FRAGE 1 (4):
a) Wer oder was ist die Software-Triade? b) Welche Funktion haben die "Spieler" auf dem Markt? c) Welche Auswirkungen hat das für Österreich?

a) Aufteilung des Softwaremarktes in drei Teile

b)
USA: Basissysteme Systemsoftware
Japan: Software für Endgerate - Embedded Systems
Europa: Spezialisierte Anwendungslösungen - Hauptanwender

c)
Österreich
als Land in Europa ist auch spezialisiert auf Speziallösungen und professionelle Services. Schwerpunkt setzen auf Qualität.

FRAGE 2 (4):
a) Warum ist Software ein interessanter Wirtschaftszweig besonders für Österreich?
b) gilt das besonders für Kleinbetriebe?

a)
- Geringe Investitionen
- Geringe Umweltbelastung
- interessant
- herausfordernd
- "nur" Können und Intelligenz
- die Schnellen fressen die Langsamen und nicht die Grossen die Kleinen.

b)
Die Zukunft gehört dem Betrieb der klein genug ist, um flexibel und unbürokratisch zu bleiben und groß genug um sich ein gutes Management leisten zu können.

FRAGE 3 (4):
a) Was meint Fred Brooks wenn er sagt, es gibt keine Silberkugeln. Womit begründet er die Probleme der Software?
b) Was folgern Sie daraus für die Software-Entwicklung?

a) Mit Silberkugeln beschreibt er den Umstand dass es keine Allgemeinlösung gegen fehlerhafte Software schlechtes Design gibt.
b) Softwareentwicklung ist ein intellektueller Vorgang, und kann nicht automatisiert werden. Menschliche Erfahrung ist hier sehr wichtig.

FRAGE 4 (8): Nehmen Sie an, Sie entwickeln für ein Software-Haus ein Entwicklungshandbuch unter der Verwendung der in der Vorlesung definierten Entwicklungsniveaus! Geben Sie für jedes Entwicklungsniveau
a) eine kurze Beschreibung (2 Zeilen) und

b) erfinden Sie pro Entwicklungsniveau die Namen für zwei typische, darin erzeugte Zwischenprodukte/Zwischendokumente!
(Skriptum 2-8)

a), soweit Möglich auch b) (links daneben steht der Teil aus dem UPEDU / RUP)
Unternehmensmodell: zuerst Daten- und Funktionsstruktur eines Unternehmens sammeln. Bilden Basis für nötige Software Anwendungen. Bebauungsplan
Anforderungen: komplexe Systeme, schwierig zu spezifizieren → Bedürfnisse der Benutzer festlegen, bevor Entscheidungen über das gewünschte System fallen. Anforderungsprofil
Spezifikation: berücksichtigt Benutzersicht und nicht primär die technischen Gegebenheiten. Benutzerseitig sichtbares Verhalten des Systems festlegen. Intensives Mitwirken der Fachabteilung und Endbenutzer. Architektenplan, Bauplan.
Entwurf (auch Implementierung (in UPEDU: IMPLEMENTATION)): Detail-Entwurf: zuerst Essenz des zu erstellenden Codes festlegen, dann technische Details fixieren. Grob-Entwurf: technische Lösung im Großen festlegen. Baumeisterarbeit
Installation: Installation und Feintuning der Software. Update von altem auf neues System. Einrichtung
Integration: wachsende Komplexität (besonders bei mehreren Versionen bzw. Konfigurationen) → mehr Aufmerksamkeit für Zusammenbau. Umfeld

FRAGE 5 (6):
a) Warum gibt es Entwicklungsniveaus (in UPEDU / RUP auch Disciplines genannt)?
b) Beschreiben und erklären Sie die grundlegenden Entwicklungsniveaus (siehe Frage 4a), was sind generell ihre Bedeutung im Entwicklungsprozess?

a) aufgrund von qualitativen Unzulänglichkeiten im Software Entwicklungsprozess. Ausgehend von der Implementierung (Code) wurden allmählich Entwicklungsniveaus vor- bzw. nachgelagert.

b) siehe auch 4a). Entwicklungsniveaus stellen verschiedene Abstraktionen des Endproduktes dar, jedes Entwicklungsniveau stellt andere Überlegungen in den Vordergrund.

FRAGE 6 (4):
a) Was sind die Entwicklungsaspekte im Vorgehensmodell.
b) Beschreiben Sie die bekanntesten Entwicklungsaspekte!
a)
Produktorientiert:
- Funktionssicht
- Datensicht
- Datenfluss
- Benutzerschnittstelle

begleitende Prozesse:
- Dokumentation
- Konfigurationsmanagement
- Qualitätssicherung
- Projektmanagement

Produktmanagement
Kundenmanagement
b) blabla

FRAGE 7 (9):
a) Definieren und charakterisieren Sie Lehmans P, S, E-Systeme und W-Systeme!
b) Welche Folgerungen leiten Sie davon für die Systementwicklung ab?
c) Geben Sie je ein Beispiel aus dem Bereich ==>

S-System: Spezifikationssystem; Lösung ist durch Spezifikation gegeben und muss nur implementiert werden. Frage: Entspricht die Lösung der Spezifikation? Ist die Spezifikation erfüllt, wird die Lösung akzeptiert.
BSP: Nummerntafelvergabe

P-System: Problem-System: Lösung muss gesucht und spezifiziert werden. Frage: Wird das Problem gelöst? Spezifikation ist nicht existent, oder pratktisch nicht verwendbar. Man findet eine Annäherung. Lösung ist akzeptierbar, wenn die Lösung in einem Lösungsraum, in dem sie eingesetzt wird, Sinn macht.
BSP: Verkehrssimulation. Spezifikation ist hier nicht vorhanden. Auswirkungen und Einflüsse bei Systementwicklung noch nicht bekannt.

E-System: Evironment-System: Es gibt verschiedene Lösungen und man muss die optimale auswählen.
BSP: Verkehrssteuerung. Es ist bekannt das das System Auswirkungen haben wird.

W-System: Bösartiges System (Wicked) Wie E-System nur mit vorgegebenen Voraussetzungen (z.B. Ziel auf Energiesparen)
ausserdem; (Unterschiede zu E System)
- Ohne Umfeld ist das Problem nicht vorhanden (das Problem fällt ohne Environment zusammen)
- Es gibt keine Richtige oder Falsche Lösung (nur besser oder schlechter)
- kein Test definierbar (kein bestimmter Test definierbar, denn wenn ein Test spezifiziert wird, so gibt es immer noch genügend andere Möglichkeiten für Versagen
BSP: Verkehrssteuerung, Verkehrsentlastungsstrecke: sehr komplexes System.

FRAGE 8 (5): Was sind die Dimensionen eines Entwicklungsprozesses? Was bedeuten sie?
- Entwicklungsniveau
- Entwicklungsrichtung: punkt im Skript durchgestrichen Bemerkung: siehe auch Teil 10 im Skript.
- Projektumfeld: Zeitdruck, Infrastruktur, Personal, Risiko, Organisation, Soft-Faktoren (Human Factors)
- Problemtypus: Lehman PSEW System. Frage 7
- Entwicklungsaspekt
- Engineering Disciplines
- Management Disciplines: mit technischer Entwicklung parallel laufende Prozesse.

FRAGE 9 (4):
a) Warum ist Abstraktion für Modellbildung wichtig (geben Sie mehrere Arguments!
b) Nennen Sie 5 wesentliche Abstraktionen aus dem Gebiet des/der ->

a)
- Trennt das Wesentliche vom Unwesentlichen
- Gemeinsames in einer Ansammlung von einzelnen Elementen sehen
- Hilft die Mechanismen zu verstehen
- Modellbildung geschieht durch Abstraktion

FRAGE 10 (5): (2-14)
a) Was sind wesentliche Eigenschaften eines Beschreibungsmittels (für ein Modell)?
b) Charakterisieren Sie die einzelnen Eigenschaften !

a&b)
Beeinflussung begrifflichen Denkens: In welche Richtung formiert die Beschreibung das begriffliche Denken? In welchen Konzepten wird der Benutzer zu denken ermutigt / entmutigt?
Expressivität: Was kann man (nicht) im Modell ausdrücken? Welche Beschränkungen werden auferlegt?
Eindeutigkeit / Verständlichkeit: Sind die verwendeten Bezeichnungen und Begriffe für alle eindeutig und verständlich?
Modularität: Inwieweit sind die Konstrukte unabhängig voneinander?
Ökonomie: Inwieweit kann man genau das ausdrücken was notwendig ist?
Konstruktions- und Interpretationskosten: wie schwierig ist es, das Modell durch einen menschlichen Betrachter und / oder durch einen Rechner zu interpretieren?
Traktabilität: Wie einfach / komplex ist es, Operationen auf dem Modell auszuführen?
Liberalität: wie strikt / liberal legt das Modell die dadurch zu beschreibenden Objekte fest? Kann man Objekte nur teilweise beschreiben und später verfeinern?
Dynamik der Beschreibung: Welche Festelgungen müssen vor der Modellierung getroffen werden, welche können während der Verwendung gefällt werden?
Weiterentwicklung: Wie gut kann man das Modell temporär / permanent an geänderte Bedingungen anpassen?

