**Unit 1**

**1.1**

1. Nennen Sie die Aktivitäten im Qualitätsmanagement.

- *Qualitätsplanung: Erstellung und Dokumentation der Qualitätsanforderungen gemeinsam mit dem Auftraggeber;*

*- Qualitätslenkung: Überwachung, Steuerung und Kontrolle von Aktivitäten zur Qualitätsprüfung im Softwareentwicklungsprozess;*

*- Qualitätssicherung (kurz: QS): Tätigkeiten, die sicherstellen, dass festgelegte Qualitätsanforderungen für Produkte, Prozesse und Leistungen erfüllt werden;*

*- Qualitätsverbesserung: Auswertung von Produkt- und Prozessdaten zur Verbesserung des Qualitätsniveaus.*

2. Wie hängen Spezifikation und Qualität von Softwaresystemen voneinander ab?

*Der Begriff Softwarequalität wird in der DIN-ISO-Norm 9126 folgendermaßen definiert: „Softwarequalität ist die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte Erfordernisse zu erfüllen.“ Laut DIN kann über die Qualität von Software also nur auf Basis der Spezifikation („festgelegte Erfordernisse“) entschieden werden. In der Praxis zeigt sich allerdings, dass sich die wahrgenommene Qualität einer Software in erster Linie dadurch bestimmt, ob der Kunde feststellt, dass die Software seine tatsächlichen Anforderungen erfüllt.*

3. Grenzen Sie die Begriffe konstruktives Qualitätsmanagement und analytisches Qualitätsmanagement voneinander ab.

*Beim konstruktiven Qualitätsmanagement werden a priori (also vor der Erstellung) alle Qualitätseigenschaften für Produkte oder Prozesse definiert, um Fehler während der Softwareentwicklung zu vermeiden und die Qualität der erstellten Artefakte zu gewährleisten beziehungsweise zu erhöhen. Beim analytischen Qualitätsmanagement werden ex post (also nach der Erstellung) Maßnahmen zur Prüfung und Bewertung des aktuellen Qualitätsniveaus der Prüfobjekte durchgeführt, um Fehler systematisch aufzuspüren und ihre Ausmaße bestimmen zu können. Dazu werden sowohl statische als auch dynamische Testverfahren eingesetzt.*

**1.2**

1. Erläutern Sie das Prinzip der produkt- und prozessabhängigen Qualitätszielbestimmung.

*Das Prinzip der produkt- und prozessabhängigen Qualitätszielbestimmung bedeutet, dass für jedes konkrete Softwareprodukt beziehungsweise für den konkreten Softwareprozess jeweils dessen Qualitätsziele neu bestimmt werden sollen. Da die potenzielle Menge an Vorgehensweisen und Techniken zur Qualitätssicherung sehr groß ist, kann eine zielgerichtete Umsetzung der geforderten Eigenschaften nur dann in Angriff genommen werden, wenn es konkrete für das Projekt bzw. für den Softwareprozess formulierte Qualitätsziele gibt.*

2. Geben Sie ein Beispiel, wie das Prinzip der unabhängigen Qualitätssicherung in der Praxis umgesetzt werden kann.

*Beispiel 1:*

*Im Programmcode werden besonders kritische Codeabschnitte von einem Kollegen des Autors gereviewt.*

*Beispiel 2:*

*Ein anderes Team, das nicht an der Entwicklung des Systems beteiligt war, testet ein erstelltes System dahingehend, ob im Zusammenspiel aller Entwicklungsartefakte die fachlichen Anwendungsfälle erfolgreich umgesetzt wurden.*

3. Erläutern Sie das Prinzip der frühzeitigen Fehlerentdeckung und -behebung.

*Mit dem Prinzip der frühzeitigen Fehlerentdeckung und -behebung soll die Erkennung von Fehlern bereits vor deren Manifestation in Programmcode gewährleistet werden. Je später ein Fehler identifiziert wird, desto höher sind die Kosten der Fehlerbehebung. Um das zu verhindern, dürfen strukturierte QS-Maßnahmen nicht erst beim Testen der Software beginnen, sondern bereits mit der Erzeugung der ersten Artefakte in einem Softwareprozess. Die Behebung von Fehlern in frühen Phasen ist daher deutlich kostengünstiger und schneller zu erledigen, als Fehler, die sich vom fachlichen Anforderungsdokument bis in die Implementierung durchziehen.*

**1.3**

1.Erläutern Sie den Grundsatz: „Wiederholungen haben keine Wirksamkeit“.

*Die mehrfache Ausführung der gleichen Testfälle mit den gleichen Eingabedaten im gleichen Systemzustand und auf Basis des gleichen Datenbestands ohne Anpassungen am Programmcode erzielen keine Wirkung, die über eine einmalige Ausführung der Testfälle hinausgeht. Die mehrfache Ausführung des gleichen Tests unter identischen Bedingungen führt daher zu keinem Erkenntnisgewinn über eventuell vorliegende Fehler.*

2.Wie können Sie bei der Auswahl der zu testenden Systemkomponenten die Grundsätze „Vollständiges Testen ist nicht möglich“ und „Häufung von Fehlern“ nutzen?

