche nicht funktionale Anforderungen werden durch die Rahmenbedingungen vorgegeben und müssen damit Beachtung finden. Diese Faktoren sind kaum beeinflussbar, schränken aber die möglichen Lösungen in den meisten Fällen ein.

Aufgabe 1: Fassen Sie die Erkenntnisse aus dem Infotext in der Tabelle zusammen!

<table>
<tr><th colspan="3">Anforderungen</th></tr>
<tr><td>Funktionale Anforderungen

was das System leisten soll</td><td colspan="2">Nicht funktionale Anforderungen

wie das System Leistung erbringen soll</td></tr>
<tr><td></td><td>Qualitätsanforderungen</td><td>Rahmenbedingungen</td></tr>
<tr><td></td><td>- Benuzbarkeit
- Zuverlässigkeit
- Effizienz
- Änderbarkeit
- Übertragbarkeit</td><td>- Technologisch
- Organisatorisch
- Normativ</td></tr>
</table>

Aufgabe 2: Erläutern Sie folgende nicht funktionale Qualitätsanforderungen und finden Sie zu jedem Punkt mindestens zwei konkrete Beispiele!

Nicht funktionale Qualitätsanforderungen	
Benutzbarkeit	➔ Anforderungen an die Benutzerfreundlichkeit der Software ➔ Berücksichtigung, wie gut der Umgang mit der Software zu erlernen ist. ➔ Ggf. Anfertigung einer Skizze der grafischen Oberfläche. **Testbereiche**: • Verständlichkeit: Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen • Erlernbarkeit: Aufwand für den Benutzer, die Anwendung zu erlernen • Bedienbarkeit: Aufwand für den Benutzer, die Anwendung zu bedienen
Zuverlässigkeit	-> Fehlertoleranz der Software -> Wiederherstellung alter Zustände nach Softwareversagen Testbereiche: - Häufigkeit des Versagens - Fehlertoleranz - Wiederherstellbarkeit
Effizienz	Zeit- und Verbrauchsverhalten der Software Testbereiche: - Antwort- und Verarbeitungszeiten - Verbrauchsverhalten: Anzahl und Umfang der benötigten Betriebsmittel

Änderbarkeit	Anpassung an neue Anforderungen Testbereiche: - Analysierbarkeit - Modifizierbarkeit - Stabilität - Prüfbarkeit
Übertragbarkeit	Übertragung der Software auf neue Umgebungen Testbereiche: - Anpassbarkeit - Installierbarkeit - Austauschbarkeit

Aufgabe 3: Nennen Sie Beispiele für nicht funktionale Rahmenbedingungen!

Rahmenbedingungen	Beispiele
Technologisch	Vorhandene, technische IT-Infrastruktur (Platformen, Schnittstellen, Nachbarsysteme usw.)
Organisatorisch	Aufbau und Prozessabläufe der Abteilungen, welche die Software nutzen sollen, unverrückbare Projekttermine
Normativ	Einzuhaltende Gesetze, Verordungen oder Normen, kulturelle aspekte: Sprachen, Gebräuche, Traditionen

Teste dein Wissen:

Bestimmen Sie, ob die nachfolgenden Rahmenbedingungen (1) organisatorisch, (2) technologisch oder (3) normativ sind.

2
1 / 3 b) Die Firma ist auch in Mittel- und Südamerika aktiv.
2
3
3
2
1 / 3 g) Die Firma ist in ganz Europa vertreten.
2
3
2
1
2
1

a) Der bisherige Quellcode liegt in Python vor.
c) Verbindung zu Auto CAD ist erforderlich.
d) Die Reifen-EU-Norm muss eingehalten werden.
e) Die DSGVO muss berücksichtigt werden.
f) Es sind nur Linux-Server vorhanden.
h) Die Importmöglichkeit von Worddokumenten besteht bereits.
i) Die Firma orientier sich an den Ferienregelungen der einzelnen Bundesländer.
j) Das Programm musss sich in die vorhandene Apple-Infrastruktur problemlos einfügen.
k) Die Software soll von zehn Abteilungen der Firma und dem Vertrieb benutzt werden.
l) Die OpenGL-Grafikschnittstelle ist Standard in der Firma.
m) Es besteht in der Firma eine IT-Abteilung, welche später die Wartung der SW übernehmen soll.