FRAGE 11 (3):
a) Was versteht James Martin unter Hyperdiagrammen?
b) Wozu dient in diesem Zusammenhang die Enzyklopädie (Repository)?

a) Ist eine Sammlung von verschiedenen Diagrammen derselben Menge von Objekten die logisch verbunden sind, und die verschiedene Aspekte und Details desselben Problems darstellen. Der logische Verbund gewährleistet die Konsistenz der Diagramme.
b) Das Repository sind die Daten aus welchen die Diagramme erstellt werden. Zentrale Entwicklungsdatenbank. Redundanz- und Widerspruchsfreie Entwurfsdaten.

FRAGE 12 (4):
a) Definieren Sie Anforderung

b) geben Sie 6 typische funktionale und 6 typische nicht-funktionale Anforderungen für das Software-Produkt >

a) Anforderung ist eine Aussage über eine zu erfüllende qualitative und / oder quantitative Eigenschaft.
Sie legen die Eigenschaften des Produkts aus der Sicht des Auftraggebers fest.
Funktional: Definieren Funktionen die das System durchführen muss. Transformation von Eingabe zu Ausgabe.
nicht-funktional: Beschreiben Eigenschaften die ein System haben muss. ("illities") z.B. Performance , Speicherverbrauch, Benutzerfreundlichkeit, Robustheit
b)
(Echtzeit Beispiel finden!!!)

FRAGE 13 (5): Nennen Sie mindestens 5 verschiedene Methoden der Anforderungsanalyse und geben Sie für jede mindestens 2 Vorteile und 2 Nachteile an (was Vor und Nachteil sind steht nicht im Skript, nur diese Beispiele)
Interviews: Zeitaufwand, Kosten, Verfälschung, Ehrlichkeit, persönlicher Kontakt
Fragebögen: Kostengünstig, Rücklaufquote, Akzeptanz, Missverständnisse, Anonymität
Analyse existierender Dokumente: "harte Fakten", Einseitigkeit, Komplettheit
Analyse existierender Arbeitsvorgänge:
Offline Beschreibung: Ehrlichkeit, Perception,
"Think Aloud": Störung, Ablenkung, Verfälschung
Externe Beobachtung/Automatische Aufzeichnung: Störung, geändertes Verhalten, Privatsphäre
Ethnographie: Eintauchen in die Arbeitswelt, Miterleben, Beobachten
Idealisierung: Top-down Entwicklung, Ableitung
Prototyping Experimente: Realismus, richtige Abstraktion
Verhandlung: Gemeinsame Abstimmung, Zeitaufwand, Machtausübung, Diskussion

FRAGE 14 (5): (3-7)
a) Was hebt die WINWIN-Methode gegenüber anderen Analyse-Methoden hervor?
b) Was versucht sie zu erreichen?
Computerunterstützungswerkzeuge, kann mit Spiralmodell vereinigt werden.
Ziel: Alle Teilnehmer zu Gewinnern machen.

FRAGE 15 (4):
a) Wer sind die Stake-holder bei einer Vorlesung?
b) Was könnten deren WINWIN-conditions sein?

a) Professor, Hörende
b) Hörende: Klausur schaffen, lernen
Professor: Wissen vermitteln, Studenten für das Fachgebiet begeistern

FRAGE 16 (8): (3-7)

a) Beschreiben Sie die Grundstruktur des WINWIN-Systems und erklären Sie die einzelnen Komponenten.
b) Was ist ein Stake-Holder bei der WinWin-Methode?
c) für die im Anschluss genannte Situation nennen Sie mindestens 8, max 10 unterschiedliche Stake-Holder an, begründen Sie ihre Wahl und geben Sie für jeden Stake-Holder eine WIN-condition an! Situation: ==>

a)
Agreements: Verhandlungen zu Beginn einer Systementwicklung. Erhalten Optionen
Options: widmen sich bestimmten Themen.
Issues (Themen): betreffen WIN Conditions
WIN Conditions: wichtiger Punkt den ein Stakeholder erfüllt haben will. Er hat gewonnen wenn dieser Punkt erfüllt ist.
Taxonomy: Beziehung zwischen den Strukturelementen

b) hat bestimmte Anforderungen an das System. Will etwas davon.

FRAGE 17 (3): Beschreiben Sie die drei UML-Diagramme, die besonders in der Phase der Anforderungsanalyse eingesetzt werden
Use-Case-Diagramm (Actors...): ein Muster eines Verhaltens des Systems: jeder Anwendungsfall beschreibt eine Sequenz von zusammenhängenden Transaktionen die von Aktor und System in gegenseitigem Dialog durchgeführt werden.
Sequenzdiagramm (Zeitliche Abfolgen...): Interaktionen mit zeitlichem Ablauf.
Klassendiagramm (Methoden, Attribute...): zeigt die Existenz von Klassen und deren Beziehung in Systemsicht

FRAGE 18 (8):
a) Welche Arten von Anforderungen (in Sinne der Anforderungsanalyse) können mit Use Case Diagrammen nicht oder kaum beschrieben werden? Charakterisieren Sie diese Art der Anforderungen!
b) Geben Sie 4 bis 5 Beispiele für derartige Anforderungen (verschiedene Arten!) für dem Bereich ==>

a) Da Use Case Diagramme rein Funktionale Zusammenhänge beschreiben, können nicht funktionale Anforderungen nicht dargestellt werden. Qualität, Performance, Speicherverbrauch, Robustheit,…

FRAGE 19 (5): Was sind die Teile einer Use-Case-Beschreibung?
Actor: interagiert mit dem System
Use Cases: ist der konkrete Anwendungsfall
Assotiations: Use Cases können sich gegenseitig beeinflussen „uses“ (zusammenhängen, benutzen) z.B. UC Zahlung verbuchen wird benutzt von UC Rechnung bezahlen und UC Direkte Zahlung der Parkgebühr

FRAGE 20 (5): Nennen Sie 5 UML-Diagramme und beschreiben Sie, in welcher Phase was mit dem jeweiligen Diagramm modelliert wird!
Use Case Diagramm, Anforderungsanalyse Frage 17
Klassendiagramm, Anforderungsanalyse Frage 17
Sequenzdiagramm, Anforderungsanalyse Frage 17
Komponentendiagramm(Entwurfsphase ... legt Abhängigkeiten dar die zwischen Komponenten existieren)
Verteilungsdiagramm, (Entwurfphase ... spricht konfiguration an(Prozesse, Hardware, Objekte)

FRAGE 21 (6): (a) Beschreiben Sie das (einfache) House of Quality ? (b) Nehmen Sie als Beispiel (ca. 5 Einträge pro Seitenkante) ==>
(3-12)

a)
Das einfache Haus der Qualität beschreibt die Objekte die für die Implementierung notwendig sind und wie sie zusammenhängen.
What's: eine strukturierte Liste von Kundenanforderungen.
How's: eine strukturierte Menge von relevanten und messbaren Produkt Charakteristika.
What's vs. How's: wie beeinflussen sich whats und hows gegenseitig (starke oder schwache Zusammenhänge, positiv oder negativ,…)?
How's vs. How's: wie beeinflussen sich die gewählten Lösungen gegenseitig?
In der Mitte ein Raster „What's vs. How's". Links die Zeilen stellen „What's" dar. Oben, die Spalten stellen „How's" dar. Über den „How's" als Dreieck angeordnet „How's vs. How's".

b)- ist schlecht, + ist gut, ++ ist sehr gut (hab nicht alles ausgefüllt, Konzept müsste aber klar sein)

	Video Lesegerät	Mehrere Einfahrten	Monats Rechnung	Mehrere Ausfahrten	
Keine Wartezeit einfahrt	++	++	+ (bezahlt wird ja normalerweise immer erst bei der Ausfahrt)	++ (Höhere Kundenfrequenz bei geringerer Wartezeit möglich)	
Keine Wartezeit ausfahrt	++		++	++	
Preis	-	-		-	
Genügend Plätze					
Gute Ausgänge					

Anmerkung: How's vs. How's ist zwecks technischer Gründe nicht als Dreieck über der Matrix eingezeichnet (Siehe Skript 4-17).