*Die tatsächlich durchgeführten Softwaretests sind immer nur Stichproben. Die Entscheidungen, welche Funktionen mit welchem Aufwand und welchen Eingabeparametern getestet werden, müssen daher bewusst und sehr sorgfältig getroffen werden. Wird also ein Fehler an einer Stelle X im System durch Tests nachgewiesen, so ist die Wahrscheinlichkeit hoch, dass im Umfeld von X weitere Fehler vorliegen. Im Umfeld von bereits identifizierten Fehlern sollte daher besonders sorgfältig und gegebenenfalls mit mehr als den ursprünglich geplanten Testfällen getestet werden.*

3.In welchem Verhältnis steht die Definition des Begriffes „Softwarequalität“ nach DIN ISO 9126 zu dem Grundsatz „Trugschluss: Keine Fehler bedeuten nicht automatisch ein brauchbares System.“?

*Die Qualität von Software und damit auch die Formulierung von Testfällen erfolgt auf Basis der Spezifikation des Systems. Aspekte, die vielleicht für die Nutzer wichtig sind, jedoch bei der Spezifikation nicht berücksichtig wurden, werden in der Regel nicht getestet. Daher kann selbst ein fehlerfreies System – das also nach DIN eine sehr hohe Qualität hat, da alle „festgelegten Erfordernisse“ umgesetzt sind – aus Nutzersicht unbrauchbar sein.*

**1.4**

1.In welcher Beziehung steht das „Prinzip der produkt- und prozessabhängigen Qualitätszielbestimmung“ zur Festlegung der kostenoptimalen Qualität eines Softwaresystems.

*Für jedes Projekt bzw. für jeden Prozess müssen die Qualitätsziele neu bestimmt werden. Nur wenn es dokumentierte Qualitätsziele gibt, können diese später von den Architekten und Entwicklern bei der Implementierung gezielt berücksichtigt und adressiert werden. Der Aufwand zur Vermeidung von Fehlern ist abhängig von den formulierten Qualitätszielen: Je höher die Ziele, desto höher auch der Aufwand. Die erforderlichen Aufwände für die Fehlervermeidung beeinflussen die Bestimmung der optimalen Qualitätskosten.*

2.Aus welchen Kosten setzen sich die Qualitätskosten von Software zusammen?

*Die Qualitätskosten fassen die Aufwände zusammen, die für die Fehlervermeidung, -erkennung, -lokalisierung und -behebung aufgewendet werden.*

3.Welche Eigenschaften erfüllt die „kostenoptimale Qualität“ von Softwaresystemen?

*Bei einer kostenoptimalen Qualität sind die Kosten für Fehlervermeidung und Fehlerbehebung gleich.*

**Unit 2**

**2.1**

1. Nennen Sie Beispiele für Maßnahmen im prozessorientierten Qualitätsmanagement.

*Beispiele für Maßnahmen zum prozessorientierten Qualitätsmanagement sind der Einsatz eines standardisierten Entwicklungsprozesses, der Rollen und Aktivitäten in einem angemessenen Maß vorgibt, der verbindliche Einsatz von Werkzeugen zum Konfigurationsmanagement oder die Vorgabe von Eskalationspfaden im Fall einer Projekthavarie.*

2. Was wird im produktorientierten Qualitätsmanagement geprüft?

*Das produktorientierte Qualitätsmanagement überprüft Softwareprodukte und Zwischenergebnisse, die bei der Erstellung, Wartung und Weiterentwicklung der Software erzeugt werden, auf festgelegte Qualitätsmerkmale und die Erfüllung von Gütebedingungen.*

3. Grenzen Sie die Begriffe Qualitätslenkung und Qualitätsplanung voneinander ab.

*Die Qualitätsplanung für einen konkreten Softwareprozess legt genau fest, welche Personen für welche QS-Aktivitäten zuständig sind, zu welchem Zeitpunkt im Softwareprozess welche QS-Aktivitäten erforderlich sind, und mit welchen Mitteln, Methoden und Werkzeugen die Qualitätssicherung durchgeführt wird. Unter der Bezeichnung Qualitätslenkung werden hingegen alle Aktivitäten zusammengefasst, die für die Überwachung und Steuerung der QS-Aktivitäten erforderlich sind. Hierbei steht nicht das Softwaresystem im Fokus der Betrachtung, sondern die Aktivitäten, die zur Qualitätssicherung durchgeführt werden. Ziel der Qualitätslenkung ist eine angemessene Qualität der Qualitätssicherung.*

**2.2**

1. Wozu werden Qualitätsziele benötigt und wie können Qualitätsmodelle bei der Herleitung der Qualitätsziele helfen?

*Mit dem Festlegen und Priorisieren von Qualitätszielen wird projektweit eine Richtlinie erstellt, an der sich alle QS-Aktivitäten orientieren müssen. Um den abstrakten Begriff Qualität zu konkretisieren, können Qualitätsmodelle helfen. Die Grundidee eines Qualitätsmodells ist die Zergliederung eines abstrakteren Oberbegriffs „Qualität“ in verschiedene Unterbegriffe. Diese Unterbegriffe werden wiederum so lange untergliedert, bis eine konkrete messbare Qualitätseigenschaft formuliert werden kann.*

**2.3**

1. Beschreiben Sie das Ziel von Quality Gates bei der Qualitätssicherung.

*Ziel von Quality Gates ist es, zu einem bestimmten Zeitpunkt im Softwareprozess sicherzustellen, dass die erzeugten Artefakte vorher festgelegte Eigenschaften erfüllen. Häufig werden Quality Gates auch aus Gründen der Transparenz und Nachvollziehbarkeit der Qualitätsentwicklung eingesetzt. Ein Quality Gate kann passiert werden, wenn die dazu vorher, beispielsweise in Form einer Checkliste, festgelegten Kriterien erfüllt wurden. Neben der Qualität der erzeugten Artefakte wird mit Quality Gates häufig auch die Einhaltung von Vorgaben und Richtlinien an den SW-Prozess geprüft.*

