PS SE II SS2002

18. Juni 2002

Fragenkatalog

Report

Academic Supervisor Dr. Schwaiger Roland

Department of Computer Science
University of Salzburg

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften und Systemanalyse
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

18. Juni 2002

Zusammenfassung

Fragenkatalog zu PS SE II SS2002

Kapitel 1

Thema 1 - Projektmodell

1. [**Q**]Fragen zum Thema “Projektmodelle”
   [**A**]Was sind die wesentlichen Merkmale eines Projekts?
   Ob gross oder klein, ein Projekt hat folgende Merkmale:

   - Ein vorher festgelegter Output: Produkte oder Ergebnisse
   - Ein vorher festgelegter Anfang und ein festgelegtes Ende: konktrete Daten wann die Projektarbeit beginnt und wann sie endet
   - Festgelegte Budgets

     Welche Mengen an Personen, Finanzmitteln, Ausrüstungen, Produktionsmittel und Informationen sind notwendig

2. [**Q**]Inwiefern können sich diese Faktoren gegenseitig beeinflussen?
   [**A**]Jedes Element beeinflusst die Anderen. Will man den Output erhöhen muss man möglicherweise mehr Zeit (einen späteren Endtermin) oder mehr Resourcen investieren. Eine Verlegung des Endtermins macht möglicherweise eine Einschränkung bei den Ergebnissen oder eine Erhöhung der Preiskosten (über das festgelegte Budget hinaus) erforderlich weil Überstunden bezahlt werden müssen.

3. [**Q**]In der Projektmanagementpraxis existiert eine Vielzahl verschiedener Phasenmodelle, welche alle darauf ausgerichtet sind, den Projektablauf grob zu strukturieren. Aus diesen unterschiedlichen Modellen und Vorgehensweisen kann man ein einfaches Phasenmodell entwickeln, welches auf einen Grossteil der Projekte anwendbar ist. Nenne die grundlegenden Phasen eines jeden Projekts?
   [**A**]

   - Vorprojektphase
   - Planungsphase
   - Durchfhrungsphase
   - Abschlussphase

4. [**Q**]Was sind die drei bedeutenden Ziele die man in jedem Projekt definieren muss?
   [**A**]

- Sachziele
- Kostenziel
- Terminziel

müssen in jedem Projekt definiert werden und stehen im direkten Zusammenhang zueinander.

5. [**Q**] Was versteht man unter einer Situationsanalyse?
[**A**]Zu Beginn der Projektplanung muss die Ausgangssituation so weit wie möglich analysiert und diskutiert werden. Man analysiert die Ausgangssituation zunächst vom Groben ins Detail (top down).

   Dann werden die grossen Problemfelder und die daraus resultierenen Probleme analysiert.

6. [**Q**]Was ist die Aufgabe des Projektcontrollings?
[**A**]Die Aufgabe des Projektcontrollings liegt im Wesentlichen darin,

   - eventuelle Abweichungen zwischen Projektplanung und Projektverlauf frhzeitig vorherzusehen (agieren statt reagieren, Erkennen möglicher Trends im Projekt) bzw.
   - bereits eingetretene Abweichungen zu erkennen und rechtzeitig geeignete Gegenmanahmen und -Strategien einzuleiten.

7. [**Q**]Was sind die entscheidenden Phasen beim Wasserfallmodell?
[**A**]

   - Planungsphase
     Festlegung funktionaler und qualitativer Systemanforderungen
   - Analysephase
     Detailierte Definition, Beschreibung und Analyse der Systemanforderungen
   - Designphase
     Erstellung einer Softwaretechnischen Systemarchitektur
   - Realisierungsphase
     Softwaretechnische Realisierung des Produktes und Tests
   - Einführung / Wartung
     Abnahme und in Betriebnahme des Gesamtproduktes durch den Anwender, Wartung, Erweiterungen

8. [**Q**]Nenne mir einige Punkte die das Spirallmodell charakterisieren!
[**A**]

   - Benannte und standardisierte Entwicklungsschritte wie beim Wasserfallmodell.
   - Phasen werden zyklisch mehrfach durchlaufen bis das Produkt fertiggestellt ist

- Schwierigkeiten der initialen Anforderungsermittlung und Lernprozesse werden berücksichtigt

9. [**Q**]Welche zwei Dimensionen kann man in der Prozessstruktur des RUP ausmachen?
   [**A**]Die dynamische und die statische Dimension.

10. [**Q**]Ein entscheidendes Merkmal des RUP ist das Iterative Prinzip. Welche entscheidenden Vorteile bringt dieses Prinzip und welche entscheidenden Aktivitäten finden pro Iteration statt? Durch dieses Prinzip lassen sich Fehler schnell finden und kostengünstig beheben. Innerhalb einer solchen Iteration findet man folgende Aktivitäten.

    - Konzeptionsphase
    - Design/Entwurf
    - Implementierung/Test
    - Freigabe des Produktes

11. [**Q**] Was versteht man im Allgemeinen unter einem Projekt?
    [**A**] ein Vorhaben, etwa innerhalb eines Unternehmens, das im Wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist; dazu zählen zum Beispiel: Zielvorgabe, äußere Rahmenbedingungen (Zeitvorgabe, finanzielle Ressourcen), Abgrenzung nach außen, projektspezifische Organisation

12. [**Q**] Was versteht man unter dem Begriff Projektmodell?
    [**A**] Abstraktion eines Projekts, die anhand einer allgemeinen Struktur Ablauf, Organisation und Durchführung eines Projekts beschreibt; ein Projektmodell beschreibt eine Folge von Aktivitäten, an deren Ende ein Produktërzeugt wird

13. [**Q**] Man nenne einige Kriterien zur Charakterisierung von Software-Projekten!
    [**A**] Projektumfang(Dauer, finanzieller Aufwand, Anzahl der Mitarbeiter), Anwendungsgebiet, Ausmaß der erwarteten Anforderungsänderungen, Unterschied zwischen Neuentwicklung und Änderung bestehender Software, Innovationsgrad

14. [**Q**] Was versteht man unter Projektphasen?
    [**A**] Einteilung in Phasen erfolgt zur Strukturierung des Entwicklungsprozesses; Projektphase beschreibt Abschnitt im Projekt, der in sich zusammenhängend und von anderen Abschnitten abgegrenzt ist → Menge von Aktivitäten, die auf gemeinsamer Grundlage aufbauen (Eingangssituation) und auf ein gemeinsames Ziel ausgerichtet sind

15. [**Q**] Wodurch unterscheiden sich Projektphasen im Wasserfallmodell von denen in iterativen / inkrementellen Projektmodellen?
    [**A**] Wasserfallmodell: Analyse, Design, Implementierung etc. als Phase → sequentielles Vorgehen; iterative / inkrementelle Modelle: eine Ausbaustufe des Programms als Projektphase → Elemente wie Analyse, Design, Implementierung etc. sind in jeder Phase enthalten

16. [**Q**] Für welche Art von Projekten lässt sich das Wasserfallmodell erfolgreich einsetzen (d.h. welche Voraussetzungen sollten erfüllt sein)?
[**A**] gedacht für große, umfangreiche Projekte mit vielen Mitarbeitern und komplexer Organisation (räumliche Trennung, viele beteiligte Stellen etc.); Projekte mit stabilen und bekannten Anforderungen, die ausführlich dokumentiert werden können; Projekte, die sich vor allem mit der Neuentwicklung von Software beschäftigen

17. [**Q**] Wie lassen sich die wesentlichen Vor- und Nachteile des Wasserfallmodells charakterisieren?
[**A**] Vorteile: klare Definition der Anforderungen, exakte Spezifikation; dadurch fundierte Basis für Vertragsabschluss (Pflichtenheft); Projektfortschritt und Projekterfolg sind gut messbar; Ansatzpunkt für funktionierende Qualitätskontrolle
Nachteile / Hauptproblem: exakte Spezifikation oft nicht realisierbar (unklare Anforderungen); Schwierigkeiten bei der Handhabung von Anforderungsänderungen (Spezifikationsfalle); nur bedingt geeignet für die Änderung bestehender Programme

18. [**Q**] Man beschreibe die grundlegende Struktur im Spiralmodell.
[**A**] Risikiominimierung durch inkrementelle Entwicklung; Entwicklungsprozess wird anhand einer Spirale mit Schleifen anstatt einer Sequenz von Phasen symbolisiert; jede Schleife steht für eine Projektphase mit Zielbestimmung, Risikoanalyse, Entwicklung und Planung der nächsten Phase