SO2022 – GA1 AE

Die Aufgaben 1 bis 4 beziehen sich auf die folgende Ausgangssituation:

Der Energieversorger Wind und Sonne AG möchte den Prozess zur Strom-Abrechnung weiter digitalisieren. Innerhalb dieses Projekts sollen mehrere Apps entwickelt werden. Dabei sollen Sie bei der Planung, Umsetzung und Einführung mitarbeiten und unterstützen.

Sie schlagen dem Vorstand des Energieversorgers vor, den Stromablesern zur Unterstützung einer effizienten Bearbeitung ihrer Tätigkeit eine mobile App zur Verfügung zu stellen.

Die benötigte App kann durch die hauseigenen IT-Abteilung oder durch einen externen Anbieter entwickelt werden. Dazu wird eine Entscheidungsgrundlage für den Vorstand des Energieversorgers benötigt.

a) Nenne Sie jeweils zwei Vor- und zwei Nachteile einer Fremdvergabe

_____

_____

_____

_____

_____

_____

b) Beschreiben Sie drei funktionale und drei nicht funktionale Anforderungen an die zu entwickelnde App.

_____

_____

_____

_____

_____

_____

_____

_____

Testverfahren

Testverfahren geben an, wie etwas getestet wird. Bei den Testverfahren wird grundsätzlich zwischen statischen und dynamischen Verfahren unterschieden.

Arbeitsauftrag 2:

1. Verschaffen Sie sich mithilfe des Infotextes einen Überblick über statische und dynamische Testverfahren!
2. Fassen Sie Ihre Erkenntnisse zusammen.
3. Beschreiben Sie die konkreten Testverfahren, sowie Vor- und Nachteile statischer und dynamische Verfahren.
4. Präsentieren Sie Ihre Ergebnisse in der Klasse.

Infotext

Statische Testverfahren dienen hauptsächlich dem Testen von Software, ohne diese auf einem Rechner auszuführen. Bei den Prüfobjekten handelt es sich meistens um Entwicklungsdokumente oder um den Quellcode selbst. Prinzipiell kann aber jedes Arbeitsergebnis der Softwareentwicklung durch statische Testtechniken überprüft werden, z.B. die Anforderungsspezifikation, die Testspezifikation oder das Benutzerhandbuch für den Kunden.

Zu den statischen Testverfahren gehören Inspektion, Review, Walkthrough und Audit. Diese Verfahren sind häufig Bestandteil formalisierter Qualitätsmanagementansätze im Projektalltag. Teammitglieder, externe Spezialisten oder auch Benutzer überprüfen dabei anhand einer aufgestellten Checkliste oder spontan die Korrektheit des erstellten Programms.

Testverfahren	**Beschreibung**
Inspektion	➔ der Programmcode wird nach genau festgelegter Vorgehensweise durch ein Gutachterteam untersucht ➔ ca. 50% bis 75% aller Entwurfsfehler können gefunden werden
Review	➔ weniger formal und weniger Aufwand (als durch Inspektion) ➔ ca. 60% bis 70% aller Fehler werden gefunden.
Walkthrough	➔ Unstrukturierte Vorgehensweise; ➔ der Entwickler präsentiert sein Programm und ein oder mehrere Gutachter stellen spontan Fragen.
Audit	➔ Strukturierte Vorgehensweise ➔ der Entwickler präsentiert sein Programm und der oder die Gutachter stellen Fragen nach einem vorher festgelegten Prüfkatalog.
Statische Code-Analyse	➔ beispielsweise bezüglich der Architektur, dem Design oder dem einheitlichen Einsatz der Programmiersprache und dem Einhalten von Namenskonventionen ➔ überwiegend werden die nicht funktionalen Aspekte des Programms beachtet. ➔ Überprüfung durch Werkzeugunterstützung.

Vor- und Nachteile der statischen Testverfahren:

Statische Testverfahren	
Vorteile	**Nachteile**

Dynamische Testverfahren dienen dem Auffinden von Fehlern durch Ausführen des Programms in einer Testumgebung. Die wichtigsten dynamischen Testverfahren sind der Whitebox und der Blackbox-Test.