**2.4**

1. Nennen Sie die Aktivitäten der Qualitätslenkung.

- *Kontinuierliche Beobachtung der Prozess- und Produktqualität,*

*- Reaktion auf Änderungen des Qualitätsgrades in Form von Anpassungen der QS-Vorgaben,*

*- Prüfung der tatsächlichen QS-Qualität.*

2. Was bedeutet Total Quality Management (TQM)?

*Total Quality Management (TQM) ist ein Konzept für eine grundsätzliche Herangehensweise an die Qualitätslenkung, das explizit alle Mitarbeiter einer Organisation bei der Qualitätslenkung miteinbezieht. Die Erreichung einer hohen Produktqualität ist beim TQM das höchste Ziel aller Mitarbeiter einer Organisation, wobei der Kunde über den erreichten Qualitätsgrad entscheidet. Dabei basiert TQM auf der Annahme, dass eine vollständige und kontinuierliche Konzentration auf eine hohe Qualität einen langfristigen Geschäftserfolg durch die Zufriedenheit der Kunden einstellt.*

3. Erläutern Sie das Prinzip „Primat der Qualität“ aus dem TQM.

*Das Prinzip „Primat der Qualität“ bedeutet, dass alle Prozesse genau so durchgeführt werden, wie sie definiert sind. Jede Person, die an der Ausführung des Prozesses beteiligt ist, erledigt ihre Aufgaben von Beginn an und wiederholt richtig. Nacharbeiten und Verschwendung von Ressourcen sollen vermieden werden. Werden Fehler identifiziert, muss deren Ursache ermittelt und beseitigt werden. Verbesserungen der Qualität werden durch die Verbesserung des festgelegten Softwareprozesses erreicht.*

**Unit 3**

**3.1**

1. Nennen Sie drei organisatorische Maßnahmen im konstruktiven Qualitätsmanagement.

*Beispiele für Methoden sind das Team Estimation (für die Priorisierung), die funktionale Äquivalenzklassenbildung (für die Testfallerstellung) und die objektorientierte Analyse (für das Requirements Engineering).*

2. Nennen Sie drei zwischenmenschliche Maßnahmen im konstruktiven Qualitätsmanagement.

*- Trainings oder Coachings*

*- Gemeinsame Aktivitäten wie z.B. regelmäßige gemeinsame Freizeitaktivitäten oder Teamevents*

*- Die Gestaltung des Arbeitsumfelds, beispiels­weise durch die Schaffung von ausgewiesenen Ruhezonen*

3. Nennen Sie drei technische Maßnahmen im konstruktiven Qualitätsmanagement.

*- Templates*

*- Richtlinien*

*- Checklisten*

**3.2**

1. Was ist die Kernidee von Timeboxing und wofür wird es eingesetzt?

*Die Kernidee von Timeboxing ist die Zuteilung von Zeit zu Aktivitäten bzw. Agendapunkten und die konsequente Überwachung und Einhaltung der vorgesehenen Zeit. Kann eine Aktivität bzw. ein Agendapunkt nicht in der vorgegebenen Zeit beendet werden, so wird sie bzw. er mit dem erreichten Stand abgebrochen und mit der nächsten Aktivität bzw. mit dem nächsten Agendapunkt begonnen. Timeboxing wird als Hilfsmittel für eine effiziente Durchführung von Meetings verwendet. Es hilft dabei, langwierige und ausufernde Diskussion zu verhindern, und durch die Vorgabe eines strengen Rahmens zu hoher Besprechungsqualität zu führen.*

2. Nennen und beschreiben Sie die sechs Schritte, die bei der Root-Cause-Analyse durchlaufen werden.

*1. Sammlung aller verfügbaren Informationen zum Fehler und dem durch den Fehler ausgelösten fehlerhaften Verhalten;*

*2. Anwenden der 5-Why-Methode;*

*3. Identifikation der Stellen im Programmcode, die verändert werden müssen, um den Fehler zu beseitigen;*

*4. Identifikation möglicher (konstruktiver und analytischer) QS-Maßnahmen, um den Fehler zukünftig zu vermeiden;*

*5. Einführung der QS-Maßnahmen;*

*6. Evaluation der QS-Maßnahmen, ob sie die Fehler zuverlässig verhindern können*.

3. Welche Aspekte sollten bei der Formulierung von Checklisten berücksichtigt werden?

*- Ein Punkt auf der Checkliste entspricht genau einem Ziel, zusammengesetzte Ziele sollten vermieden werden und auf mehrere Punkte aufgeteilt werden.*

*- Eine klare und einfache Formulierung der Checkpunkte erhöht die Lesbarkeit und die Verständlichkeit.*

*- Die zur Erfüllung eines Checkpunktes erforderlichen Kriterien müssen transparent sein, bei Bedarf werden zu einzelnen Checkpunkten Untercheckpunkte oder eigene Checklisten erstellt.*

**Unit 4**

**4.1**

1. Mit welchen statischen Verfahren können erstellte Testfälle qualitätsgesichert werden?

*Die erstellten Testfälle werden mit Review-Techniken dahingehend überprüft, ob sie für die Sicherstellung des für das aktuelle Projekt bestimmten Qualitätsgrades geeignet sind.*

2. Welche statischen Verfahren können im Umfeld von Modul- und Komponententests durchgeführt werden?

*Zwar sind Modul- und Komponententests keine statischen Verfahren, jedoch werden in dem Umfeld dieser Tests Maßnahmen zur statischen Codeanalyse durchgeführt. Hierbei wird der erzeugte Programmcode auf die Einhaltung geforderter Eigenschaften automatisch überprüft und Fehlerwahrscheinlichkeitsanalysen durchgeführt.*