19. [**Q**] Worin liegt der wesentliche Unterschied zwischen Agilen Prozessen und "konventionellen" Projektmodellen? Welches Problem kann sich daraus in der Praxis für die Organisation ergeben?
[**A**] Unterschied: Verzicht auf alle Designdokumente (statische Programmbeschreibungen); einzige Dokumentation ist der Source-Code; daraus ergeben sich besondere Forderungen an die Gestaltung des Programmcodes;
Problem: häufig wird gerade im Bereich der Qualitätssicherung umfangreiche Projektdokumentation gefordert (ISO-Normen), durch fehlende Dokumente wird der "formale" Vertragsabschluss erschwert

20. [**Q**] Welche Voraussetzungen sollten für das gelingen agiler Prozesse gegeben sein?
[**A**] keine allzu großen Projekte (max. 15 Personen); alle Entwickler am selben Ort; Änderungskosten dürfen nicht zu rasch wachsen; Bereitschaft des Kunden zur ständigen Zusammenarbeit

Kapitel 2

Thema 3 - TOLEDO vs. UML

1. [**Q**] Was zeichnet eine gute Anforderungsspezifikation aus?
   [**A**] Adäquatheit (das beschreiben, was der Kunde will bzw braucht), Vollständigkeit (alles beschreiben, was der Kunde will bzw braucht), Widerspruchsfreiheit (sonst ist die Spezifikation nicht realisierbar), Verständlichkeit (für den Kunden und für die Softwareersteller), Eindeutigkeit (damit Fehler durch Fehlinterpretationen vermieden werden). Prüfbarkeit (damit feststellbar ist, ob das realisierte System die Anforderungen erfüllt)

2. [**Q**] Erklären Sie die folgenden OO-Begriffe: Objekt, Klasse.
   [**A**] Objekt ist ein Gegenstand der Betrachtung. Das können physikalische Dinge sein wie Auto, Haus, Computer, können aber auch Gedanken, Wahrnehmungen, Gefühle usw sein. Objekte haben Eigenschaften bzw Zustände, die Attribute und haben Operationen, die Methoden. Objekte mit gleichen Attributen und Operationen können zu einer Klasse zusammengefaßt werden.

3. [**Q**] Erklären Sie den OO-Begriff Instanz.
   [**A**] Die zu einer Klasse gehörenden Objekte werden als Instanzen einer Klasse bezeichnet.

4. [**Q**] Wann nennt man eine Klasse abstrakt.
   [**A**] Besitzt eine Klasse keine Objekte, so nennt man sie abstrakte Klasse.

5. [**Q**] Erklären Sie die folgenden OO-Begriffe: Assoziation, Aggregation, Komposition.
   [**A**] Eine Assoziation beschreibt eine Beziehung zwischen Klassen bzw deren Instanzen. Durch die Verbindung über eine Assoziation können die Objekte der verbundenen Klassen miteinander kommunizieren. Eine Aggregation ist eine Sonderform der Assoziation, bei der die beteiligten Klassen keine gleichwertige Beziehung führen, sondern eine Ganzes-Teile-Hierarchie darstellen. Eine Aggregation beschreibt, wie sich etwas Ganzes (Aggregat) aus seinen Teilen zusammensetzt. Die Komposition ist eine strenge Form der Aggregation. Hier sind die Teilobjekte abhängig von ihrem Aggregat. Ohne Aggregat kann auch kein Teilobjekt existieren.

6. [**Q**] Was versteht man unter Toledo?
   [**A**] Toledo ist eine objektorientierte Spezifikationsmethode und wurde von Mitarbeitern der Firma sd&m entwickelt. Toledo ist textorientiert und hat eine einheitliche

Struktur. Sie besteht teilweise aus formalen Notationen - also aus genau definierten Schlüsselwörtern, teilweise aus formfreien Beschreibungen.

7. [**Q**] Zählen Sie einige UML-Diagramme auf.
[**A**] Use-Case-Diagramm, Klassendiagramm, Aktivitätendiagramm, Kollaborationsdiagramm, Sequenzdiagramm, Zustandsdiagramm, Komponentendiagramm, Verteilungsdiagramm

8. [**Q**] Erklären Sie kurz das Use-Case-Diagramm von UML.
[**A**] In Use-Case-Diagrammen (Anwendungsfalldiagrammen) werden die funktionalen Anforderungen eines Systems festgelegt. Es besteht aus den Anwendungsfällen, den Akteuren und den Beziehungen.

9. [**Q**] Was kann man aus den Klassendiagramm von UML ersehen?
[**A**] In einem Klassendiagramm kann man folgendes sehen: Klassen bzw Objekte mit ihren eindeutigen Bezeichnungen, deren Attribute (Bezeichnung, Datentyp, Zusicherungen) und Methoden, die Sichtbarkeit von Attributen und Methoden, Zuständigkeiten, Ausnahmen, Vererbungsbeziehungen, Assoziationen mit Kardinalitäten.

10. [**Q**] Wie wird eine Vererbung in einem Klassendiagramm dargestellt?
[**A**] Eine Vererbungsbeziehung wird durch eine Verbindungslinie mit einem Pfeil von der Unterklasse zur Oberklasse symbolisiert. Der Pfeil zeigt dabei in Richtung Oberklasse.

11. [**Q**] Wozu dient eine Spezifikationsphase in einem Softwareentwicklungsprozess?
[**A**] Die Spezifikationssphase dient zur Festlegung der Anforderungen an das künftige System. Sie liefert ein Dokument, welches als Grundlage für die Implementierung des Systems, sowie für das Testen vom fertigen System benutzt wird.

12. [**Q**] Wie wird die Kardinalität einer Beziehung in der TOLEDO-Notation dargestellt?
[**A**] Die Kardinalität einer Beziehung wird durch Doppelpunkte, bzw. Pfeile ausgedrückt (abhängig von der Beziehungsart: Aggregation oder Verweis). Zum Beispiel:

    - ‘:’ bezeichnet eine 1:1 Beziehung;
    - ‘–[1..5]->’ bezeichnet eine 1:1..5 Beziehung.

13. [**Q**] Wozu dient der LEBENSLAUF-Abschnitt in einer TOLEDO-Notation?
[**A**] Der LEBENSLAUF-Abschnitt in einer TOLEDO-Notation dient zur Darstellung von Zuständen die ein Objekt annehmen kann und der Zustandsübergängen.

14. [**Q**] Durch welche reservierte Wörter werden die Operationen in einer TOLEDO-Objekttypbeschreibung eingeleitet?
[**A**] Operationen in einer TOLEDO-Objekttypbeschreibung werden durch KANN, KANN_MAN oder KANN_SICH eingeleitet.

15. [**Q**] In welche Gruppen werden die UML Diagramme unterteilt?
[**A**] Die UML Diagramme werden in zwei Gruppen unterteilt:

    - Strukturdiagramme - beschreiben die dynamischen Schichten eines Systems und
    - Verhaltensdiagramme - beschreiben die statischen Schichten eines Systems.

16. [**Q**] Welche UML-Diagramme werden als Verhaltensdiagramme bezeichnet?
[**A**] Als Verhaltensdiagramme werden bezeichnet:

    - Anwendungsfalldiagramm,
    - Sequenzdiagramm,
    - Kollaborationsdiagramm,
    - Zustandsdiagramm und
    - Aktivitätsdiagramm.

17. [**Q**] Welche Beziehungstypen unterscheidet man bei einem UML Klassendiagramm?
[**A**] Bei einem UML Klassendiagramm unterscheidet man folgende Beziehungstypen:

    - Assoziation - allgemeine Beziehung zwischen zwei Klassen, z.B. “gehört”, “redet mit”.
    - Aggregation - “Ist-Teil-von” (“besteht aus”) Beziehung, Beziehung zwischen einem Ganzen und seinen Teilen.
    - Komposition - stärkere Form der Aggregation, realisiert physikalisches Enthaltensein.
    - Vererbung - Generalisierung (“ist ein”), Beziehung zwischen (spezieller) Unterklase und (allgemeiner) Oberklasse.

18. [**Q**] Wodurch unterscheiden sich ein Sequenz- und ein Kollaborationsdiagramm?
[**A**] Ein Sequenzdiagramm beschreibt die zeitliche Abfolge von Nachrichten zwischen Objekten, d.h. der zeitliche Ablauf steht im Vordergrund; dagegen ein Kollaborationsdiagramm beschreibt die Verantwortlichkeiten der einzelnen Objekten an den Interaktionen, d.h. der zeitliche Ablauf steht im Hintergrund.

19. [**Q**] Nennen Sie einige Vorteile der TOLEDO-Notation gegenüber UML.
[**A**] Vorteile sind:

    - TOLEDO ist offen - es ist leicht zu erweitern um eigene Elemente.
    - TOLEDO-Texte eignen sich gut für die Kommunikation mit dem Anwender.
    - Auswertung der TOLEDO-Texte mit einfachen Werkzeugen.
    - Einfache Versionsführung mit den Standardwerkzeugen (z.B. mit rcs, cvs).