How's vs. How's:

	Video Lesegerät	Mehrere Einfahrten	Monats Rechnung	Mehrere Ausfahrten
Video Lesegerät		- (jede Einfahrt benötigt ein Lesegerät)	+	-
Mehrere Einfahrten	-		+	
Monats Rechnung				
Mehrere Ausfahrten				

FRAGE 22 (8):
a) Zählen Sie 5 Gründe auf, warum sich Anforderungen während eines Projektes ändern! (Requirements Creep)
b) Für jeden der Gründe sagen Sie, was man dagegen tun kann und zeigen dies am Beispiel aus dem Gebiet ==>

a&b) (hab die Abhilfen intuitiv zugeordnet. Übernehme also keine Gewähr. ☺)
Missverständnisse während des Kundengespräches: Alle Stake-Holder einbinden (z.B. Win-Win Methode)
Die Welt läuft dynamisch weiter: Rapid Application Development, Prototypen.
Geänderte Konkurrenzsituation: Risiko Analyse
Kunde lernt durch Entwicklungsprozess: Change Control Boards,
Weil Software die Prozesse im Unternehmen ändert: Change and configuration management system.
Goldplating („funcional overkill") („diese Funktion will ich auch noch"): cost per Function Point Contracts

FRAGE 23 (5):
Was versteht man unter CBS, ECBS? Was sind die Eigenschaften von CBS? (3-16)

CBS = Computer Based System
ECBS =Engineering of CBS
Definition CBS: ein System, dass als informationsverarbeitendes Teile ein oder mehrere Computer benutzt. Es beinhaltet somit jene Komponenten die nötig sind um: zu erfassen (capture), zu speichern (store), zu bearbeiten (process), zu übermitteln (transfer), anzuzeigen (display).
Inkludierte Einheiten:
Verarbeitungs- und Speichereinheiten aufgebaut aus digitaler und analoger Hardware, sowie Software und Firmware
Kommunikationseinheiten bestehend aus Netzwerk-Service und Transport Medium.
Menschliche Operatoren, Mensch-Computer Interaktionsservices und Steuer Services
Dokumentation und Benutzerhandbücher

Wege der Interaktion mit ihrer physischen Umgebung mittels Sensoren und Auslösern (actuators)

FRAGE 24 (5):
a) Definieren sie "harte Zeitschranke", "weiche Zeitschranke" und "feste" Zeitschranke und b) Erklären Sie die Unterschiede einem/r NichttechnikerIn
c) Geben Sie je ein Beispiel (mit Begründung) für die drei Arten von Zeitschranken aus dem Gebiet:->

a)
Harte Zeitschranke: tritt dann auf wenn unter allen Umständen keinerlei Überschreitung von Zeitbedingungen auftreten dürfen. Ist nicht allein von der Rechenzeit abhängig sondern auch von vorhersehbaren Zeiten.
Weiche Zeitschranke: tritt dann auf wenn ein Prozess (Prozesse) nur im Durchschnitt die Zeitschranken erfüllen müssen.
Feste Zeitschranke: Kombination aus weich und hart. Hat eine kürzere weiche und eine längere harte Zeitschranke.

b)
Die harte Zeitschranke ist eine Angabe die unbedingt eingehalten werden muss, ansonsten kommt es zu Fehlern und das Ergebnis wird wertlos.
Die weiche Zeitschranke gibt einen Soll-Wert an der in den meisten Fällen erfüllt werden soll. Gibt es einen Ausreißer so ist das hier nicht so schlimm.
Die feste Zeitschranke hingegen ist wie eine weiche, nur dass es einen Schranke gibt die eine Ausreißer in keinen Fall überschreiten darf. In diesem Sinne ist sie eine Kombination aus den Beiden vorherigen.

c)
hart: Herzschrittmacher….(tot bei Überschreitung)
weiche: Schutzimpfungen…. (sollen halt mal gemacht werden….)
fest: Auto Pickerl:… (Sollwert: beim Ablauf, kritischer Wert: 1 Monat danach)

FRAGE 25 (5): Welches sind die wichtigen Zeiten bei einen Echtzeitprozess? Beschreiben Sie diese und zeichnen Sie die grundlegenden Zusammenhänge.

Reaktionszeit: Zeit bis eine Benutzeranfrage angenommen wird. Reaktion des Systems
Antwortzeit: Zeit bis man eine Antwort vom System erhält.

FRAGE 26 (4): (4-2)
a) Geben Sie eine Definition von Architektur im Sinne der Systemlehre
b) Welche Bedeutung hat die Architektur für den Benutzer (Vorteile!)

a)

Die Architektur eines Systems kann als sein funktionelles Erscheinungsbild dem Benutzer gegenüber (seine Phänomenologie) definiert werden. (=Schnittstelle eines Systems die der Benutzer sieht)

b)
Die Architektur entscheidet über eine gute Benutzbarkeit eines Systems. Die Wahl der richtigen Architektur ist somit entscheidend für den Benutzer.

FRAGE 27 (3): a) Was ist SSADM? In welchem Land wird es hauptsächlich eingesetzt? Wofür wird SSADM besonders eingesetzt?
SSADM = Structured Systems Analysis and Design Method
Weitgehend Standard in Großbritannien (UK). Ist eine Datenflussdarstellung. Basiert auf SA (Structured Analysis).
Wird speziell für die Wartung eingesetzt (Wartungsmodell).

FRAGE 28 (5): (5-5)
a) Welche UML-Diagramme beschreiben besonders die Spezifikation (Analyse)?
b) Beschreiben Sie diese Diagramme!

a&b)
Zustandsdiagramm: Zustandsänderungen eines Objektes einer Klasse. Zeigt: Lebensgeschichte einer Klasse, Auslöser für Zustandänderungen, Aktionen die von Zustandänderungen herrühren. Für Objekte mit dynamischen Verhalten. Aktionen sind atomar.
Kollaborationsdiagramm: Interaktion von Objekten untereinander.
Aktivitätsdiagramm: Beschreibt Prozess bestehend aus Aktionen und Aktivitäten, Kontrollfluß, benötigte und resultierende Objekte, verantwortliche Objekte.
Eine Aktion ist atomar und wird durch einen Aktionszustand repräsentiert.
Eine Aktivität kann hierarchisch weiter unterteilt werden, und wird ebenfalls durch einen Aktivitätszustand repräsentiert.
Kontrollfluß: Reihenfolge von Aktionen und Aktivitäten, dargestellt durch Transitionspfeile, Bedingungen und Aktionen sind erlaubt, Sobald Ausführung des Vorgängers abgeschlossen ist wird Ausführung des Nachfolgers begonnen.
Verantwortlichkeitsbereich (swimlane): verantwortliches Objekt kann zugeordnet werden (z.B.: Objekte des zu realisierenden Systems, Akteure oder Organisationseinheiten des Unternehmens)

FRAGE 29 (8):
a) Nennen Sie die 8 Software-Prinzipien und erklären Sie diese
b) Erläutern Sie, was die Einhaltung jedes dieser Prinzipien für die Software-Qualität spezifisch bringt (nicht nur "Erhöhung") Erhöhung der Qualität,
c) Geben Sie für zwei Prinzipien ein praktisches Beispiel aus einer Programmiersprache

a) (steht nicht mehr im Skript)
Prinzip der...
-Abstraktion
-Strukturierung

-Hierarchisierung
-Modularisierung
-Geheimnisprinzip (=Information Hiding)
-Lokalität (...lokale Variablen)
-Verbalisierung
-Ergonomie

FRAGE 30 (4): a) Welche UML-Diagramme beschreiben besonders die Software-Architektur (d.h. den Entwurf)?
b) Beschreiben Sie diese Diagramme! c) Was fällt Ihnen bei UML bezüglich der weiterführenden Implementierung eines Software-Projektes auf

a&b)
Komponentendiagramm
Beschreibt Softwarekomponenten und ihre Abhängigkeiten. Pakete können Komponenten beinhalten oder umgekehrt.
Verteilungsdiagramm
Zeigt die aktuelle Hardware Konfiguration bestehend aus: Knoten (im Normalfall Prozessoren, I/O,…), SW-Komponenten, Prozessen, Objekten und den Kommunikationskanälen zwischen den Knoten.
Pakete:
Gruppiert eine beliebige Anzahl an UML-Modellelementen. Können selbst wieder Pakete enthalten. Ein UML-Modellelement kann höchstens zu einem Paket gehören. In verschiedenen Paketen können dieselben Namen verwendet werden.

c)
vielleicht genaue richtlinien für jeden Programmierer. Schnittstellen sind spezifiziert, und Namenskonflikte durch Pakete gelöst, sodass verschiedene Komponenten zusammenspielen können, und somit eine Entwicklung verteilt auf verschiedene Personen möglich ist.