3. Worin unterscheiden sich statische Verfahren zur Qualitätssicherung von dynamischen Verfahren?

*Im Gegensatz zu dynamischen Verfahren wird in statischen Verfahren der Prüfling nicht ausgeführt. Alle Verfahren der statischen Qualitätssicherungen können grundsätzlich ohne den Einsatz von Software, also nur „mit Papier und Stift“ durchgeführt werden, jedoch wird insbesondere das Messen und die Berechnung von Metriken durch Werkzeuge zur statischen Codeanalyse unterstützt.*

**4.2**

1. Welche Rollen werden für welche Review-Technik benötigt?

*Inspektion: Moderator, Gutachter, Autor (des Prüflings), Protokollführer*

*Walkthrough: Moderator, Gutachter, Autor (des Prüflings)*

*Stellungname: Gutachter, Autor (des Prüflings)*

**4.3**

1. Stellen Sie Vor- und Nachteile von Softwaremetriken gegenüber.

*Vorteile: Softwaremetriken sind relativ einfach zu ermitteln. Weiterhin sind die meisten Metriken unabhängig von der konkreten Programmiersprache oder der Projektorganisation und können daher in den meisten Projekten einfach eingesetzt werden.*

*Nachteile: Aussagen über die tatsächliche Konsequenz bestimmter Eigenschaften oder Kombinationen von Eigenschaften insbesondere mit Softwaremetriken sind oft nicht möglich. Auf Basis von Softwarevermessungen lassen sich in der Regel keine gesicherten Schlussfolgerungen zur tatsächlichen Qualität des Systems ableiten.*

2. Nennen Sie mindestens je zwei Metriken zur strukturellen Komplexität von Komponenten und zu objektorientierten Systemen.

*Metriken zur strukturellen Komplexität:*

*- Fan-in*

*-Fan-out*

*Metriken für objektorientierte Systeme: Beispiele für typische Metriken für objektorientierte Systeme sind unter anderem:*

*- Grad von Vererbungsbäumen*

*- Tiefe einer Klasse (DIT)*

*- Kopplung zwischen Objekten (CBO)*

*- Response for a class (RFC)*

3. Was sind Metriken und wozu werden sie eingesetzt?

*Metriken sind Werte von Kenngrößen, die mit Hilfe von Werkzeugen bestimmt und analysiert werden. Auf Grundlage der gemessenen Werte werden Aussagen über den aktuellen Zustand des Produktes oder des Prozesses abgeleitet.*

**4.4**

1. Nennen Sie zwei konkrete Beispiele von Fehlermustern, die mit der statischen Codeanalyse identifiziert werden können.

*- Mögliche Bugs durch leere try/catch- oder switch-Blöcke;*

*- Erstellte, aber nicht genutzte Variablen, Parameter und private Methoden;*

*- Überflüssiger Einsatz von String- und StringBuffer-Methoden;*

*- Identifikation unnötiger oder zu komplizierter if-Statements;*

*- Codestellen, die durch Copy & Paste dupliziert wurden;*

*- Implementieren der equals()-Methode ohne die hashCode()-Methode oder ein Stringvergleich mit == statt mit equals();*

*- Die falsche Verwendung der Inkrementfunktion (i++) bei return-Statements oder das Hinzufügen eines Collection-Objektes zu sich selber;*

*- Speichern eines Wertes in einer lokalen Variable, die niemals gelesen wird.*

2. Nennen Sie drei Vorteile des Einsatzes von Werkzeugen zur automatischen Stilanalyse.

*- Automatisches Feedback an den Entwickler, ob er sich an festgelegte Konventionen gehalten hat. Beispiele hierfür sind die richtige Verwendung von Annotationen, die Einhaltung von Namenskonventionen für Klassen, Attribute, Methoden und Variablen, Prüfung auf das Vorliegen von Javadoc-Kommentaren.*

*- Bei manuellen Codereviews oder der Entwicklung im Team, bei der mehrere Entwickler an einer Komponente arbeiten, kann mit der automatischen Stilanalyse die Einarbeitungszeit und das Codeverständnis gefördert werden.*

*- Darüber hinaus erleichtert ein einheitlich formatierter Programmcode die Arbeit mit Versionskontrollsystemen. Bei einer einheitlichen Formatierung werden in der Änderungsverfolgung weniger Formatierungsänderungen, beispielsweise Zeileneinrückungen, angezeigt.*

3. Was bedeutet statische Codeanalyse und wie unterscheidet sie sich von Metriken?

*Verfahren zur statischen Codeanalyse werden zur qualitativen Bewertung von Programmcode eingesetzt. Im Unterschied zu Metriken werden bei der statischen Codeanalyse keine Eigenschaften des Programmcodes gemessen, sondern der Code wird inhaltlich analysiert und ausgewertet.*

**Unit 5**

**5.1**

1. Worin unterscheiden sich Pfadtest und Anweisungsüberdeckung?

*Anweisungsüberdeckung: Jede relevante Anweisung/Funktion wird beim Test mindestens 1 x aufgerufen. Hierzu müssen Testdaten erstellt werden, mit denen jede Anweisung im Code (White Box-Test) mindestens 1 x ausgeführt wird bzw. mit denen jede Fachfunktion (Black Box-Test) des unterstützten Geschäftsprozesses mindestens 1 x ausgeführt wird. Eventuell auftretende Schleifen werden nicht explizit mit berücksichtigt.*