20. [**Q**] Nennen Sie einige Vorteile der UML gegenüber TOLEDO.
[**A**] Vorteile sind:

    - Sehr großer Umfang der Sprache.
    Viele Aspekte einer Spezifikation (Analyse-, Entwurfs- und Implementierungsentscheidungen) werden durch die UML-Diagramme abgedeckt.
    - Übersichtlichkeit und Verständlichkeit der Diagramme.
    - UML ist ein Standard.
    - Viele Produkte unterstützen UML.
    - UML ist sehr verbreitet.

- Einsatzmöglichkeiten.
  UML wird auch zur Modellierung außerhalb der Software Systeme verwendet, z.B zur Modellirung von Arbeitsläufen im Rechtsystem.

Kapitel 3

Thema 4 - Dialogoberflächen

1. **[Q]** Was versteht man unter *Usability*?
   [A] Die Bedienbarkeit und Benutzerfreundlichkeit eines Systems. Den Grad, wie schnell sich ein (neuer) User mit einer Oberfläche zurechtfindet und wie praktisch und effizient er damit arbeiten kann.

2. **[Q]** Welche drei Haupt-Prinzipien beeinflussen die Usability?
   [A]

   Learnability (Erlernbarkeit) bestimmt, wie einfach und schnell sich neue Benutzer in der Umgebung zurechtfinden.

   Flexibility (Flexibilität) befasst sich mit der Vielfalt an Möglichkeiten, mit der Benutzer und System Infomationen austauschen können.

   Robustness (Robustheit) bestimmt, inwieweit der Anwender vom System dabei unterstützt wird, seine Ziele zu bestimmen und dann anschließend auch zu erreichen (der Umgang mit Fehlern ist hierbei eingeschlossen).

3. **[Q]** Was sind Metaphern und wie helfen sie dem Benutzer beim Umgang mit dem System?
   [A] Metaphern sind Bezeichnungen/Darstellungen von realen, dem Benutzer vertrauten Dingen, die als Sinnbilder das Auffinden und Verstehen der System-Funktionen erleichtern sollen. (z.B. Papierkorb zum Löschen von Dateien)

4. **[Q]** Was ist *Chunking*, wieso wendet man es an?
   [A] Die Kapazität des Kurzzeitgedächtnis ist sehr beschränkt (7$\pm$2 Einträge). Die Leistung des KZG lässt sich aber erhöhen, wenn man die gebotene Information in grösserer (semantisch bedeutsame) Einheiten, die Chunks, aufteilt. (z.B. 121014921830 lässt sich schwerer erfassen/merken, als 12.10.1492 18:30)

5. **[Q]** Was versteht man unter *Primacy-* und *Recency Effect*?
   [A] Wenn man dem User eine Liste von Informationen, eine Abfolge von einzelenen Fenstern,... präsentiert, so wir er sich in der Regel jene merken, die am Anfang und die am Ende stehen. Die in der Mitte werden vergessen.

6. **[Q]** Was sind die wichtigsten Punkte bei der farblichen Gestaltung einer Benutzerschnittstelle?
   [A]

- Max. vier Farben verwenden.
- Diese sollen kontrastreich aber harmonierend sein (→ besonders gut: Farbvierklänge).
- Gleiche Sachverhalte durchgehend mit den gleichen Farben darstellen.
- Beim Design nicht vergessen, dass es Benutzer gibt, deren Farbsehen beeinträchtigt ist. (entweder durch monochrome Bildschirme, oder Farbenblindheit,...)

7. **[Q]** Was sind die wichtigsten Punkte bezüglich Einsatz von Schriften in Benutzerschnittstellen?
   [A]
   - Nur wenige verschiede Schriftarten verwenden.
   - Ein Mixen von serifen und sans-serifen Schriftarten vermeiden.
   - Sparsamer Umgang mit fettem und kursivem Text.
   - Min. 10pt Schriftgrösse. (Wert stark abhängig von Einsatzumgebung → z.B. Infoterminal in Altersheim)

8. **[Q]** Welche vier Prinzipien sind wichtig für das Usability Engineering?
   [A]

   Early Focus on Users Schon von Anfang an sollte der Entwickler stets einen direkten Kontakt mit den späteren Usern pflegen, um etwas über die Charakteristiken der Personen und ihrer Arbeitsumgebung zu erfahren.

   Integrated Design Sämtliche Bestandteile des Benutzer Interfaces (wie die Oberfläche, das Hilfe-System, die Dokumentation,...) sollten parallel Entwickelt werden.

   Early And Continual User Testing Um ein benutzerorientiertes Interface entwerfen zu können, muss man im Vorfeld schon „empirisch“vorgehen. Man muss den zuküntigen User analysieren, das gesammelte Feedback sorgfältig auswerten,... und aus dem gesammelten Material eine wohl durchdachte Systemlösung entwickeln.

   Itertive Design Das Projekt muss während der Entwicklungsphase immer wieder auf seine Usability hin getestet werden, um es mit den gewonnenen Erkenntnissen ständig iterativ verbessern zu können.

9. **[Q]** Was versteht man unter *Participatory Design*?
   [A] Participatory Design beruht auf der Idee, dass die zukünftigen User die Domain-Experts sind. Sie wissen am besten, welche Arbeitsschritte ihre Aufgaben umfassen, wo sie sich Computerunterstützung wünschen,... Aus diesem Grund werden die Benutzer beim Participatory Desing in den gesamten Design- und Entwicklungsprozess „ihres“Interfaces miteinbezogen.

10. **[Q]** Was sind die Vor- und Nachteile eines CLI (Command Line Interface)?
    [A] **Vorteile**:
    - Der geübte Benutzer kann sehr schnell und effizient arbeiten.

- Es sind mächtige Befehle realisierbar.
- Aktionen über viele Objekte lassen sich leicht durchführen.

Nachteile:

- Hoher Lernaufwand, außerdem ist eine Nutzung ohne Kenntnis der Syntax nicht möglich.
- Die Eingabe ist fehleranfällig, da Befehle und Parameter errinnert werden müssen.
- Die Auswirkungen von Fehlern können beträchtlich sein.
- Die Auswirkung eines Befehls rückgängig zu machen ist nur in Ausnahmen möglich.

Kapitel 4

Thema 5 - Prototypen

1. [**Q**] Was versteht man unter Prototypen?
   [**A**] Ein Software-Prototyp ist ein mit wesentlich geringerem Aufwand als das geplante Produkt hergestelltes und zu erweiterndes, ausfhrbares Modell des geplanten Software-Produkts, das nicht notwendigerweise alle Eigenschaften des Zielsystems besitzen muss, jedoch so geartet ist, dass vor der eigentlichen Systemimplementierung der Anwender die wesentlichen Systemeigenschaften erproben kann.

2. [**Q**] Was ist der Unterschied zwischen einem horizontalen und vertikalen Prototyp (Skizze). Beschreiben Sie kurz beide Varianten.
   [**A**] Beim horizontalen Prototyp wird jeweils eine einzelne Schicht der Systemarchitektur (z.B. GUI, Datenbank) implementiert. So wird zum Beispiel die komplette Benutzeroberflche, oder nur der funktionale Teil des Programms entwickelt.
   Im Gegensatz dazu wird beim vertikalen Prototyping stets ein Teil des Anwendungssystems ber alle Schichten hinweg realisiert.

Abbildung 4.1: Beispiel einer Systemarchitektur

3. [**Q**] Erklren Sie die explorative Entwicklung von Prototypen.
   [**A**] Ziele der explorativen Entwicklung von Prototypen sind ein besseres Problemverstndnis und das Aufdecken von Spezifikationslcken. Deshalb wird diese Art des Prototyping hufig in der Phase der Anforderungsdefinition verwendet. Hier erarbeiten Anwender und Entwickler gemeinsam die Systemfunktionen, damit die Entwickler einen Einblick in den Anwendungsbereich bekommen und gemeinsam mit den Anwendern verschiedene Lsungsmglichkeiten diskutieren knnen. Verwendung beim explorativen Prototyping findet hauptschlich der horizontale Prototyp.

4. [**Q**] Erklren Sie die experimentelle Entwicklung von Prototypen.
   [**A**] Ziele bei dieser Entwicklung sind die berprfung von Architekturmodellen, Teilentwrfe und Lsungsideen und Machbarkeitsstudien. An der Entwicklung dieses Prototypen sind nur die Software-Entwickler beteiligt. Zur Erprobung der verschiedenen Lsungsmethoden wird oft ein Durchstich verwendet. Anhand des Durchstichs kann man z.B. die Wechselwirkung zwischen den Systemkomponenten simulieren, oder aber auch neue Software-Produkte testen. Die Beurteilung der Ergebnisse dient als Entscheidungsgrundlage fr die Risikoabschtzung.