FRAGE 31 (4):Warum trennen wir Produkt- und Prozess-beschreibung?Was bringt eine gesonderte Betrachtung des Prozesses?

Die Produktbeschreibung definiert WAS erzeugt werden soll.
Die Prozessbeschreibung dagegen WIE es erzeugt werden soll.

Langfristig gesehen (über ein Produkt hinaus) ist es sinnvoll sich Gedanken über den Entwicklungsprozess zu machen nicht nur über das Produkt denn:

Der Prozess ist:
Auf verschiedene Produkte anwendbar
Analysierbar
Durch Erfahrung verbesserbar
➔
Abstraktion vom einzelnen Produkt
Abstraktion gleichartiger Funktionen

Abstraktion vom individuellen Ingenieur

Ein definierter Prozess bringt:

Ordnung
Richtige Reihenfolge
Durchführung notwendiger Tätigkeiten
Größere Unabhängigkeit von Personen
Gesamtsicht des Prozesses
Lehrbarkeit
Akkumulierung von Know-how
Prozessanalyse (Zertifizierung)

Und schlussendlich natürlich Qualität…

FRAGE 32 (5): a)Wenn man einen Prozess als Modell beschreibt, was sind die Abstraktionen? b) Wie entsteht aus dem Modell realer Prozess?

(Wie oben)
Abstraktion vom einzelnen Produkt
Abstraktion gleichartiger Funktionen
Abstraktion vom individuellen Ingenieur

Das Modell muss instantiiert werden. Mit anderen Worten, das Modell beschreibt die Vorgehensweise ➔ Vorgehensmodell und dieses wird bei der Arbeit zum fertigen Produkt hin angewandt.

FRAGE 33 (5): a)Was ist ein Vorgehensmodell? Beschreiben Sie das Metamodell aus der Vorlesung!

Definition:
Unter einem Vorgehensmodell verstehen wir eine Darstellung, die weitgehend den Systementwicklungsprozess beschreibt und Analysen des Prozesses gestattet.

Bedeutung:
Komplette Festlegung, Planbarkeit
Kommunikationsverbesserung
Schulbarkeit
Begleitende Kontrolle
Revisionsfähigkeit, Zertifizierung
Verbesserungsfähigkeit des Prozesses
Qualitätssteigerung der Produkte
Kostenreduzierung

Metamodell: (siehe Abbildung 9-1 auf Seite 9-6)
Beschreibung?

FRAGE 34 (5): a)Was beschreibt ISO12207? b)Welche Hauptprozesse (essential processes) werden beschrieben?

a. Der ISO 12207 Standard beschreibt die Aktivitäten der Software Entwicklung.
b.
Primary Life Cycle Processes
Supporting Life Cycle Processes
Organizational Life Cycle Processes

Für Unterpunkte siehe Seite 9-11

FRAGE 35 (6): a)Was ist Information Engineering (nach J. Martin) ? b)Welche Ideen stecken dahinter (Erläuterung) ?

a.
Definition:
The application of an interlocking set of formal and automated techniques for the planning, analysis, design and construction of information systems on an enterprise- wide basis or across an major section of the enterprise, built upon a comprehensive knowledge base.

Frei Übersetzt:
Information Engineering ist die Anwendung von einer verzahnten Menge von formalen und automatisierten Techniken für Planung, Analyse, Design und Konstruktion von Informationssystemen auf einer Basis die Unternehmensweit ist, oder zumindest einen Hauptteil des Unternehmens abdeckt; Aufbauend auf einer umfassenden Wissens Basis.

b.
Bevor Anwendungen entwickelt werden, sollen zuerst die Daten, Informationsflüsse und Prozesse eines Unternehmens ermittelt werden. Seine kritische Bewertung zusammen mit der Festlegung einer strategischen Zukunftsplanung sind die Basis für die Identifizierung der notwendigen Anwendungen die anschließend implementiert werden.

FRAGE 36 (4): Beschreiben Sie das Wasserfallmodell! b)Was sind seine Vorteile? c) was sind seine Nachteile?
a.
Es gibt fix definierte Stufen die in genau dieser Reihenfolge abgearbeitet werden.
Nach jeder Stufe gibt es eine Phase der Verifikation wobei auf die vorige Stufe zurückgesprungen werden kann.
b.
+ Etabliert
+ Validierung nach jeder Phase
c.
- Starr/ Unflexibel
- Fehler am Anfang ziehen sich bis zum Ende hin durch und werden teuer in der Korrektur.

FRAGE 37 (6): a)Was ist das (deutsche) V-Modell (Sinn und Aufgabe)? b)Welche 4 Teilmodelle enthält es (Erklärung!)? c) Erklären Sie das "Tailoring" im V-Modell!