*Pfadtest: Erstellen von Testdaten, sodass der gesamte Kontrollflusspfad innerhalb einer Komponente im Programmcode (White Box-Test) oder ein Pfad durch den Geschäftsprozess (Black Box-Test) von Anfang bis Ende durchlaufen werden kann. Dabei muss sichergestellt sein, dass dabei auch der mehrfache Durchlauf von Schleifen getestet wird.*

2. Worin unterscheiden sich Black-Box-Test und White-Box-Tests und wozu werden diese jeweils eingesetzt?

*Bei Black-Box-Tests liegen über die innere Struktur des Prüflings keine Informationen vor. Nur das nach außen hin sichtbare Verhalten ist relevant. Der Tester liest und interpretiert die Spezifikation. Dann erstellt er anhand seiner Interpretation der Spezifikation die Testfälle und führt diese aus. Anschließend beurteilt der Tester die Testergebnisse anhand seiner Interpretation der Spezifikation. White-Box-Tests werden mit Kenntnis des Programmcodes erstellt, der im Gegensatz zu Black-Box-Tests bekannt ist.*

3. Was ist ein Testfall und aus welchen Elementen besteht er?

*Ein Testfall dient als Handlungsanweisung zur Durchführung von Softwaretests und besteht mindestens aus den folgenden Elementen:*

*- Vorbedingungen, die vor dem Prüfschritt sichergestellt werden müssen;*

*- Testdaten bzw. Testaktionen, die während des Prüfschrittes eingegeben bzw. durchgeführt werden;*

*- Nachbedingungen, deren Erfüllung nach dem Prüfschritt geprüft werden; sowie*

*- beschreibende Daten zum Testfall, wie Name, ID, betreffende Komponente, Funktion oder Eigenschaft, die geprüft wird, Anwendungsfall oder Name des Erstellers.*

**5.2**

1. Nennen und beschreiben Sie vier mögliche Überdeckungskriterien für Use Cases.

*- Anweisungsüberdeckung: Jede Funktion wird mind. 1 x aufgerufen.*

*- Zweigüberdeckung: Jeder Kontrollflusspfeil wird mind. 1 x durchlaufen.*

*- Bedingungsüberdeckung: Jede Bedingung wird mind. 1 x zu WAHR und 1 x zu FALSCH ausgewertet.*

*- Pfadtest: Jeder mögliche Pfad durch den Use Case wird vollständig („am Stück“) durchlaufen, dabei werden mögliche Schleifen berücksichtigt.*

2. Nennen und beschreiben Sie die Kriterien, anhand derer der Detailgrad der Testfälle für Anwendungsfälle bestimmt werden kann.

*- Wertbeitrag: Wie hoch ist der Beitrag dieser Funktion zur Wertschöpfungskette?*

*- Nutzungshäufigkeit: Wie viele Nutzer greifen wie häufig auf die Funktion zu?*

*- Schadenspotential: Welcher Schaden kann durch nicht identifizierte Fehler in der Funktion entstehen?*

*- Typische Fehler: Welche Funktionen bergen häufig Fehler in der Umsetzung? Wo sind die typischen Schwachstellen bei vergleichbaren Alt- oder Konkurrenzsystemen?*

*- Geforderte Testabdeckung: Welcher Grad der Testabdeckung ist für die fachliche Ebene tatsächlich gefordert?*

**5.3**

X

**5.4**

1. Nennen und beschreiben Sie kurz die Schritte zur zustandsbasierten Testfallerstellung.

*Schritt 1: Konstruktion einer Zustandsübergangstabelle*

*Zur einfachen Identifikation von erlaubten und nicht erlaubten Transitionen wird das Zustandsdiagramm in eine Zustandsübergangstabelle überführt. Pro definiertem Zustand im Diagrammm wird jeweils eine Spalte und eine Zeile angelegt.*

*Schritt 2: Ausfüllen der Tabelle*

*Gültige Kombinationen aus Zuständen und Transitionen im Diagramm führen zu einem Folgezustand. Der Name einer Transition wird jeweils genau in die Zelle notiert, deren Ausgangszustand durch die Spalte und deren Zielzustand durch die Zeile vorgegeben wird.*

*Schritt 3: Fehlerzustände in Tabelle eintragen*

*Je nach Größe der Zustandsübergangstabelle und dem Einsatzgebiet des zu testenden Systems können die nicht erlaubten Transitionen (mit „-“ in der Tabelle markiert) in die Kategorien Fehlersituation bzw. nicht erlaubt, jedoch triviale Transition.*

*Schritt 4: Testfälle aus Tabelle ableiten*

*Aus der Zustandsübergangstabelle mit gekennzeichneten Fehlerzuständen werden Testfälle abgeleitet. Ergebnis der Testfallerstellung ist eine Menge aus Folgen von Transitionen, mit denen die vom System zu implementierenden Funktionen als auch die vom System nicht zu unterstützenden Transitionen getestet werden können.*

**5.5**

1. Nennen Sie zwei Szenarien für den Einsatz von Zufallstestdaten.

*Einfache Zufallsdaten: Wenn die Wertebereiche zu Eingabeparametern bekannt sind, beispielsweise mit der Äquivalenzklassenbildung ermittelt, kann die Auswahl der konkreten Eingabewerte zufällig erfolgen.*

*Komplexe Zufallstestdaten: Eine vorgegebene, nicht-zufällig erzeugte Menge von Testdaten wird nach einem festgelegten Prinzip zufällig kombiniert.*