5. **[Q]** Erklren Sie die evolutionre Entwicklung von Prototypen.
   [A] Der evolutionre Ansatz bedeutet eine Systementwicklung in mehreren Stufen. Die erste Version realisiert die unstrittigen Benutzeranforderungen und jede weitere Folgestufe integriert weitere Anforderungen. Wichtig dabei ist, dass nach jeder Entwicklungsstufe eine Validierung erfolgt und die nchsten Schritte geplant werden. Bei dieser Methode kann sehr flexibel auf neue Anforderungen reagiert werden, was zum einen vorteilhaft ist und zum anderen eine Erschwerung des Systemdesigns durch eine sich stndig erweiternde Systemspezifikation zur Folge haben kann.

6. **[Q]** Was ist ein Demonstrationsprototyp?
   [A] Der Demonstrationsprototyp dient zur Auftragsakquisitation und verschafft einen Eindruck, wie das Produkt aussehen kann. Er vermittelt Ideen fr die Oberflche und fr die Mglichkeiten des neuen Systems. Weiters soll er dem Auftraggeber und dem Hersteller des Systems helfen, eine Vorstellung zu entwickeln, wie die Benutzer mit dem zuknftigen System arbeiten. Auerdem kann der Demonstrationsprototyp die Entscheidungstrger davon berzeugen, dass eine neue Anwendung sinnvoll ist und Nutzen bringt.

7. **[Q]** Was ist ein Labormuster bzw. ein Durchstich?
   [A] Mit Hilfe dieser Prototypen sollen bestimmte technische Details auf ihre Realisierbarkeit getestet werden. Das Labormuster eignet sich zum Testen einer Schnittstelle, zur Vorbereitung der Entwicklungsumgebung, zur Bewertung von neuen Software-Werkzeugen oder zum Ausprobieren neuer Algorithmen. Die Aufgabe des Durchstichs ist die Erprobung des Datenverarbeitungskonzepts. Whrend das Labormuster eine Teillsung ausprobiert, untersucht der Durchstich die Gesamtlsung. Dazu implementiert man einen schmalen Ausschnitt der Funktionen, also einen vertikalen Prototypen. Der Durchstich ist dann zu empfehlen, wenn bei der Systemarchitektur Neuland betreten wird.

8. **[Q]** Welcher Nutzen ergibt sich bei Verwendung von Prototypen?
   [A] Prototypen helfen beim Lsen typischer Probleme traditioneller Modelle: z.B. Der Auftraggeber kann oft die Anforderungen nicht explizit oder vollstndig formulierien. Im Prototypen-Modell schafft der Spezifikationsprototyp abhilfe.
   Weiters schafft das gemeinsame - Anwender und Entwickler - Entwickeln von Prototypen eine gemeinsame Sprache und ein gemeinsames Verstndnis des Problems. Diese Kommunikationsbasis erleichtert die Zusammenarbeit und vermeidet kostspielige Miverstndnisse.
   Prototypen knnen helfen, Risiken frh zu erkennen und zu beseitigen - die bekannten wie die unvermuteten.
   Oft existieren unterschiedliche Lsungswege, die besser experimentell erprobt werden und mit dem Auftraggeber diskutiert werden knnen.
   Auch lassen sich manche Anforderungen theoretisch nicht garantieren und sollten einer Machbarkeitsstudie - z.B. mittels Durchstich oder Labormuster - unterzogen werden.

9. **[Q]** Vorgehensweise bei der Erstellung eines Prototyp-Projekts.
   [A] Jeder Prototyp ist ein Projekt und wird genauso systematisch entwickelt wie eine Anwendung. Fr deb Prototyp gelten alle Regeln eines Projektes:

Auftrag : Der Projektauftrag fixiert schriftlich die Ziele des Prototypen. Er bestimmt den Zeitraum und das Budget.
Entwurf : Auch ein Prototyp verdient einen guten Entwurf, welcher in einem Datenverarbeitungskonzept beschrieben wird.
Realisierung : Programmrichtlinien gelten auch fr einen Prototypen. Alle Dokumente unterliegen der Versionskontrolle.
Evaluierung : Demonstrations-, Oberflchen- bzw. Spezifikationsprototypen werden an die zuknftigen Anwender zur Evaluierung weitergegeben. Die Ergebnisse und die gewonnene Erfahrung werden dokumentiert, indem man das Datenverarbeitungskonzept aktualisiert und beschreibt, welche Lsungen untersucht wurden und mit welchem Ergebnis.
Weggwerfenoderweiterentwickeln? Die frhe Entscheidung - Wegwerfprototyp oder wiederverwendbarer Prototyp - ist besonders wichtig fr das Projekt, unabhngig von der Art des Prototyps.

10. [**Q**] Beschreiben Sie kurz das Verhlnis: Prototyp und Anwendungssystem.
[**A**] Der Prototyp gehrt zur Spezifikation des Anwendungssystems. Diese Prototypen werden so schnell wie mglich konstruiert und realisieren meistens nur Aspekte der Benutzeroberflche. Ein Prototyp kann aber auch zum Anwendungssystem ausgebaut werden. Hierbei wird der Prototyp schrittweise in das Zielsystem berfhrt. Prototypen knnen aber auch ohne Verbindung zu einem konkreten Entwicklungsobjekt erstellt werden. Diese sollen lediglich Fragen klren.

11. [**Q**] Zählen Sie die drei Arten von Softwareprototypen auf?
[**A**] Explorativer-, Experiementeller- und Evolutionaärer-Prototyp

12. [**Q**] Beschreiben Sie kurz die Aufgabe des Rapid Throwaway Prototyping?
[**A**] Erweitert den Ansatz der Anforderungsanalyse mit dem Ziel einer Kostenreduzierung für den gesamten Lebenszyklus. Die wesentliche Funktion des Prototyps besteht in der Bestimmung der Anforderungen, um Prozessrisiken abschätzen zu können. Nach der Bewertung wird der Prototyp weggeworfen.

13. [**Q**] Beschreiben Sie kurz die Aufgabe eines experimentellen Prototypen?
[**A**] Dieser Prototyp dient dazu, die Tauglichkeit von Teilspezifikationen, Architekturmodellen und Lösungsideen für einzelne Systemkomponenten experimentell nachzuweisen. Ziel dieser Vorgehensweise ist eine volständige Spezifikation von Teilsystemen die als Grundlage für die Implementierung des Enprodukts dienen soll.

14. [**Q**] Beschreiben Sie kurz die Aufgabe eines evolutionären Prototyps?
[**A**] Das Evolutionary Prototyping basiert auf dem Prinzip, zunächst eine erste Implementierung zu entwickeln, diese dem Kunden zur Kommentierung vorzustellen und sie anschließ;end zahlreichen Schritten immer weiter zu verfeinern, bis eine angemessene Systemlösung erreicht ist.

15. [**Q**] Welche Hauptvorteile des Evolutionary Prototyping kennen Sie?(mit Beschreibung)
[**A**]

(a) *Beschleunigte Fertigstellung des Systems:* Die Software muß; schnell verfügbar sein, oft ist es wichtiger, dass Software schnell verfügbar ist als die Details der Funktionalität oder Wartbarkeit.

(b) *Die Identifikation des Benutzers mit dem System:* Einbindung des Benutzers ins System erhöt die Wahrscheinlichkeit, dass Anforderungen erfüllt werden und dass der Benutzer Interesse daran hat, dass das System funktioniert.

16. [**Q**] Beschreiben Sie kurz die Technik des RAD?
[**A**] RAD wird als Prozess definiert der es erlaubt Systeme in weniger als 60-90 Tagen zu entwickeln, allerdings mit einigen Kompromissen. Es gibt mehrere Prinzipien die hinter dieser Definition stecken.

    - In einigen Situationen, kann eine 80%ige Lösung in 20% der Zeit entwickelt werden, die für eine komplette Lösung benötigt werden würde.
    - In einigen Fällen können die business Anforderungen für ein System voll erfüllt werden, obwohl einige Anforderungen für die Betriebsfähigkeit nicht erfüllt sind.
    - In einigen Fällen, kann ein System, mit weniger als den vereinbarten Anforderungen, akzeptiert werden.

17. [**Q**] Beschreiben Sie kurz die Technik des JAD?
[**A**] Application Development (JAD) ist ein Managementprozess, welcher Entwicklern hilft effektiv mit Benutztern informationstechnologiesche Lösungen zu entwickeln die wirklich funktionieren. Das Ziel von JAD ist es ein Projekt zu definieren, eine Lösung zu designen und es bis zu seinem Abschluss zu überwachen.