Testverfahren	**Beschreibung**
Whitebox-Test	- Am häufigsten genutztes dynamisches Testverfahren - Vorr: Kenntnis des Quellcodes und der SW-Struktur - Verfahren: o Anweisungsüberdeckung o Zweigüberdeckung o Pfadüberdeckung o Einfache und mehrfache Bedingungsüberdeckung - Anwendung überwiegend bei Unit- und Komponententests.
Blackbox-Test	- Überprüfung des Verhaltens der Testobjekte, ohne Kenntnis über die SW-Struktur oder Quellcode. - Basis: Softwarespezifikation - Vergleich der realen Testergebnisse mit den erwarteten Ergebnissen - Relativ aufwendig (und nicht wirtschaftlich), alle Testfälle aus der Spezifikation abzuleiten, deshalb Minimierung der Anzahl durch: o Äquivalenzklassenbildung o Grenzwertanalyse

Vor- und Nachteile der dynamischen Testverfahren:

Vorteile	**Nachteile**

Funktionale Anforderungen was das System leisten soll	

Teststufen

Die geplanten Tests werden in einem Softwareprojekt in der Regel nicht zur gleichen Zeit durchgeführt, sondern sie erfolgen in Abhängigkeit vom **Entwicklungsstand der zu entwickelnden Software**. Dabei werden je nach Entwicklungsstand verschiedene Aspekte der Software getestet:

Test	Beschreibung
Modul-, Unit-, Komponententest	Test einer einzelnen Einheit oder Teile dieser Einheit auf Funktionalität
Integrationstest	Überprüfung des fehlerfreien Zusammenwirkens voneinander abhängiger Systemkomponenten; Testschwerpunkt sind die Schnittstellen zwischen den Komponenten
Systemtest	Abschließender Test des Gesamtsystems hinsichtlich der funktionellen und nicht funktionalen Anforderungen, welche zu erfüllen sind
Abnahmetest	Test des Gesamtsystems in der Kundenumgebung; wird in der Regel durch den Kunden selbst oder mit dessen Beteiligung durchgeführt; dieser Test bildet die Grundlage für die Entscheidung zur Abnahme des Projektes

Eine weitere Einteilung von Softwaretests richtet sich nach den Softwareversionen, welche den Entwicklungsstand der Software widerspiegeln. Solche Versionen sind beispielsweise **Alpha- und die Betaversion** einer Software oder der **Release Candidate**. Dementsprechend gibt es auch entsprechend benannte Tests wie den Alpha- oder den Betatest.

Alphatests werden meistens noch „intern“ von den jeweiligen Entwicklern und einem kleinen Kreis ausgewählter externer Tester durchgeführt, während die verschiedenen Betatest unter realitätsnahen Bedingungen durch mögliche spätere Benutzer durchgeführt werden.

Teste dein Wissen:

Aufgabe 1:

Gegeben ist die Methode `umlautParser(String input)` der Klasse `Umlaut` mit folgender Signatur:

```
public class Umlaut {
        public static String umlautParser(String input){
                // Inhalt unbekannt
        }
}
```

Die Methode bekommt als Eingabeparameter „input“ einen beliebigen String. Sie durchsucht den String nach den zeichen „ä“, „ö“, „ü“, „Ä“, „Ö“, „Ü“ und ersetzt sie jeweils durch „ae“, „oe“, „ue“, „Ae“, „Oe“, „Ue“. Wenn weitere Sonderzeichen im String vorkommen (also alle Zeichen außer 0-9, a-z, A-Z) werden diese entfernt.

a) Überlegen Sie sich mind. 5 Äquivalenzklassen, die Sie sinnvollerweise testen sollen.

_____

_____

_____

_____

_____

b) Geben Sie für jede Äquivalenzklasse wenigstens einen Eingabestring und das Soll-Ergebnis an.

_____

_____

_____

_____

_____

Aufgabe 3: (WI_21_22 AE: GA1)

Die Aufgaben 1 bis 4 beziehen sich auf die folgende Ausgangssituation:

Sie arbeiten im Projekt zur Entwicklung einer Software zur Terminvergabe in Arztpraxen mit. Dabei gehört es zu Ihren Aufgaben, bei der Planung Sicherheitsaspekte einzubeziehen sowie Teststrategien herauszuarbeiten.