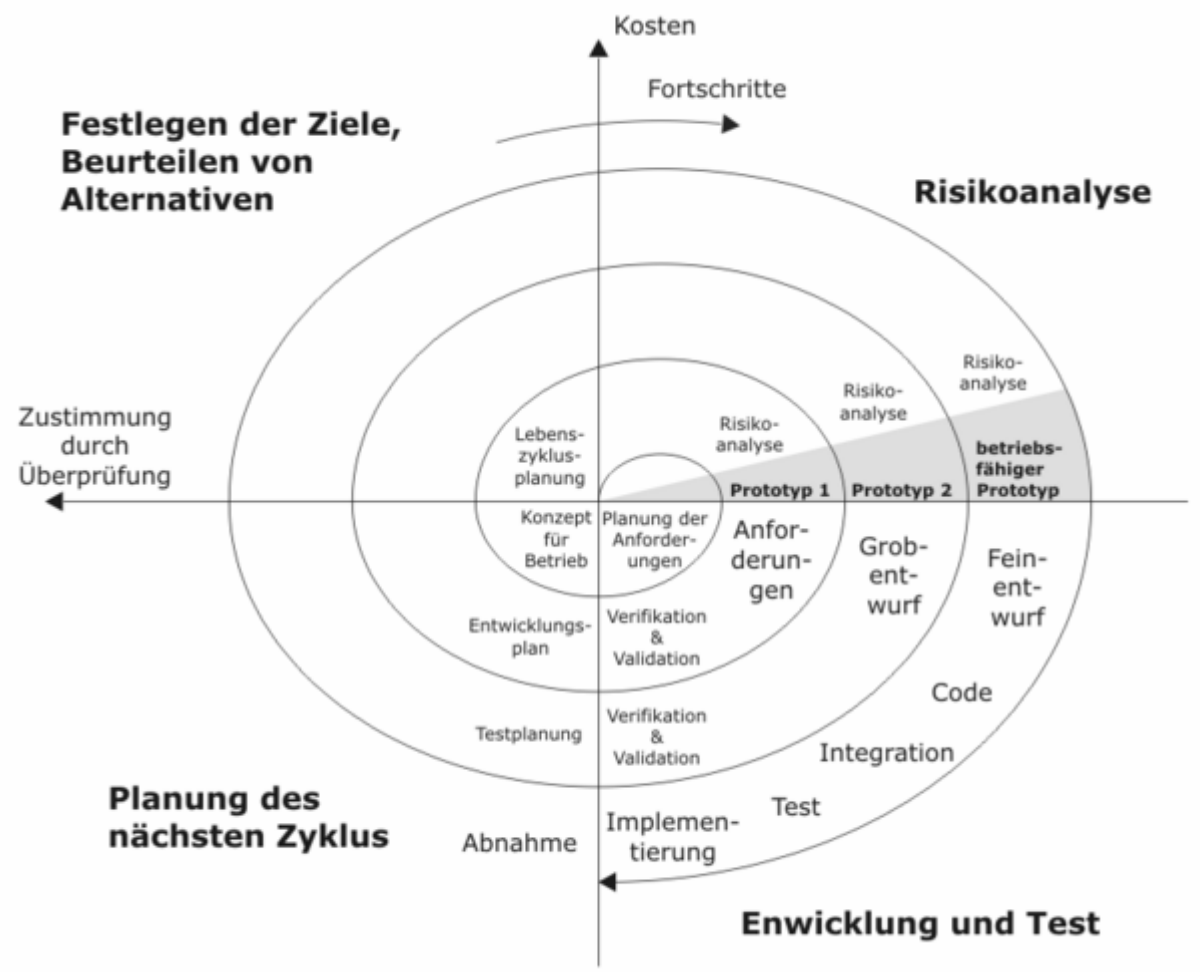
Seiten 10-5 und 10-6

Projektmanagement, Qualitätssicherung, Konfigurationsmanagement, Software- Erstellung

FRAGE 38 (5): a)Was ist die Grundidee? b)Warum wird Concurrent Engineering immer wichtiger? c) Welche organisatorischen Massnahmen sind notwendig?

a. Durch Parallelisierung wird die Entwicklung beschleunigt.
b. Time to Market wird geringer, was wichtig ist weil die Computerbranche immer kurzlebiger wird.
c.
Überlappung der Produktionsphasen.
Parallelisierung und Trennung von Aspekten.
Dislozierte Gruppen.
Wichtig!!: Erhöhte Kommunikation zwischen den Entwicklern des Projekts.
Die Planung muss mit noch mehr Sorgfalt durchgeführt werden.

FRAGE 39 (6): a)Was sind die wesentlichen Annahmen und Prinzipien des Spiral-Modells? b) In welchen wesentlichen Punkten unterscheidet es sich vom Wasserfallmodell? c) In welchen wesentlichen Punkten unterscheidet es sich von XP?



Das Spiralmodel ist eine Weiterentwicklung des Wasserfall Modells in dem die Phasen mehrmals durchlaufen werden. Der Grund für diese Architektur ist, dass sich die Anforderungen während des Projektes ändern könnten oder klarer werden könnten und diese sollen in das Produkt mit eingehen!
Das Spiralmodel wiederholt die Phasen zyklisch immer wieder wodurch der ganze Entwurf viel flexibler wird und auf Änderungen reagiert werden kann. Es ist für relativ dynamische Anwendungsbereiche vorzuziehen
XP ist sehr Codezentriert. Auch hier gibt es sehr häufige Validierungen, aber alles eher auf der Programmierebene denn XP versucht den Management Aufwand gering zu halten und dadurch die Time to Market zu verkleinern. Das Spiralenmodell hingegen gleicht seine „Trägheit“ durch zyklische Wiederholung von Phasen aus wodurch die Chance steigt, dass ein passendes Programm rauskommt.

FRAGE 40 (5): Vergleichen Sie das Wasserfall-Paradigma mit dem Prototyping-paradigma! Vorteile? Nachteile?

Das Wasserfall Paradigma gibt einen Ablauf der Phasen (Teilprozesse) an und beinhaltet dabei auch die Validierung nach jeder abgeschlossenen Phase sowie den Rücksprung auf eine vorhergegangene. Prototyping ist in Ingenieursdisziplinen durchaus üblich und beschäftigt sich von der Grundidee her eher damit ein vorläufiges Modell der zu erstellenden Software herzustellen um eine gemeinsame Gesprächsbasis aufzubauen oder unbekannte Eigenschaften zu ermitteln. Bei Evolutionären Prototyping jedoch geht es darum einen Prototyp weiter zu entwickeln bis er zum fertigen Produkt wird.

Vorteile/Nachteile:
Das Wasserfall Paradigma geht an ein Problem strukturiert ran, während das Prototyping zwar evolutionär sein kann aber weniger Management Aspekte zulässt.
Meiner Meinung nach ist Prototyping eher für Gesprächsbasis und Schätzungen gut und kann dadurch als Hilfmethode bei anderen Paradigmen eingesetzt werden.

FRAGE 41 (5): a)Was ist die Grundidee von Extreme Programming (XP) und seines Namens?
b) Beschreiben Sie 6 der 12 Praktiken von XP!

a. Man nehme bewährte Praktiken aus dem Software Engineering und drehe die Regler auf vollen Anschlag ➔ Extreme. Arbeit ist Code- zentriert. Programmierer programmieren im Kleingruppen (zu 2t). Spornen sich an, programmieren so dass beide es verstehen ➔ einfach. Der Kunde wird maximal eingebunden.

b.
Pair Programming (eh klar- aber Paare werden gewechselt)
On Site Customer (Kunde wird stark miteinbezogen, befindet sich im Extremfall sogar bei den Entwicklern)
Simple Design (Alles einfach halten, nichts redundantes oder zusätzliches das nicht benötigt wird)
Testing (Tägliche Tests (inkrementell))
Short Releases (Das Zwischenprodukt soweit lauffähig immer wieder freigeben)
Collective Ownership (Auch der Code an dem man arbeitet wird gewechselt (zirkulär))
40 Hour Week
…

FRAGE 42 (7): a) Nennen Sie die wesentlichen Entwicklungsrichtungen und b) für diskutieren Sie, welche Entwicklungsrichtung Sie in welcher Situation einsetzten würden (Vorteile, Nachteile, besondere Eignung, ...

a+b.
Vorwärtsentwicklung
Wenn die Vorgaben/Spezifikation klar und statisch ist. Kann nicht auf dynamische Anforderungs -Änderungen reagieren ist sonst aber zielgerichtet (zb Wasserfall modell)

Interativ(Prototyping)
Wenn evolutionäres Vorgehen Vorteilhaft, zB. Anforderungen von vorne herein noch nicht ganz klar. Verwaltungsaufwand unter Umständen relativ hoch.
Weiterentwicklung
Wenn von einem guten Grundstock ausgegangen werden kann und dieser von der Technik her noch nicht veraltert ist. Schneller als Neuentwicklung. Nachteilig ist, dass Fehler mitgezogen werden und eventuell nicht passende Strukturen umgebessert werden müssen.
Fehlergesteuert(Wartung)
Wenn schnelle Time- to- Market wichtig ist und das Produkt in einem Fehlertoleranten Umfeld eingesetzt wird. Eher für Einzelkunden Programme. Vertrauen des Kunden wird nicht unbedingt gesteigert. (Typisch bei manchem Open Source Produkten weil Testaufwand/Kosten gering).
Re-Engineering
Wenn sich eine Weiterentwicklung des Systems nicht rentieren würde oder sich grobe Fehler (richtig geiler Pfusch) im vorhanden Produkt befindet. Zeitaufwand höher als bei Weiterentwicklung, aber Erfahrungen können mit einfließen.
Wiederverwendung
Typisch auf Komponentenbasis. In Europa durchaus üblich (laut Software Triade). Endprodukt kann schnell zusammengestöpselt werden und auf die Einzelkomponenten kann man sich meist verlassen (natürlich abhängig vom Hersteller). Bei Mängeln an vollkommener Kompatibilität kann der Gesamteindruck des Endproduktes leiden.

FRAGE 43 (7): a) Nennen Sie die 4 Arten der Wartung und ihr prozentuales Vorkommen!
b) Zählen Sie 5 bis 6 verschiedene Gründe auf, warum Software-Wartung in vielen Fällen günstiger im Vergleich zu Neu-Entwicklung ist und geben Sie zu jedem Ihrer Gründe ein Bespiel aus dem Gebiet der ==>

a.
Korrektive Wartung (Behebung von Fehlern) 22%
Adaptive Wartung (Anpassung an Änderungen) 23%
Perfektive Wartung (Verbessern und Erweitern) 42%
Präventive Wartung (vorsorglich ➔ Zukunftssicherheit) 13%

b.