18. [**Q**] Welche Aktivitäten unterstützt ein Prototyp in der Anforderungsbestimmung?
[**A**] Systemprototypen erlauben es den Benutzern, die entstehende Anwendung auszuprobieren, um festzustellen, wie das System ihre Arbeit unterstützt.Sie gewinnen dadurch neue Ideen für Anforderungen und können Stärken und Schwächen in der Software aufspüren. Sie können dann neue Systemanforderungen vorschlagen.

19. [**Q**] Wie unterstützt ein Prototyp die Validierung von Anforderungen?
[**A**] Der Ptototyp kann Fehler und Lücken in den vorgeschlagenen Anforderungen aufdecken. Eine in einer Spezifikation beschriebene Funktion mag sinnvoll und genau definiert erscheinen. Wenn diese Funktion jedoch mit anderen zusammenwirkt, stellen die Benutzer häufig fest, dass ihre anfängliche Sicht der Dinge falsch oder unvollständig war. Die Systemspezifikation kann in diesem Falle an das veränderte Verständins der Anforderungen angepasst werden.

20. [**Q**] Geben Sie eine Genaue Definition eines Software Prototyps?
[**A**] Ein Software-Prototyp ist ein - mit wesentlich geringerem Aufwand als das geplante Produkt - hergestelltes und erweiterndes, ausführbares Modell des geplanten Software-Produkts, das nicht notwendigerweise alle Eigenschaften des Zielsystems besitzen muss, jedoch so geartet ist, dass vor der eigentlichen Systemimplementierung der Anwender die wesentlichen Sytemeigenschaften erpoben kann.

Kapitel 5

Thema 7 - Architektur verteilter Systeme

1. [**Q**] Define the term "Distributed System".
   [**A**] A distributed system is a set of physically separate processors connected by one or more communication links.

2. [**Q**] Explain the difference between *loosely* and *tightly coupled* systems.
   [**A**] In a tightly coupled system the processors share clock and memory and run one operating system. In a loosely coupled system, they have their own clock and memory and each runs its own instance of an operating system.

3. [**Q**] Name three goals of distributing software.
   [**A**] Choose three of:

   - Decentralization of data and functions of an overall application
   - Cooperation of distributed processing units
   - Improvement of locality-properties and efficiency of an application
   - Integration of so far separate distributed applications
   - Distributed access to special resources
   - Improvement of fault-tolerance and availability of an application

4. [**Q**] Software architecture can be viewed from an external and an internal point of view. Name three "internal views" of a software architecture.
   [**A**] Choose three of:

   - Component View
   - Administrative View
   - "Building" View
   - Physical View
   - Run-Time View

5. [**Q**] There are several hard criteria to measure the quality of a software architecture by. Name three of them.
   [**A**] Choose three of:

- Performance
- Security
- Availability and Reliability
- Robustness
- Range of Functions
- Usability

6. **[Q]** Define “framework”.
   [A] A framework is an integrated collection of classes that collaborate to produce a reusable architecture for a family of related applications.

7. **[Q]** Define “component”.
   [A] A component is an encapsulation unit with one or more interfaces that provide clients with access to its services.

8. **[Q]** Shortly describe the Client/Server model.
   [A] In the Client/Server model, the software system is split between the server and the client. The client sends requests according to some protocol, asking for information or action, and the server responds.

9. **[Q]** Formulate the basic concept of the *peer-to-peer* model.
   [A] In a peer-to-peer application, the software is distributed among machines that all have the same capabilities and responsibilities.

10. **[Q]** Shortly explain the term “separation of concerns” in the context of software architecture.
    [A] “Separation of conecerns” means that a single software component is concerned only with a single aspect of the overall application.

11. **[Q]** Welche zwei technologischen Entwicklungen schufen die Hardware-Basis für den Aufbau verteilter Systeme?

    [A] Dies war zum einen die Produktion leistungsfähiger Mikroprozessoren in großer Stückzahl und damit zu sehr günstigen Preisen und zum anderen die Entwicklung leistungsfähiger und kostengünstiger Netzwerke.

12. **[Q]** Welche Haupttypen von Systemen unterscheidet Sommerville?

    [A] Persönliche, eingebettete und verteilte Systeme.

13. **[Q]** Was sind die wesentlichen Elemente eines verteilten Systems in der Definition nach Coulouris et.al?

    [A] Dies sind Komponenten, welche auf vernetzten Computern installiert sind, wobei sich diese nur durch den Austausch von Nachrichten koordinieren bzw. darüber miteinander kommunizieren.

14. **[Q]** Was wird als die zentrale Aufgabe der SW-Architektur als eigenständige Informatik-Disziplin gesehen?

    [A]Die Reduktion der Komplexität oder zumindest Beherrschbarkeit großer verteilter SW-Systeme.

15. [**Q**] Wie ließe sich, trotz in der Literatur mehr oder weniger abweichender Definition von SW-Architektur, das gemeinsame Grundverständnis von SW-Architektur formulieren?

    [**A**] Eine Systemarchitektur stellt ein Modell zur Verfügung, welches Implementierungsdetails unterdrückt. Dadurch kann sich ein Systemarchitekt auf die Analyse und jene Entscheidungen konzentrieren um das System so zu strukturieren, dass es den gestellten Anforderungen entspricht.

16. [**Q**] Was sind die Schlüsselfragen auf dem Abstraktionsniveau der SW-Architektur?

    [**A**] Die Grobunterteilung eines Systems in interagierende Subsysteme, die Zuteilung von Funktionen an Recheneinheiten (computational units), die Kommunikationsprotokolle zwischen diesen Einheiten, globale Systemeigenschaften wie Durchsatz und Latenzzeit und Lebenszyklusfragen des zu konstruierenden Systems wie Wartbarkeit, Wiederverwendbarkeit, Skalierbarkeit, Platformunabhängigkeit

17. [**Q**] Was sind die drei von einer ADL (Architecture Description Language) unabhängigen syntaktischen Basiselemente zur Darstellung eines Architektur-Styles?

    [**A**] Es handelt sich dabei um Komponenten (components), Konnektoren (connectors; zwischen den components) und Konfigurationen von Komponenten und Konnektoren (configurations).

18. [**Q**] In welche zwei großen Klassen kann die Architektur verteilter Systeme unterteilt werden?

    [**A**]Logische Architekturen und technische Architekturen

19. [**Q**] Welchem der drei syntaktischen Basiselemente zur Darstellung eines Architektur-Styles sind in einem Client/Server Architektur-Style der Client bzw. der Server zuzuordnen und mit Hilfe welcher Methode kommunizieren diese miteinander?

    [**A**] Das syntaktische Basiselement ist die Component und Client bzw. Server kommunizieren mittels RPCs (Remote Procedure Calls) miteinander.

20. [**Q**] Nennen Sie die beiden syntaktischen Basiselemente aus denen Konfigurationen in [**A**]Styles gebildet werden und mindestens eine Bedeutung (Interpretation) die ihnen zugeordnet wird.

    [**A**]Die beiden syntaktischen Basiselemente sind Components und Connectors. Mögliche Bedeutungen von Components sind Programme, Prozesse oder Filter bzw. Remote Procedure Calls, Message-passing Protokolle oder Datenströme (data streams wie z.B. Unix-Pipes).

Kapitel 6

Thema 9 - Software Entwicklungs Umgebungen

1. **Die Entwicklung von nur-text-basierten User-Interfaces zu Grafischen User-Interfaces (GUI) war ein enormer Fortschritt. Was sind die wesentlichen Vorteile von GUIs?**

   - Informationsdarstellung mit Hilfe von Grafiken, Interaktion durch Optionen und Schaltflächen anstatt über Terminal-Kommandos und zugehörigen Optionen, Anpassung an die menschlichen Bedürfnisse und Denkweise, Reduzierung des notwendigen Trainings für die Anwender.

2. **User Interfaces können für verschiedene Aufgaben ausgelegt sein. Erklären Sie in diesem Sinne, was es für User Interfaces bedeutet, schnell erlernbar bzw. zweckerfüllend zu sein. Nennen Sie auch jeweils ein Beispiel.**

   - **Schnelle Erlernbarkeit:** Das UI führt den Benutzer Schritt für Schritt durch das System, die Oberfläche ist einfach und selbsterklärend. Ein Benutzer soll mit den System umgehen können, ohne jemals zuvor damit gearbeitet zu haben.(Beispiel: Touristeninformations-Terminal). Ein **zweckerfüllendes User Interface** verzichtet auf Hilfestellungen und ermöglicht es dem Anwender die Daten so einfach, schnell, effektiv wie möglich in das System einzugeben. (Beispiel: Datenerfassung für eine Datenbank).