2. Nennen Sie zwei Vor- und zwei Nachteile von Zufallstestdaten.

*Vorteile: In Kombination mit der Äquivalenzklassenbildung ist die Erstellung von Zufallstestdaten eine Technik, bei der mit vergleichbar geringem Aufwand viele verschiedene Testdatensätze generiert werden können.*

*• Sobald ein geeigneter Testdatengenerator existiert, können in der Regel beliebig viele Testdatensätze erzeugt werden.*

*• Dabei wird sichergestellt, dass jeder Testdatensatz die geforderten Eigenschaften hat und menschliche Fehler durch eine manuelle Testdatenerzeugung ausgeschlossen werden können.*

*Nachteile: Die zufällige Erstellung von Testdaten erzeugt zwar viele verschiedene, jedoch nicht in jedem Fall wirklich realistische Daten.*

*• Es müssen für die Erstellung des Testdatengenerators die Anforderungen an die Struktur und den Inhalt der Testdaten detailliert spezifiziert werden.*

*• Spezifikation und Implementierung von Testdatengeneratoren erfordert zusätzliche Ressourcen. Ob die dazu erforderlichen Aufwände in einem sinnvollen Verhältnis zum Nutzen stehen,muss für jedes Projekt individuell entschieden werden.*

3. Beschreiben Sie kurz das Grundprinzip bei der Erstellung von Zufallstestdaten.

*Das Grundprinzip der Testdatenerstellung basiert darauf, dass im Rahmen eines bestimmten Wertebereichs von Eingabeparametern gültige Werte zufällig erzeugt werden.*

**Unit 6**

**6.1**

1. Nennen Sie fünf typische Quellen für die Erstellung von Testdaten.

*Neben der Spezifikation und Architekturbeschreibung eignen sich auch der Programmcode des Systems sowie bestehende Produktionsdaten oder der existierende Testdatenbestand.*

2. Nennen Sie die Aktivitäten zum methodischen Testen.

*- Testanforderungsanalyse*

*- Testplanung*

*- Testfallspezifikation*

*- Testdatenerstellung*

*- Testausführung*

*- Testauswertung*

**6.2**

1. Erläutern Sie kurz, was ein Modultest ist.

*Das isolierte Prüfen von einzelnen Softwarebestandteilen wird Komponententest genannt. Jede Komponente wird separat implementiert und nach Fertigstellung in das System eingebracht (auch: integriert). Während bzw. nach der Fertigstellung einer Softwarekomponente kann diese bereits mit Modultests auf die Einhaltung von spezifizierten Vorgaben überprüft werden.*

2. Erläutern Sie kurz das Vorgehen beim Test Driven Development.

*Die Aktivitäten „Testfall erstellen“ und „Programmcode erzeugen“ werden in genau der umgekehrten Reihenfolge als bei der „klassischen“ Testfallerstellung durchgeführt Die Erstellung von Unit-Tests erfolgt, bevor der eigentliche Programmcode zur Implementierung der Funktion programmiert wird. Nachdem die Notwendigkeit einer neuen Funktion in einer Klasse oder einer Komponente identifiziert wurde, werden zunächst Unit-Tests erstellt, mit denen die neu zu erstellende Funktion getestet werden soll. Werden diese Unit-Tests ausgeführt, müssen sie fehlschlagen, da die relevanten Systemfunktionen noch nicht umgesetzt werden. Mit ihrer Umsetzung wird erst nach der Testfallerstellung begonnen. Die Systemfunktionen werden nun schrittweise implementiert und bereits während der Implementierung getestet.*

3. Erläutern Sie kurz, wie bei der Durchführung von Unit-Tests deren Unabhängigkeit sichergestellt wird.

*Jeder Test muss individuell dafür sorgen, dass die von ihm benötigten Vorbedingungen hergestellt werden. Dazu zählt beispielsweise auch, dass bei der Durchführung von Tests zuerst der erforderliche Datenbestand in die Datenbank eingespielt wird, auf dem dann der Test durchgeführt wird. Wird anschließend ein weiterer Test durchgeführt, müssen alle Änderungen am Datenmodell und dem internen Systemzustand wieder rückgängig gemacht bzw. bei der Ausführung eines neuen Tests der benötigte Systemzustand wiederhergestellt werden.*

**6.3**

1. Erläutern Sie kurz, was ein Integrationstest ist.

*Sobald mindestens zwei Softwarekomponenten fertiggestellt wurden, können sie zu einem System zusammengeführt, das heißt integriert, werden. Während bzw. nach der Integration wird in Integrationstests das Zusammenspiel von Gruppen von Komponenten getestet. Die Integrationstests prüfen, ob die Komponenten so zusammenarbeiten, wie es in der Spezifikation beschrieben wurde.*

2. Erläutern Sie die Nachteile der Bottom-up-Strategie und beschreiben Sie, wie diese mit der By-Value-Strategie vermieden werden können.

*Die Bottom-up-Strategie hat als Nachteil, dass den Anwendern erst am Ende der Integration das System präsentiert werden kann und damit ein wesentliches Projektrisiko nicht frühzeitig adressiert werden kann. Weiterhin können keine gezielt falschen Rückgabewerte von aufgerufenen Komponenten getestet werden, da bei der Bottom-up-Strategie keine Dummies zum Einsatz kommen.*

*Die By-Value genannte Integrationsstrategie bietet dem Projektteam die Möglichkeit je nach aktuellem Bedarf im Projekt mit der Erstellung und Integration der Komponenten zu beginnen, mit denen der größte Wertbeitrag geschaffen werden kann. So kann beispielsweise gleichzeitig mit der GUI und kritischen Basiskomponenten begonnen werden, wenn dort das größte Projektrisiko vermutet wird. Falls zum Testen des Zusammenspiels von Komponenten sowohl Treiber als auch Dummies benötigt werden, können diese bei Bedarf erstellt und eingesetzt werden.*

3. Erläutern Sie, was unter den Begriffen Dummy und Treiber verstanden wird und wie diese beim Integrationstest verwendet werden.