Neuschreiben teurer als Wartung
Verfügbarkeit der neuen Software zu spät
Oft höhere Qualität
Verlängert die Lebensdauer
Grundlegende Funktionstüchtigkeit
Verborgene Funktionalität

FRAGE 44 (4): a)Welche wesentlichen vier Felder kann man bei der Portfolio-Analyse unterscheiden! (Diagramm!)
b) Beschreiben Sie die beste Strategie für jedes der Felder"

Einsatzerwartung
^ solide Basis Sorgenkind
|
| net amol ignorieren Hackers Freude
----------> Wartungswahrscheinlichkeit
Solide Basis: Typischer Fall von genauer Spezifikation, bei der Implementierung sollte auf eine iterative Methode zurückgegriffen werden und kumulierende Tests durchgeführt werden.
Net amol ignorieren: Diese Bezeichnung gefällt mir nicht. Bsp.: Notfallfallschirm in einem Kampfflugzeug. Muss ganz einfach funktionieren auch wenn nicht von einem Einsatz ausgegangen wird (das Glas ist halb voll ;). Daher würde ich wie bei der Soliden Basis vorgehen, zumindest wenn es sich um kritische Anwendungen handelt.
Hackers Freude: Hier ist wohl eine Neuentwicklung oder Weiterentwicklung angebracht um die Wartungswahrscheinlichkeit zu mindern.
Sorgenkind: Schwieriger Fall, Weiterentwicklung ist angebracht, denn für eine Neuentwicklung ist keine Zeit.

FRAGE 45 (5): Zählen Sie die wesentlichen 5Wiederverwendungskonzepte auf ("die 5 Re's") und diskutieren Sie diese!

Re-Do ---- Neuentwickeln
Re-Engineer --- Neugestalten
Re-Structure ---- NeuStrukturierung
Re-cap ---- Altes System in neues integriert.
Re-main --- Beim Alten bleiben.

FRAGE 46 (4):Was muß man beachten, wenn man Software entwickelt, die wiederverwendbar sein soll?

systematisch (für Suche und Auswahl)
generisch (für Anwendbarkeit, frei von speziellen Abhängigkeiten)
umfassend(für ein Spezialgebiet)
effizient (in der Laufzeit)

FRAGE 47 (5): Stellen Sie die Entwicklungsschritte für Software-Montage dar!

Abbildung 12.17 auf (12-13)

FRAGE 48 (5): Skizzieren Sie ein mögliches Vorgehensmodell für die Wiederverwendung von Software

Abbildung 12.11 auf (12-8) zumindest eventuell…

FRAGE 49 (4): Bei einer Portfolio-analyse von Altlasten muß man welche 4 Felder unterscheiden? Diskussion!

Wie Frage 44 nur dass man nun von einem bereits bestehenden System ausgeht.

FRAGE 50 (5): a)Was ist der Zusammenhang zwischen Produktivität, Qualität und Wartungsaufwand (Skizze!) b) Kommentieren Sie!

a. Abbildung 14.2 auf (14-2)
b.
steigender Wartungsaufwand ➔ sinkende Produktivität (und umgekehrt)
Steigende Qualität ➔ sinkender Wartungsaufwand (und umgekehrt)
Steigende Produktivität ➔ steigende Qualität (und umgekehrt)

Bei und Umgekehrt bleibt die Richtung gleich nur steigend und sinkend toggeln.

FRAGE 51 (8): Nennen und kommentieren(!) Sie acht wesentliche Unterschiede zwischen ISO 9000 und ISO 15504.

9000 ist branchenneutral während 15504 auf Software zugeschnitten
Resultat von 9000 (ja oder nein). Bei 15504 feingranularer (Abstufungen)
9000 bewertet ganzen Prozess. 15504 trennt in Prozess Kategorien.
9000 für Zertifizierung. 15504 auch direkt für Prozessverbesserung
9000 nur Aussage über Prozeßqualität. 15504 -> Profil mit stärken und Schwächen (auch für Wahl des richtigen Unternehmens gebrauchbar)
15504 für Selbstevaluierung geeignet
15504 für Kleinbetriebe direkt nutzbar (selbstevaluierung), 9000 durch Teilzertifizierungen auf selben Niveau.
… keine Ahnung

FRAGE 52 (5): Nennen Sie die 5 Reifegradstufen (maturity levels) von CMM und charakterisieren Sie jede!

Initial (Anfang)
Repeatable (Software configuration management, qualitätssicherung, projektplanung…)
Defined (peer reviews, organisation process definition, inter group coordinition…)
Managed (software quality management, quantitatives prozess management)
Optimizing (Prozess change management, defect prevention, technology change management)

FRAGE 53 (6):Was sind die 5 CMM-Levels und was ist die Charakteristik der Software-Entwicklung auf jedem Level?

Die 5 CMM Levels geben an wie gut ein Entwicklungsprozess strukturiert und ge- managed ist. Die Meisten befinden sich zwischen Level 1 und 2 und nur eine Hand voll ganz oben.

Ansonsten wie 52. zumindest hab ich nichts gegenteiliges gefunden.

FRAGE 54 (7): a)Was ist ISO 9000? b)Warum gibt es ISO9001, bzw 9000-3? c)Was ist ein prinzipieller Unterschied zwischen einerseits der ISO 9000 Familie und andrerseits BOOTSTRAP und SPICE=ISO15504?

a.
Es ist nicht leicht Prozessqualität zu messen, darum gibt es den ISO 9000 Standard. Er ist:
Eine Gruppe von Prozess- Standards, wie ein QS System dokumentiert sein soll
Branchenneutral
International anerkannt
b.
Der ISO 9000 ist Branchenneutral während ISO 9001 für Softwareproduzenten abgestimmt ist (wird durch ISO 9000-3 interpretiert!)
c.
ISO 9000 gibt als Resultat eine boolesche Entscheidung (ja, nein / bestanden, nicht bestanden)
Während Spice und Bootstrab weit feingranularere Ergebnisse liefern und in Klassen aufteilen.
FRAGE 55 (6): a)Was ist ISO 15504? b) Nennen Sie die 5 "Process Categories" c) Welche Bereiche der Software Entwicklung decken die einzelnen "Process Categories" ab?

a.
ISO 15504 = Software Process Assessment = SPICE Software Process Improvement Capability dEtermination
Standard Referenz Modell für Assessments(= Auswertungen).
Öffentlich verfügbarer nicht proprietärer Standard für Prozess Assessment und Ratgeber für Prozess Optmierung/Verbesserung. Gibt auch kleinen Firmen die Möglichkeit sie zu verwenden.

b+c.
Primary Processes:
Costumer – Supplier
Engineering
Support Processes:
Support
Organistional Processes:
Management
Organisation

FRAGE 56 (4): a)Warum ist Software Qualität wichtig? b)Wie hängt Sie mit den Produktkosten zusammen?

a.
Die Qualität ist wichtig denn sie ist ein Maß wie zufrieden der Kunde mit dem Produkt ist (erfüllt seine kombinierten Erwartungen). Bei der Wiederverwendung und Weiterentwicklung ist es essentiell auf einem guten Grundstock aufzubauen. Für Qualität steigen die Kosten denn:

b.
Es erfordert mehr Produktivität um ein qualitativ hochwertiges Produkt herzustellen. Außerdem steigen Sekundärschäden bei zu geringer Qualität:
Fehlerrisiko steigt
Imageverlust
Behebungskosten
Schadenersatz
Kleinbetrieb (Existenzverlust)

FRAGE 57 (4): a) Definieren Sie ’Fehlfunktion’, ’Fehlerzustand’ und ’Mangel’! b)Was sind die Bedingungen für Fehlererkennung

a. Definitionen:
Fehlfunktion: Wenn das System von der Spezifikation abweicht.
Fehlerzustand: Fehlerzustand ist jener Systemzustand der zur Fehlfunktion führt.
Mangel: Bezeichnet die tatsächliche Ursache für den aufgetretenen Fehler.

b.

Mangel muss ausgeführt werden.
Die Ausführung mussden nachfolgenden Datenzustand beeinflussen.("infizieren")
Die Infektion im Datenzustand muss weitergegeben werden und eine Ausgabe hervorrufen.

FRAGE 58 (5): Beschreiben Sie Wege, um (im allgemeinen) ein qualitativ hochwertiges Produkt herzustellen (mind. 7)!Welche davon sind auch im Software Engineering praktikabel anwendbar?