3. **Was sind die Eigenschaften, welche ein gut implementiertes User Interface beschreiben?**

   - Leicht erlernbar, intuitiv, einfach, konsistent und ästhetisch ansprechend.

4. **Eine Entwicklungsumgebung ist eigentlich eine Sammlung von mehreren Tools, welche dem Programmieren bei seiner Arbeit unterstützen. Welche Tools sind in diesen IDEs im Allgemeinen zu finden und wozu dienen sie?**

   - **Tools zum User Interface Design**: Ermöglichen es dem Programmierer ohne Codetipperei UIs zu implementieren und zu testen; **Intelligente Editoren**: Unterstützen beim Schreiben des Codes durch Textformatierung, Kontext-Sensitive Hilfe und automatische Codevervollständigung; **Compiler/Interpreter**:

Kontrollieren und übersetzen des Source Code und Programmausführung; **Debugger**: Dienen zur Suche von logischen Programmfehlern, Finden von Bugs; **Projektverwaltung**: Übersichtliches Verwalten des gesamten Projekts (Source, Bibliotheken, Ressourcen)

5. **Erklären Sie die wesentlichen Aufgaben eines intelligenten Editors einer IDEs. Was muss er unterstützen und wie kann er dem Programmierer helfen?**

   - Syntaxhighlighting (hervorheben von Schlüsselwörtern, Methoden, ...), Sourcecode-Formatierung und Strukturierung (automatisches Zeileneinrücken, einfügen von Leerzeichen und Tabulatoren, einheitliche Formatierung nach allgemeinen Code Conventions), Programmrümpfe einfügen (Codeteile, welche fast immer gleich sind, automatisch einfügen, um dem Programmierer Arbeit zu ersparen), Codevervollständigung und Hilfe (Wenn Methodennamen eingegeben werden, zeigt der Editor die Parameter/-typen und Vorschläge an)

6. **Ein Compiler übersetzt den Sourcecode in ein lauffähiges Programm. Wie läuft das Compilieren (speziell in einer Entwicklungsumgebung) ab?**

   - Der Präprozessor wertet Anweisungen an den Compiler aus (Optionen, etc) und entfernt diese anschließend aus dem Sourcecode. Anschließend übersetzt der Compiler den Sourcecode in eine Zielsprache (bei IDEs eigentlich immer Maschinencode). Der Linker bindet zusätzliche (bereits im Maschinencode vorliegende) Bibliotheken ein und fügt alle Dateien zu einem großen Ganzen zusammen. Zum Schluss setzt der Loader die absoluten Adressen der Variablen ein.

7. **Es werden, abhängig von der jeweiligen Programmiersprache, Compiler bzw Interpreter zur Programmerstellung/-ausführung verwendet. Nennen Sie die wichtigsten Unterschiede.**

   - Interpreter übersetzen ein Programm jedesmal, wenn es aufgerufen wird. Die Übersetzung wird nicht auf Dauer gespeichert. Compiler übersetzen den Quellcode eines Programms als Ganzes und speichern die Übersetzung in einer Datei. Compilierte Programme laufen schneller ab als interpretierte, da das übersetzen bei jedem Programmstart wegfällt. Es gibt auch Mischformen von Interpretern und Compilern.

8. **Nennen Sie die Aufgaben des Compilers in der Analysephase.**

   - Erkennen von Sourcecodebestandteilen wie Literalen, Schlüsselwörtern, Bezeichner und Operatoren. Syntaxprüfung, Semantikprüfung und Typumwandlungen.Wahl der richtigen überladenen Funktionen. Erzeugt einen abstrakten Zwischencode für die folgende Synthesephase.

9. **Nennen Sie die Aufgaben des Compilers in der Synthesephase.**

   - Optimieren des in der Synthesephase erzeugten Zwischencodes nach Größe/ Geschwindigkeit. Erzeugung des Zielcodes, zuerst Assembler und anschließend Maschindencode.

10. **Entwicklungsumgebungen für interpretierende Sprachen (z.B. Java) enthalten einen Interpreter für die Sourcecodeübersetzung/Programmausführung. Wie läuft das Interpretieren in einer IDE ab und wie nennt man diese spezielle Art von compilieren?**

    - Zuerst wird der Quellcode in eine interpretierbare Zwischensprache (abstrakter Maschinencode) übersetzt. Dieser dient als Quelle für den Interpreter, welcher dann für jede Codezeile ausführbaren Maschinencode erzeugt. Dieser Prozess wird als **Just-in-time** Compilierung bezeichnet. Der erzeugte Code wird nur in den Speicher geschrieben, muss also bei jedem Programmaufruf erneut compiliert werden.

11. **Erklären sie kurz den Begriff Translator. Ist ein Compiler ein Translator?**

    - Ein Translator nimmt ein Quellprogramm als Eingabe und wandelt es in Maschinencode um. War der Quellcode in einer Hochsprache geschrieben, wird der Translator Compiler genannt.

12. **Welche zwei Arten von Programm-Fehlern gibt es und was sind die Hilfsmittel in einer IDE, um diese zu finden?**

    - Syntaktische Fehler: Fehler, die der Syntax einer Programmiersprache widersprechen. Diese Fehler findet der Compiler in der Analysephase.
      Logische Fehler: Fehler in der Denkweise, falsche Algorithmen zur Lösung von Problemen,... Solche Fehler entdeckt man im Allgemeinen nur daran, das ein Programm etwas falsches macht. Um die genaue Stelle des Fehlers zu finden, verwendet man einen Debugger.

13. **Nennen Sie einige Abarbeitungsmodi eines Debuggers und erklären Sie den Begriff Breakpoint.**

    - Single-Stepping Modus (Step-Into Modus), Step-Over Modus, Step-Out, Go-to-Breakpoint, Go-to-Cursor; **Breakpoints**: Breakpoints sind Punkte, an dem der Debugger die Programmausführung unterbricht. Es können Zeilen im Sourcecode als Breakpoints markiert sein aber es können auch logische Bedingungen für das Unterbrechen gesetzt werden.

Kapitel 7

Thema 10 - Konfigurationsmanagement

1. [**Q**] Was ist eine Konfiguration?
   [**A**] Eine Konfiguration ...

   - ... beschreibt die Menge aller Zustände eines Projektes.
   - ... ist eine bestimmte Art der Gestaltung.
   - ... die Anordnung von Komponenten, definiert durch deren Anzahl, Anordnung und Interaktionen.

2. [**Q**] Was ist Konfigurationsmanagement?
   [**A**] Konfigurationsmangement ist eine Disziplin zum Überwachen von Konfigurationen (gemäß *IEEE*).

3. [**Q**] Wie sieht ein Projektzyklus aus, und was beschreiben die einzelnen "Stationen"?
   [**A**]

   (a) [Änderungsmanagement]. Hier wird das Pflichtenheft geschrieben; bei Änderungsanforderungen werden diese mitgeteilt. Auch die zu erledigenden Aufgaben werden hier festgelegt.
   (b) [Projektmanagement]. Hier werden die Verantwortlichkeiten festgelegt (wer hat was und wann zu tun?).
   (c) [Versionsmanagement]. Hier wird das Projekt durch die Verantwortlichen bearbeitet.
   (d) [Erzeugungsmanagement]. Hier wird das Produkt erzeugt.
   (e) [Ausgabenmanagement]. Hier wird das Produkt an die Kunden gegeben.

4. [**Q**] Was versteht man unter der Entkopplung in Räumen?
   [**A**] Räume teilen das Projekt in einzelne Zustände oder Bestandteile auf. Jeder Raum hat gesonderte und wohldefinierte Zugriffsrechte. Das Benutzen eines Raumes, in dem man etwas ausführen kann (z.B. kompilieren), übergibt das Ausgangsprodukt an den nächsten Raum.

5. **[Q]** Wie funktioniert die Versionierung?
   [A] Die Versionierung ist ein Prinzip der eindeutigen Archivierung. Ein Versionsraum (cf *Räume*) beinhaltet alle Versionen eines Elementes, einer Einheit, eines Modules oder einer Ausgabe.

6. **[Q]** Was bedeutet bei einer Versionierung der Begriff *Variante*?
   [A] Wenn dieselbe Version eines Produkte auf verschiedene Plattformen und Umgebungsbedingungen angepasst werden muss, wird eine *Variante* von ihr angefertigt.

7. **[Q]** Was ist in ISO 9000 festgelegt, was im Gegensatz dazu im nächsten Punkt?
   [A] ISO 9000 beinhaltet die grundlegenden Termini von Konfigurationsmangement. ISO 9001 beinhaltet das Qualitätsmanagement.