*Insbesondere bei komplexen Softwaresystemen mit sehr vielen Komponenten sind nicht alle Komponenten zur gleichen Zeit fertig. In diesen Fällen werden fehlende Komponenten durch sogenannte Treiber und Dummies simuliert, damit bereits mit dem Test begonnen werden kann. Als Treiber werden dabei Softwarefragmente bezeichnet, die das Aufrufen anderer Komponenten simulieren. Dummies hingegen simulieren Komponenten, die durch andere Komponenten aufgerufen werden. Insbesondere technische Schnittstellen zu externen Systemen werden im Integrationstest durch Dummies und Treiber simuliert. Denn erst das fertige Softwaresystem kann an externe Systeme angeschlossen werden, das Zusammenspiel der Schnittstellen muss jedoch bereits vorher getestet werden.*

**6.4**

1. Beschreiben Sie kurz, was ein Systemtest ist.

*Nach Abschluss der Entwicklungsarbeiten und der Integration aller Softwarebestandteile zu einem fertigen System wird mit dem sogenannten Systemtest das System als Ganzes getestet. Ziel des Systemtests ist die Überprüfung, ob das System als Ganzes die spezifizierten Anforderungen erfüllt. Neben Funktionstests werden dabei auch Lasttests und Stresstests durchgeführt.*

2. Beschreiben Sie, wozu ein Performancetest durchgeführt und was dabei geprüft wird.

*Mithilfe von Performancetests wird das Verhalten des Systems unter Belastung geprüft und dabei ermittelt, ob es die spezifizierten Qualitätseigenschaften unterstützt. Mit Ergebnissen des Performancetests können konkrete Aussagen über das Systemverhalten des erstellten Systems in seiner zukünftigen Systemumgebung gemacht werden.*

*Auf Basis der Testergebnisse kann auch überprüft werden, ob die vom IT-Betrieb zur Verfügung gestellte IT-Infrastruktur, wie Speicherplatz, Arbeitsspeicher, Rechenzeit oder Netzwerkanbindung ausreichen. Performancetests müssen daher sehr sorgfältig vorbereitet werden. Typische Indikatoren für die Leistungsfähigkeit von Softwaresystemen sind unter anderem Latenz, Durchsatz und Transaktionsraten. Der Performancetest dient auch zur Bestimmung bzw. zur Prüfung der Servicelevel, die ein Systembetreiber im Rahmen von Serviceverträgen in den sogenannten Service Level Agreements (SLA) zugesichert hat.*

3. Erläutern Sie, warum die Regressionsfähigkeit eine wichtige Eigenschaft für Systemtests ist.

*Die Eigenschaft „Regressionsfähigkeit von Testfällen“ bezeichnet die Fähigkeit zur wiederholten Durchführung des Tests. Mit der Durchführung von Regressionstests wird sichergestellt, dass durch Anpassungen am Programmcode bereits bestehende Funktionen nicht versehentlich beeinflusst werden. Auch falls im Projekt nicht evolutionär entwickelt wird, müssen bei jedem neuen Release der Software auch die bereits bestehenden Systemtestfälle durchgeführt werden, auch wenn die Release-Zyklen mehrere Monate oder Jahre dauern.*

**6.5**

1. Wodurch unterscheiden sich Systemtest und Abnahmetest?

*Der Auftraggeber führt den Abnahmetest selber durch oder wird in der Durchführung miteinbezogen. Von der Art der Durchführung ist der Abnahmetest eine spezielle Art des Systemtests, jedoch mit einer geringeren Testabdeckung. Der wichtigste Unterschied gegenüber dem Systemtest ist jedoch die Durchführung durch eine andere Organisation als die, die das System entwickelt hat. Ein erfolgreicher Abnahmetest ist häufig die vertraglich vereinbarte Voraussetzung für die Rechnungslegung.*

2. Warum wird das Ergebnis des Abnahmetests in der Praxis oft verhandelt?

*Im Unterschied zu den anderen Teststufen steht am Ende des Abnahmetests die Entscheidung: Annahme oder Ablehnung. In der Praxis ist es dabei sehr unwahrscheinlich, dass während der Abnahmetests keine Fehler mehr identifiziert werden. Daher wird nach der Durchführung des Abnahmetests das Testergebnis in der Regel verhandelt, welche Fehler zur endgültigen Abnahme noch behoben werden müssen. Erst wenn diese Fehler behoben wurden, wird offiziell die Abnahme erteilt.*

3. Beschreiben Sie kurz, was ein Abnahmetest ist.

*Die letzte Teststufe ist der Abnahmetest (auch: Akzeptanztest), bei dem das fertige System bereits beim Auftraggeber installiert ist und unter den tatsächlichen Betriebsbedingungen getestet wird. Beim Abnahmetest wird geprüft, ob das System aus Kundensicht die vertraglich vereinbarten Leistungsmerkmale aufweist.*