Die Webapplikation für die Terminbuchung soll nun entwickelt werden.

a) Wesentliche Kriterien für die Akzeptanz einer Website sind Usability und positive User Experience.
aa) Verdeutlichen Sie anhand von vier verschiedenen Punkten, was darunter zu verstehen ist.

_____

_____

_____

ab) Die Benutzung soll außerdem barrierefrei möglich sein. Erläutern Sie den Begriff.

ac) Im Verlauf der Entwicklung soll die Güte der Usability periodisch überprüft und verbessert werden. Beschreiben Sie zwei Möglichkeiten zum Test der Benutzerfreundlichkeit.

Aufgabe 4: (WI_21_22 AE: GA1)

c) Die Geschäftsführung Ihres Unternehmens beauftragt Sie, sich über Möglichkeiten zur Qualitätssicherung von Software zu informieren und Vorschläge zu unterbreiten.

ca) Bei Ihrer Recherche finden Sie die ISO Normen 9126 und 25010, welche Qualitätsmerkmale von Softwareanwendungen beschreiben. Zwei der darin beschriebenen Merkmal sind „Effizienz“ und „Änderbarkeit“. Beschreiben Sie, was unter diesen beiden Qualitätsmerkmalen zu verstehen ist.

cb) Aus Ihrem Projektteam kommt der Vorschlag, zur Qualitätssicherung ausschließlich automatische Code-Reviews und Unit-Tests einzusetzen. Nehmen Sie Stellung zu diesem Vorschlag und begründen Sie fachlich Ihren Standpunkt.

Funktionale Anforderungen

was das Syste

d) Sie schlagen Ihrem Projektteam vor, zusätzlich für jeden Release einen Black-Box und White-Box-Test durchzuführen. Beschreiben Sie diese beiden Testarten.

Testdokumentation

Die einzelnen Testschritte und deren Ergebnisse müssen protokolliert und dokumentiert werden. Dazu hat das IEEE einen Standard veröffentlicht, nach dem man sich richten kann. Im IEEE 829 wurden der Inhalt und die Form von acht Basisdokumenten beschrieben. Dieser Standard wurde dann durch die ISO/IEC/IEEE 29119 ersetzt. Die Dokumente lassen sich in drei große Kategorien einteilen, welche hier nur kurz überblicksmäßig dargestellt werden.

Funktionale Anforderungen

was das System

Kategorie	Dokumente
Übersicht	• Testkonzept
Testspezifikation	• Testentwurfsspezifikation • Testfallspezifikation • Testablaufspezifikation
Testbericht	• Testobjektübergabebericht • Testprotokoll • Testabweichungsbericht • Testabschlussbericht

Testprotokoll Nr. 1

Datum: 03.05.20.. Tester: Karin Mäuser

Uhrzeit: 11:00 Uhr Aufsicht: Felix Schmidt

Verwendetes Testszenario: Testszenario 1

Eingabe eines Datums. Wenn das Datum korrekt ist, dann soll dieses angezeigt werden. Ansonsten wird auf die fehlerhafte Eingabe hingewiesen.

Beim Test bitte ausfüllen:

Nr.	Testdaten	Erwartete Ausgabe	Ergebnis	Bemerkungen
1	10.03.2020	10.03.2020	Ok	
2	29.02.1900	Datum inkorrekt	Fehler	Datum wird angezeigt
3	29.02.2000	29.02.2000	Ok	
4	abc	fehlerhafte Eingabe	Ok	
5	0.13.1000	Datum inkorrekt	Ok	
6	...	...	...	
...				

Anzahl der positiv verlaufenden Tests: 4

Anzahl der negativ verlaufenden Tests: 1

Notwendige Korrektur: Schaltjahre werden nicht richtig erkannt.