8. **[Q]** Wie wird gemäß ISO 9000:2000 zertifiziert?
   [A] Der Zertifikatsprüfung, welche zu 70 % bestanden sein muss, geht ein Kurs voraus, der aus 6 Einheiten besteht:

   (a) Grundlagen.
   (b) Strukturen.
   (c) Formen und Verfolgbarkeit.
   (d) Projektzyklen.
   (e) Gesetz.
   (f) Implementierung und Prüfung.

9. **[Q]** Welche sind die 6 W-Fragen des österreichischen V-Modells der IT-Branche?
   [A]

   (a) [Wer] darf etwas tun/hat etwas getan; Verantwortlichkeiten?
   (b) Aus [was] bestehen die Einheiten, Komponenten und Subkomponenten?
   (c) [Wo] sind die Komponenten situiert (vgl. *Raumkonzep*)?
   (d) [Wann] wird oder wurde ein Softwareelement bearbeitet?
   (e) [Warum] wird oder wurde wurde ein Softwareelement bearbeitet?
   (f) [Wie] werden die Komponenten zusammengebaut?

10. **[Q]** Warum ist Konfigurationsmanagement bei Softwareentwicklern etwas unbeliebt?
    [A] Durch die Forderung, alles zu dokumentieren, sehen sich Softwareentwickler in der kreativen Freiheit eingeschränkt. Zumal auch jeder "Holzpfad" mitgeschrieben wird, würde genaue Protokollierung gerade dann Chaos verursachen. Außerdem fordert Konfigurationsmanagement, dass die Projekte rein funktioneller und nicht kreativer Natur sein sollen.

Kapitel 8

Thema 12 - Projektmanagement

1. [**Q**]In welche 3 Typen kann man Programmierer einteilen?
   [**A**]

   - Aufgabenorientiert - wird alleine durch die Arbeit motiviert
   - Zusammenarbeitsorientiert - Mitarbeiter leisten durch Anwesenheit und Aktionen die gesamte Motivation
   - Selbstorientiert - Drang nach persönlichem Erfolg

2. [**Q**] Wie viele Kommunikationsverbindungen gibt es in einer Gruppe mit n Mitgliedern?
   [**A**] $n/2 * (n - 1)$

3. [**Q**] Welche Möglichkeiten gibt es, die Kommunikation zu koordinieren? [**A**]

   - Einsatz eines zentralen Koordinators
   - Minimierung der Gruppengröße

4. [**Q**] Welche sind die bekanntesten Kalkulationsverfahren? [**A**]

   - CoCoMo - basiert auf der Anzahl der zu entwickelnden Codezeilen
   - Function Point - Basis: Kenngrößen wie Ein-/Ausgabeoperationen
   - Function Weight - basiert auf Kennzahlen aus früheren Projekten

5. [**Q**] Welche Formel kann man zur Berechnung der maximalen Teamgröße verwenden?
   [**A**] maxTeamgröße= $\sqrt{GeplanterAufwandInBearbeitermonaten}$

6. [**Q**] Was versteht man unter *Meilensteinen*?
   [**A**] Bei der Planung eines Projekts sollten sogenannte Meilensteine festgelegt werden, bei deren Erreichen ein formaler Bericht erstellt werden soll. Jeder Meilenstein stellt den Höhepunkt einer Phase dar. Ein guter Meilenstein ist z.B. die Fertigstellung des ersten Entwurfs

7. [**Q**] Wozu dienet das *Projekttagebuch*?
   [**A**] Das Projekttagebuch hält wichtige Entscheidungen und noch zu entscheidende Sachverhalte auf Managementebene fest.

8. [**Q**] Wie sind Chef-Programmierer-Teams aufgebaut? [**A**]

   - Erfahrener, hochqualifizierter Chef-Programmierer (übernimmt die gesamte Verantwortung)
   - Geschickter, erfahrener Zweitprogrammierer (verifiziert die Arbeit des Chef-Programmierers)
   - Bibliothekar (übernimmt Verwaltungsaufgaben)

9. [**Q**] Was versteht man unter *Standard*?
   [**A**] Ein Standard ist eine abstrakte Darstellung, die minimale Leistungs-, Robustheits und Organisationsgrade definiert, die das zu entwickelnde Projekt anpeilen soll

10. [**Q**] Was sind die Aufgaben eines Projekt-Managers? [**A**]

    - Den richtigen Mitarbeiter für die richtige Aufgabe finden
    - Eine angenehme Arbeitsumgebung schaffen
    - Freiräume geben
    - Fehler zulassen, aber auch
    - Erwartunen deutlich machen
    - Gerecht sein (gleiche Maßstäbe an die Leistungen und Ergebnisse der Mitarbeiter anlegen)
    - Streng sein (klare und eindeutige Haltung besonders gegenüber nicht zufriedenstellenden Leistungen)

11. [**Q**] Im Zuge der Grobplanung wird eine Aufwandsschätzung durchgeführt. Erklären sie die unter anderen dabei verwendete "Function-Point-Methode"! Nennen Sie hierauf das wichtigste Stichwort im Bezug auf Aufwandsabschätzung!
    [**A**] Bei dieser von IBM entwickelten Methode wird auf Basis bestimmter Kenngrössen der Aufwand abgeleitet. Dabei wird das Softwarevorhaben in seine Grundfunktionen zerlegt, beispielsweise Eingaben, Ausgaben, Abfragen an das System, zu speichernde Daten, Durchsatz, etc.. Danach wird für jede dieser Funktionen eine Komplexität, die von "leicht" über "mittel" bis "schwer" geht, festgelegt. Aus der Gesamtheit der Funktionskomlexitäten kann die Gesamtkomplexität abgeleitet werden.
    Stichwort: "Erfahrung"

12. [**Q**] Welche vier Punkte sollten bei der Meilensteinplanung beachtet werden?
    [**A**]

    - Die Meilensteine müssen überprüfbar sein.
    - Die Meilensteine sollten zeitlich nicht zu weit auseinander liegen.
    - Es ist auf eine zeitliche Gleichverteilung der Meilensteine zu achten.
    - Der Projektmanager hantiert bei der Meilensteinvergabe mit der kostbaren Ressource "Zeit"!

13. **[Q]** Erläutern Sie in ein paar Worten wozu eine Stellenbeschreibung dient und nennen Sie anschliessend jene Fragen, welche durch eine korrekte Stellenbeschreibung geklärt werden sollten!
    [A] Die Intention einer Stellenbeschreibung ist, eine klare, lückenlose und überschneidungsfreie Zuständigkeitsordnung der Mitarbeiter zu schaffen beziehungsweise ihre konkreten Aufgaben zu präzisieren.
    Dabei sollten folgende Fragen geklärt werden:
    - Von wann bis wann wird die Stelle benötigt?
    - Welche Qualifikation ist für die Stelle erforderlich?
    - Wieviel Erfahrung sollte die Person mitbringen?
    - Welche Aufgabe soll vom Stelleninhaber erledigt werden?
    - Welche Arbeitsmittel sollten dazu verwendet werden?
    - Wie ist die Stelle organisatorisch eingeordnet?
    - Wie ist die Frage der Stellvertretung bei Ausfall der stellenbesetzenden Person geregelt?
    - Wieviel Prozent einer Arbeitskraft werden benötigt?

14. **[Q]** Zeichnen Sie das "Wasserfallmodell mit Rücksprüngen" auf!
    [A]

15. **[Q]** Was geschieht in der "Analyse- und Definitionsphase" des Wasserfallmodells?
    [A] Es werden die spezifischen Produktanforderungen ermittelt. Auch eine Ist-Analyse ist in dieser Phase erforderlich. Auf die Analyse folgt eine Zieldefinition. Das heisst man definiert ausführlich die erwarteten Produktanforderungen. Folglich bieten sich in diesem Abschnitt auch das Machen einer Durchführbarkeitsstudie und das Erstellen einer Kosten-Nutzen-Analyse an.
    Als Phasenergebnis wird im Wesentlichen die vollständige Produktdefinition erwartet.

16. **[Q]** Was wissen Sie über Prototyping?
    [A] Beim Prototyping differenziert man zwischen zwei Arten. Erstens dem "Rapid Prototyping" bei dem nur eine Art Wegwerfprototyp erstellt wird, der zwar lauffähig aber nicht einsatzfähig ist. Er wird in einer frühen Projektphase erstellt um beispielsweise dem Auftraggeber einen Eindruck über das fertige Produkt zu verschaffen. Zum Zweiten gibt es noch den sogenannten "evolutionären Prototypen". In diesem Fall wird eine Serie von lauffähigen Prototypen entwickelt, die direkt in die Produktionsversion der Software einlaufen und nach und nach zum Fertigprodukt führen. Diese zweite Art des Prototypings verwendet man vor allem bei grossen Softwareprojekten, welche wegen ihres Umfangs Aufteilung erfordern.