Nachher: Test, Inspektion, Beweis
Vorher: Konstruktion, Anleitung
Prototyping
Dokumentation der Arbeit
Aufgabenorientierung
Standards einhalten
Erfahrung und Training

Alle dieser Wege sind anwendbar denn sie kommen direkt aus dem Bereich des SE.

FRAGE 59 (6): a)Was meint Crosby mit seinem Ausspruch (und seinem Buch) ’Quality is free’? b) Welche 5 Stadien des Qualitätsbewußtseins identifiziert Crosby? c)Welche Schlußfolgerungen ziehen Sie für die Software-Entwicklung daraus? (Diskussion!)

a.
Qualität kann auch ohne großen finanziellen Aufwand erreicht werden.

b.

Ungewissheit - Wir wissen nicht warum wir Schwierigkeiten mit der Qualität haben
Erwachen - Müssen wir immer Probleme mit der Qualität haben.
Erleuchtung - Mittels Management-Engagement und Qualitätsverbesserungen können wir unsere Probleme identifizieren und lösen
Wissen - Fehlervermeidung ist Routine in unserem Prozess.
Gewissheit - Wir wissen, warum wir keine Schwierigkeiten mit der Qualität haben.

c. Es ist wichtig sich über die Qualität und Qualitätsmängel bei dem eigenen Produkt im Klaren zu sein. Wenn welche bestehen sollen diese behoben werden und aus der Behebung soll man etwas für spätere Entwicklungen lernen damit sie gar nicht erst auftreten….. usw. blablabla.

FRAGE 60 (6): a) Beschreiben Sie ISO9126 b) Vergleichen Sie ISO9126 mit ISO12207: Was sind die Anwendungsgebiete, was sind wesentliche Unterschiede, was Ähnlichkeiten?

ISO9126 ist ein Qualitätsstandard der in Qualitätsattribute aufteilt:
Functionality (Funktionalität)
Reliability (Verlässlichkeit)
Usability (Benutzbarkeit)
Efficiency (Effizienz)
Maintainability (Erweiterbarkeit)
Portability (Portierbarkeit)

b.
Vergleich:
ISO9126 gibt die Eigenschaften an die eine gute Software haben sollte während ISO12207 die Aktivitäten der Software Entwicklung beschreibt, also eher den Weg wie man zu den vorher genannten Eigenschaften kommt. Die Ähnlichkeit ergibt sich aus dem Schnittpunkt der Endzwecke- die Software soll gewissen Standards genügen.

FRAGE 61 (5):Welche Qualitätskriterien aus ISO 9126 kennen Sie und wie kann man sie in Haupt- und Nebencharakteristiken klassifizieren?

Functionality - suitability, accuracy, interoperability, security
Reliability - maturity, fault tolerance, recoverability
Usability - understandability, learnability, operability, attractiveness
Efficiency - time behaviour, resource utilization
Maintainability - analyzability, changeability, stability, testability,
Portability - adaptability, installability, co-existence, replaceability

FRAGE 62 (7): a) Erklären Sie Ihrem Chef (Jurist) den Begriff ’good-enough quality’! b) Beschreiben Sie den Bewertungsprozess c) Beschreiben Sie die für das Produkt notwendigen Bedingungen an Hand des Beispieles ==>

a.

Produkt bietet
genügend Nutzen
hat keine kritischen Probleme
der Nutzen überwiegt deutlich die Probleme
weitere Verbesserung schadet mehr als dass es nutzt
jetzige Form des Produkts ist mehr wert, als wenn es nicht ausgeliefert würde

b.
Evaluierungsprozess
Identifizieren : Nutzen - Probleme
Wahrscheinlichkeit des Auftretens der Probleme
Auswirkungen auf Kunden
Kritikalität des der Probleme abschätzen
Abwägen: Nutzen - Probleme
Logistik weiterer Verbesserungen identifizieren

FRAGE 63 (7): a)Was ist Validierung und was ist Verifizierung (Definition)?Was ist der Unterschied?
b) Welche Wege zu einem Qualitätsprodukt kennen Sie (mit Erklärung)?
c) Welche sind in der Sw-Entwicklung für die Validierung und welche für die Verifizierung anwendbar?

a.
Verifizierung : Ist das Produkt korrekt ? vertikale und horizontale Verifizierung:
vertikal: Resultate eines Entwurfschritts entsprechen Vorraussetzungen und der Transformation
horizontal: ob die Elemente untereinander konsistent sind

Validierung : Ist es das richtige Produkt ?
menschliche Bewertung, wird es Markt - und Kundenwünschen entsprechen? Egal ob es korrekt ist.

b.
Nachher: Test, Inspektion, Beweis
Vorher: Konstruktion, Anleitung
Prototyping
Dokumentation der Arbeit
Aufgabenorientierung
Standards einhalten
Erfahrung und Training

FRAGE 64 (7): a) Beschreiben Sie den Ablauf einer klassischen Inspektion (nach Fagan)
b) beschreiben Sie die Vorteile (mindestens 5 verschiedene Vorteile)
c) diskutieren Sie auftretende Probleme, bzw. Nachteile (mindestens 5 verschiedene Nachteile, aber nicht nur die Negation von b!)

a.
Kickoff Meeting -> Inspection Meeting -> Rework -> Evalutation

Vom Kickoff bis zum Inspection Meeting erfolgt individuelle Vorbereitung.
Planung erfolgt parallel zu allen schritten und beeinflusst diese.

b. Vorteile:
Konkurrenzaspekt (Wettrennen -Fehler finden)
Checklisten werden verwendet und eingehalten (Typische Problemklassen)
Implizit erfolgt eine Mitarbeiter Evaluierung
Fehlersuche durch andere als Entwickler selbst (bessere Erfolgswahrscheinlichkeit)
Erklären des Codes -> Entwickler wird sich seiner Fehler selber bewusst

c. Nachteile:
Disziplin (Fehler finden aber nicht sofort beheben dürfen/können)
Emotionale Barrieren (kann nicht einfach zum Entw. sagen -> „das ist scheiße“)
Ungeduld der Entwickler („Wir habe den Fehler schon, machen wir Schluss“)
Sehr kleine Firmen (z.B.: 100% der Gesamtarbeitkraft wird in Inspektion gesteckt)
Entscheidung über Re-Inspektion (Schmal Grad –„bei schweren Fehlern“- was ist schwer?)

FRAGE 65 (6): a) Definieren Sie Blackbox-, Whitebox- und Greybox-Test. b) Diskutieren Sie die Vorteile/nachteile jeder Art

a+b.
Blackbox: Es ist nur der In und Output bekannt (Schnittstellen). Wie das System intern funktioniert ist gänzlich unbekannt/ bzw. das System wird nur in dieser Abstraktionsstufe betrachtet!

+ Test auf der Basis wie das System auch real benutz wird.
-- Das zu testende System/Teilsystem muss bereits komplett sein.
-- Fehler müssen erst auf ihren Ursprung zurückgeführt werden.

Whitebox: Das System ist vollkommen bekannt. Alle Internen Vorgänge sind bekannt.

+ Das System muss unter Umständen noch nicht komplett sein.
+ Fehler können lokal gefunden werden (exaktere Fehlerbeschreibung)
+ Für kumulierende Testfallentwicklung geeignet
-- Komplex und Zeitintensiv

Graybox: Mischform- es ist mehr als die Schnittstelle bekannt aber nicht die ganzen internen Vorgänge.

+ Genauere Fehleranalyse als bei Black Box
-- Es müssen immer Mittelwege gegangen werden
-- Verborgene Systemteile müssen komplett sein

FRAGE 66 (4):Welche 4 Grundgrößen des Aufwandes sind bei SW-Projekten zu berücksichtigen?Wie hängen Sie voneinander ab?Welche grundlegenden Einflußfaktoren gibt es?
????
Implementierung
Test
Management
Integration

Oder natürlich (Sind allerdings Einflussfaktoren)
Kosten, Zeit, Funktionalität und Qualität

FRAGE 67 (5): Wofür benötigt man Produktmanagement? Was sind seine Aufgaben?
- unternehmensinterne Koordination mit anderen Produktentwicklung
- Analyse der Marktsituation und der Konkurrenz
- Preisbildung
- Auswahl der Marketingstrategie
- Werbung für des Produkt
- Ausbildung des Verkaufspersonals
- Ausbildung des technischen Personals
- Ankündigung des Problems
- Schulung der Kunden
- technische Beratung und Unterstützung beim Einsatz

FRAGE 68 (5): Sie sind VerkäuferIn eines UML-basierten Modellierungswerkzeuges (Paradigm Plus, Rational Rose o.ä).Welche Vorteile würden Sie mit welchen Argumenten einem potentiellen Kunden im Vergleich mit einem Zeichentool (PowerPoint, Visio) anpreisen?