**Unit 7**

**7.1**

1. Nennen Sie die Aktivitäten zur Qualitätssicherung von Anforderungen.

*- Prüfkriterien festlegen*

*- Prüfprinzipien und Prüftechniken auswählen*

*- Prüfung durchführen und Ergebnisse dokumentieren*

*- Abstimmen der Anforderungen/Konfliktmanagement*

2. Nennen Sie drei Qualitätsaspekte von Anforderungen.

*Inhalt*

*Dokumentation*

*Abgestimmtheit*

3. Grenzen Sie die Begriffe Prüfprinzipien und Prüftechniken voneinander ab.

*Im Rahmen der Prüfung von Anforderungen können verschiedene Prüfprinzipien unterschieden werden, die zwar selber noch keine Prüftechnik darstellen, jedoch bei der Auswahl der Prüftechniken helfen. Prüftechniken hingegen sind konkrete Verfahren oder Vorgehensweisen, die zur Durchführung der Prüfung eingesetzt bzw. genutzt werden. Eine Prüftechnik kann ein oder mehrere Prüfprinzipien unterstützen.*

**7.2**

1. Nennen und beschreiben Sie die Aktivitäten von ATAM und ordnen Sie diese den einzelnen Phasen von ATAM zu.

*Phase 1: Vorbereitung und Präsentation*

*In Schritt 1 wird allen Beteiligten die Vorgehensweise ATAM vorgestellt und erläutert, aus welchen Aktivitäten diese besteht.*

*In Schritt 2 werden die wichtigsten funktionalen Anforderungen, Qualitätsanforderungen und Randbedingungen vorgestellt.*

*In Schritt 3 erfolgt die Vorstellung der durch die Architekten erstellten Basisarchitektur.*

*Phase 2: Untersuchung und Analyse*

*In Schritt 4 werden auf Grundlage der Basisarchitektur verschiedene Architekturvarianten erstellt, mit denen die vom System geforderten Funktionen und Eigenschaften erfüllt werden können.*

*In Schritt 5 werden Anwendungsszenarien erstellt und priorisiert, mit denen die konkret geforderten Qualitätseigenschaften beschrieben werden.*

*In Schritt 6 von ATAM werden die erstellten Architekturvarianten auf Erfüllung der in Schritt 5 hergeleiteten Szenarien grob geprüft.*

*Phase 3: Test*

*In Schritt 7 wird die bereits erstellte Menge von Szenarien verfeinert, erweitert und priorisiert.*

*In Schritt 8 erfolgt die detaillierte Analyse der Architekturvarianten.*

*Phase 4: Berichterstattung*

*In Schritt 9 findet die Berichterstattung gegenüber den beteiligten Stakeholdern statt.*

2. Was sind die Ziele der QS von Architekturen nach der Implementierung?

*Mit der Prüfung der Architektur begleitend zur Implementierung und/oder nach Abschluss der Implementierung soll sichergestellt werden, dass sich das Ergebnis der Entwicklungsarbeiten an die durch die Architekturdefinition aufgestellten Vorgaben hält. Diese Aktivitäten sind in etwa vergleichbar mit denen einer Bauüberwachung auf einer Baustelle: Dort müssen die einzelnen Gewerke koordiniert und deren Zwischen- und Endergebnisse überwacht und dahingehend kontrolliert werden, ob sie in Art und Qualität dem Bauplan entsprechen. So muss auch bei der Erstellung eines Softwaresystems der Architekt sicherstellen, dass die vom Architekturteam getroffenen Entscheidungen auch tatsächlich im Programmcode umgesetzt werden.*

3. Was sind die Ziele der QS von Architekturen vor der Implementierung?

*Zum einen kann die Eignung einer konzipierten Architektur vor der Implementierung (ex ante) und auf Basis der Architekturbeschreibung geprüft werden. Zum anderen kann begleitend zur Implementierung und nach Ende der Implementierung (ex post) geprüft werden, ob die tatsächlich erstellte Architektur auch die Vorgaben der vor der Implementierung erstellten Architekturbeschreibung hält. Diese beiden Prüfziele sind so unterschiedlich, dass für jede dieser Prüfungen verschiedene Prüftechniken eingesetzt werden.*

**7.3**

1. Nennen Sie fünf Qualitätsattribute für Softwareprozesse.

*- Verständlichkeit*

*- Standardisierung*

*- Sichtbarkeit*

*- Messbarkeit*

*- Unterstützbarkeit*

*- Akzeptanz*

*- Zuverlässigkeit*

*- Stabilität*

*- Wartungsfreundlichkeit*

*- Schnelligkeit*

2. Welche Voraussetzungen müssen sichergestellt werden, bevor Änderungen im Softwareprozess aktiv umgesetzt werden können?

*- Die Änderungen im Softwareprozess müssen allen Betroffenen bekannt sein.*

*- Ggf. benötigte neue fachliche Kompetenzen müssen erlangt werden.*

*- Alle angepassten Artefakte wie Vorlagen, Dokumentationen, Werkzeuge und Vorgehensweisen müssen verfügbar sein.*

3. Beschreiben Sie, was die Stufen des CMMI-Modells sind und wozu diese eingesetzt werden.

*Das CMMI-Modell ist als Rahmenwerk für Prozessverbesserungen konzipiert. Es enthält dazu unter anderem ein Stufenmodell zur Beurteilung des Reifegrades von Software- und Managementprozessen. Der aktuelle Fähigkeitsgrad zu einem Prozess wird dabei auf einer 5-stufigen Skala bestimmt. Die Bestimmung des Reifegrades basiert auf der Analyse der aktuellen IST-Prozesse eines Unternehmens. CMMI-Grad 1 ist dabei die niedrigste Stufe, Grad 5 entspricht der höchsten Fähigkeitsstufe. Die CMMI-Reifegrade werden zur Bewertung einer Organisation bzw. der Prozessreife von Softwareprozessen von Organisationen eingesetzt werden.*