Karin Mäuser — Unterschrift des Testers

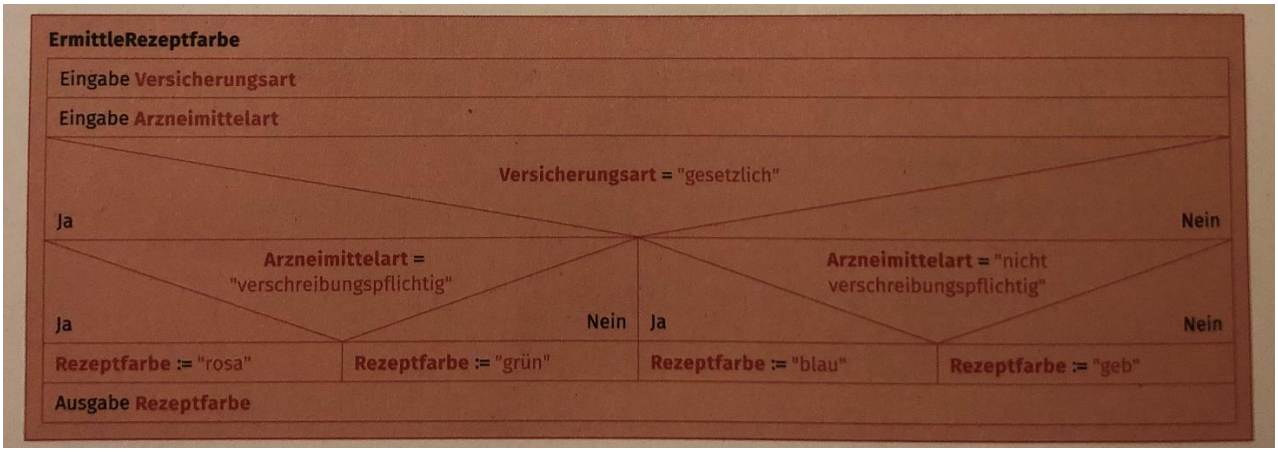
Felix Schmidt — Unterschrift der Aufsicht

Teste dein Wissen:

Rezepte haben im medizinischen Versorgungszentrum unterschiedliche Farben. Die Rezeptfarbe, welche gewählt wird, hängt von mehreren Parametern ab. Diese sind in der nachfolgenden Tabelle aufgeführt:

Rezeptfarbe	Beschreibung
Rosa	Gesetzlich versicherte Patienten Verschreibungspflichtiges Arzneimittel
Grün	Gesetzlich versicherte Patienten Nicht verschreibungspflichtiges Arzneimittel
Blau	Privat versicherte Patienten Verschreibungspflichtige und nicht verschreibungspflichtige Arzneimittel
Gelb	Gesetzlich oder privat versicherte Patienten Verschreibungspflichtige Arzneimittel, die unter das Betäubungsmittelgesetz fallen

Es liegt ein erster Entwurf für die Ermittlung der Rezeptfarbe als Struktogramm vor.



Von Ihnen soll ein Test des Algorithmus durchgeführt werden. Dazu liegt Ihnen folgende Tabelle mit Testdaten vor. Ermitteln Sie für alle Testdaten das zu erwartende Ergebnis laut Vorgabe und das tatsächliche Ergebnis aus dem Test. Tragen Sie die jeweilige Rezeptfarbe in die Tabelle ein.

Test-Nr.	Testdaten	Erwartetes Ergebnis	Testergebnis
1	Versicherungsart: „gesetzlich“ Arzneimittelart: „verschreibungspflichtig“	Rezeptfarbe:	Rezeptfarbe:
2	Versicherungsart: „gesetzlich“ Arzneimittelart: „betäubungsmittel“	Rezeptfarbe:	Rezeptfarbe:
3	Versicherungsart: „privat“ Arzneimittelart: „betäubungsmittel“	Rezeptfarbe:	Rezeptfarbe:
4	Versicherungsart: „gesetzlich“ Arzneimittelart: „nicht verschreibungspflichtig“	Rezeptfarbe:	Rezeptfarbe:
5	Versicherungsart: „privat“ Arzneimittelart: „verschreibungspflichtig“	Rezeptfarbe:	Rezeptfarbe:

Vergleichen Sie die erwarteten Ergebnisse mit den Testergebnissen und ziehen Sie eine Schlussfolgerung.