Es gibt eine Art Datenbank um alle Diagramme konsistent zu halten.
Sie ermöglichen leichteres Einfügen von neuen Elementen sowie bisher gespeicherte Objekte können sehr leicht eingefügt werden.
Des weiteren implementieren diese Programme bereits den UML-Standard und bieten entsprechende Unterstützung sowie auch die Möglichkeit das Metamodell von UML zu verändern und diese Veränderungen innerhalb des Unternehmens zu verbreiten.
sontiges blabla

FRAGE 69 (5): Welche Verkaufsargumente würden Sie für ein System wie Composer (oder COOL) ins Treffen führen? (Pro-Argumente)

Methoden unterstützt (Konsistenz, Überprüfung, Qualität)
Generatoren einsetzen (Arbeitsschritte eliminieren)
Qualität der Doku verbessern
Wieder- Verwendbarkeit
Konfigurationen und Versionen verwalten

Metriken einsetzen

Unterstützung des jeweiligen Prozeßmodells
Verbesserung der Entwicklungsmethoden
Erhöhung der Standardisierung
Jederzeit Information über soll/ist Zustand
Flexible Anpassung an geänderte Rahmenbedingungen

FRAGE 70 (5):
a) Was sind die wesentlichen Anforderungen an ein CASE-Werkzeug?
b) Mit welchen Argumenten würden Sie einem Kunden ein CASE-Tool schmackhaft machen?

a)
Automatisierung: Entlastung der Softwerker von administrativen und automatisierbaren Aufgaben
Graphische Oberfläche: Ergonomische und „natürliche“ Mensch/Maschine Schnittstelle
integrative Abdeckung des gesamten Lebenszyklus: keine Insellösungen
weitgehende "methodentreue" Unterstützung : Syntaktische und semantische Einschränkungen
Bereitstellung notwendiger Basisfunktionalität: z.B. Text- und graphische Editoren, Analyse-Werkzeuge
Bereitstellung effizienzsteigender Methoden: für die Bearbeitung (z.B. Zoom, cut-and-paste, automatische Indexe)
Bereitstellung qualitätssteigernder Funktionen: Regelbasis (Enzyklopädie), Querprüfung
intuitivie Bedienung: Selbsterklärung, Hilfetexte, einheitlicher „look and feel“, undo-Funktion, Individualisierbarkeit
Übernahme von Hilfs- und Routinearbeiten: „Software Assistenten“, Ratschläge, Rearrangieren von Graphiken
Bereitstellung von Import / Export Schnittstellen sowie definierter Datenaustauschformate: z.B. CDIF (Case Data Interchange Format)

b)
bla bla bla (soiche Fragen mag ich nicht)

FRAGE 71 (5):
Mit welchen Widerständen müssen Sie rechnen, wenn Sie in einer Firma ein (komplexes) CASE-Tool einführen?

Allgemeiner Widerstand gegen Neuerungen
Trägheit (Warum soll ich mich ändern, was neues lernen?)
Bequemlichkeit (wenn ich nicht mitarbeiten konnte, kann es nicht gut sein)
Vertrautheit mit altem (was gestern gut war kann heute nicht schlecht sein)
Misstrauen (was wird aus mir und meiner Stellung)

FRAGE 72 (5):
Beschreiben Sie die wesentlichen Aufgaben und Komponenten einer Software-Entwicklungsumgebung!
(das hier ist ein bisschen dürftig. Siehe 17-9)

Syntaxprüfung, semantische Prüfungen
etablieren und erzwinge Standards.
Konfigurations- und Versionsmanagement
Generatoren
Prozessmodellunterstützung
Jederzeitige Information

FRAGE 73 (5):
Nennen Sie 6 bis 7 unterschiedliche Argumente, warum Lines of Code (LOCs) kein gutes Maß für den notwendigen Aufwand für die Entwicklung eines Software-Produktes sind. Begründen Sie jedes Argument!

Die Komplexität der Implementierung (Rekursivität, ...) kann dadurch nicht ausgedrückt werden.
Die LOCs sind bei jeder Progammiersprache anderes.
Sie behandeln nur die Implementierung und berücksichtigen keine anderen Themen, wie komplexe Anforderungen, etc.
Sehr abhängig vom Programmierstil.
Es ist nicht generell definiert was gezählt wird (Kommentare, Deklarationen, Anweisungen, Leerzeilen,…). Macht jede Firma anders.
Misst eigentlich nur Größe für die Speicherung, oder für einen Ausdruck.
???

FRAGE 74 (5):
Wie kann die UML zur Beschreibung einer Software-Architektur eingesetzt werden? (dürfte das gleiche wie Frage 29 sein bzw. 30 im alten Katalog. Hab die Erklärung von dort rein kopiert.)

UML bietet spezielle Diagramme dafür an. Z.B. Komponentendiagramm, Verteilungsdiagramm, Pakete,…
Komponentendiagramm
Beschreibt Softwarekomponenten und ihre Abhängigkeiten. Pakete können Komponenten beinhalten oder umgekehrt.
Verteilungsdiagramm
Zeigt die aktuelle Hardware Konfiguration bestehend aus: Knoten (im Normalfall Prozessoren, I/O,…), SW-Komponenten, Prozessen, Objekten und den Kommunikationskanälen zwischen den Knoten.
Pakete:
Gruppiert eine beliebige Anzahl an UML-Modellelementen. Können selbst wieder Pakete enthalten. Ein UML-Modellelement kann höchstens zu einem Paket gehören. In verschiedenen Paketen können dieselben Namen verwendet werden.

FRAGE 75 (5):
Wie kann die UML zur Beschreibung der Anforderungen eingesetzt werden?

UML eignet sich nur für die Beschreibung von funktionalen Anforderungen (Use Case Diagramme können nur die darstellen). Nicht funktionale Anforderungen wie z.B. Qualität, Performance, Speicherverbrauch, Robustheit,… können nicht dargestellt werden.
Folgende Diagramme werden verwendet:
Use-Case-Diagramm (Actors...): ein Muster eines Verhaltens des Systems: jeder Anwendungsfall beschreibt eine Sequenz von zusammenhängenden Transaktionen die von Aktor und System in gegenseitigem Dialog durchgeführt werden.
Sequenzdiagramm (Zeitliche Abfolgen...): Interaktionen mit zeitlichem Ablauf.
Klassendiagramm (Methoden, Attribute...): zeigt die Existenz von Klassen und deren Beziehung in Systemsicht

FRAGE 76 (6):
a) Vergleichen Sie ein reines Zeichenwerkzeug (z.B. PowerPoint oder VISIO) mit einem gängigen, Ihnen bekannten Software-Modellierungswerkzeug (UPEDU) b) Nennen und beschreiben Sie fünf Vorteile des Modellierungswerkzeugs c)Wie verhält sich das Modellierungswerkzeug bezüglich ==>

a)
UPEDU liefert uns Verfahren und Vorgehensmodelle für einen Software Entwicklungsprozess. Es ist unabhängig von Software Tools. PowerPoint und VISIO sind Software Tools, die man verwenden kann um dem Kunden Ergebnisse zu präsentieren, bzw. um gewisse Diagramme zu erstellen.

b)
Durch die Modellierung wird es leichter für den Entwickler, die verschiedenen Aktivitäten und ihre Resultate zu beschreiben
Durch die Modellierung tritt einen Vereinfachung des Sachverhaltes ein. Somit wird das System leichter verständlich.
Vereinheitlichung, Verlässlichkeit und Produktivität.
???

FRAGE 77 (5):
Wie kann die UML zur Beschreibung des Systemverhaltens eingesetzt werden?

Siehe auch Frage 74. UML bietet Komponenten und Verteilungsdiagramme an, um zu visualisieren welche Komponenten im System wie zusammenspielen.