17. **[Q]** Welche bewährten Konzepte werden im "Rational Unified Process" aufgegriffen?
    [A]

- Iterative Softwareentwicklung
- Management der Anforderungen
- Verwendung komponentenbasierender Architekturen
- Visuelle Modellierung
- Qualitätssicherung
- Kontrolle bei Änderungen

18. [**Q**] In einem Softwareunternehmen gibt es die Abteilung "Methoden und Verfahren". Was macht diese Abteilung und mit welchen Problemen ist sie konfrontiert?
[**A**] Die Abteilung "Methoden und Verfahren" wird eingesetzt um im Unternehmen eine Standardisierung der eingesetzten Methoden und Verfahren zu gewährleisten. Dazu gehört das Auswählen von Methoden und Verfahren und deren Vermittlung an die Mitarbeiter. Weiters zählt die Erstellung eines Projekthandbuchs zu ihrem Aufgabenbereich. Nebenbei unterstützt diese Abteilung die Entwicklungsteams, sollte es zu Unklarheiten im Bezug auf Methoden und Verfahren kommen.
Zu kämpfen hat diese Abteilung vor allem damit, dass sie oft nur eine Stabstelle darstellt und daher keine direkte Weisungsbefugniss besitzt, was bedeutet, dass sie nur beschränkten Einfluss auf die Projektteams hat.

19. [**Q**] Erklären sie die Einfluss-Projektorganisation!
[**A**] Bei der Einfluss-Projektorganisation hat der Projektleiter nur begrenzten Einfluss auf das Projekt, denn die fachliche und disziplinarische Verantwortung liegt bei den Leitern der jeweiligen Fachabteilungen. Der Projektleiter übernimmt hier Planungsaufgaben und ist lediglich für die Entscheidungsvorbereitung zuständig (d.h er ist eine reine Stabsstelle).
Der Vorteil der Einfluss-Projektorganisation ist, dass gegebenenfalls die Mitarbeiter für mehrere Projekte eingesetzt werden können, da sie nicht einem Projekt zugeordnet sind, was oft eine bessere Auslastung und weniger Leerphasen des Einzelnen bedeutet.

20. [**Q**] Ein Projektmanager hat neben fachlichen Fragen auch mit soziologischen Aufgaben zu tun, darunter mit "Kommunikation". Erklären Sie diesen Aspekt!
[**A**] Da sich der Erfolg eines Projekts durch das Erreichen der Ziele definiert, muss ein Projektmanager seinen Kollegen diese auch mitteilen, da bei vielen kleinen Teilaufgaben die Mitarbeiter ansonsten das Gesamtziel aus den Augen verlieren könnten. Es liegt in der Natur des Menschen, dass er etwas lieber macht, wenn er weiss für was es gut ist. Es motiviert nach einem Leitbild zu arbeiten. Fernerhin ist wichtig, dass der Projektmanager seinen Mitarbeitern seine persönlichen Erwartungen deutlich machen und deren Erfüllung in regelmässigen Abständen kontrollieren soll. Dabei dürfen Gerechtigkeit, aber auch Strenge nicht fehlen. Trotzdem gehört dazu, dass man seinen Mitarbeitern Fehler verzeiht und Toleranz übt.
Jedoch auch die Kommunikation zwischen den einzelnen Teammittgliedern ist nicht zu vergessen, denn sie sollten ebenfalls ausreichend miteinander in Kontakt treten, miteinander Probleme lösen und aktiv zusammenarbeiten. Ausgiebiger Informationsaustausch hilft auf jeden Fall vermeidbare Missverständnisse auszuschliessen.

Kapitel 9

Thema 14 - Testen von Software

1. [**Q**] Wozu dienen Akzeptanztests?

   [**A**] Um den Fortschritt des Projekts aus Sicht des Kunden zu messen.Diese Tests werden vom Kunden spezifiziert und verifiziert. Akzeptanztests haben besonders unter XP wesentliche Bedeutung, da sich hier das Design der Software grossteils erst aus der Programmierung ergibt und daher sehr schnell von den Wünschen des Auftraggebers abweichen kann.

2. [**Q**] Was ist bei Black-Box-Tests (im Gegensatz zu Strukturtests) nicht bekannt?

   [**A**] Der Source-code. (Daher muss der Tester aufgrund seiner Erfahrung und Intuition versuchen, bei der Ausfuehrung des Programms Eingaben zu finden, die zu Fehlern fuehren.)

3. [**Q**] Aequivalenzklassen: angenommen die Eingabe positiver Zahlen und negativer Zahlen soll zu unterschiedlichen Ausgaben fuehren, 0 sei eine ungueltige Eingabe. Welche Eingaben wuerden sie testen?

   [**A**] Die 3 Aequivalenzklassen sind hier positive Zahlen, negative Zahlen und ungueltige Eingaben. Getestet werden sowohl Randwerte als auch gewoehnliche Werte, also etwa 0 (Klasse 0) 1,(Rand pos.), -1 (Rand neg.), und zb. -1123, 214

4. [**Q**] Nennen sie mind. eine andere Bezeichnung fr Strukturtests

   [**A**] White-Box-Tests, Glass-Box-Tests. (Diese Bezeichnungen druecken den Gegensatz zu Black-Box-Tests deutlich aus. Hier ist also der Sourcecode bekannt.)

5. [**Q**] Was ist das Ziel von Pfadberdeckungstests?

[**A**] Jede Anweisung des Programms soll mindestens einmal durchlaufen werden. (Zu diesem Zweck werden meistens Flussdiagramme verwendet, die die moeglichen Wege durch ein Programm visualisieren.)

6. [**Q**] Interaktion zwischen Objekten: Was muss bei so genannten "collection classes" getestet werden?

   [**A**] Es handelt sich hierbei um Interaktionen, die daraus bestehen, dass Eine Klasse eine Sammlung von Objekten einer anderen Klasse ist. Es muss daher nur das Hinzufgen, bzw. Entfernen von Objekten getestet werden.

7. [**Q**] Hierarchien: Was muss fr systematisches Testen von Subklassen gewaehrleistet sein?

   [**A**] Einhaltung des Ersetzungsprinzips, bzw wohlgeformte Hierarchien. (das Ersetzungsprinzip besagt, dass ein Objekt einer Unterklasse immer ein Objekt der Oberklasse ersetzen koennen muss. Wird dieses nicht eingehalten kann es leicht zu unberechenbaren und damit schwer testbaren Vererbungsstrukturen kommen.)

8. [**Q**] Welche 2 Arten von Tests sind bei XP (eXtreme Programming) von wesentlicher Bedeutung?

   [**A**] Unit Tests (Programmeinheiten werden getestet, begonnen mit der ersten geschriebenen Methode bis zu Komponenten aus mehreren Klassen. Muessen immer fehlerfrei laufen, da darauf aufgebaut wird.) und Akzeptanztests (siehe Frage oben)

9. [**Q**] Nennen sie zwei Vorteile des Test-first-Prinzips

   [**A**] Einfacherer Code, bessere Testbarkeit, Test dokumentiert Code. (Ersteres ergibt sich aus dem Prinzip immer den einfachstmoeglichen Code zu schreiben, der den Test erfuellt.)

10. [**Q**] Welche Programmiersprache(n) muss jemand beherrschen, der Software mit Hilfe von "JUnit" testen mchte?

    [**A**] Java. Aus 2 Gruenden: erstens wird damit Java-Code getestet, zweitens wird der Test selbst in Java geschrieben. (Es gibt aber aequivalente Testtools auch fuer C++ und etliche andere Sprachen.)

Kapitel 10

Thema 15 - Personalmanagement

1. [**Q**]Welche zwei primären Faktoren erschweren das Management von Softwareprojekten im Vergleich zu anderen Projekten (wie z.B. Bauvorhaben)?
   [**A**]Zwei primäre Ursachen:

   - Die Hauptursache die das Management von Softwareprojekten erschwert, ist die Tatsache, dass Software ein immaterielles Produkt ist, dessen Fertigstellungsgrad und dessen Qualität nur schwer wahrgenommen werden kann.
   - Ein weiterer erschwerender Faktor liegt im sehr **dynamischen Umfeld seitens der Technologie und der Anforderungen** sowie der fehlenden Normierung der Komponenten.

2. [**Q**]Welche wichtigen Fragen sollte Auwandsschätzung beantworten?
   [**A**]

   - Zum einen *Welchen Aufwand verursacht das Projekt?* Hierbei sollte geklärt werden was genau unter einem Personenmonat
     (gebräuchliche Maßeinheit) zu verstehen ist bzgl. Verrechnungssatzes, Urlaubsgeldern, Krankheitszeiten etc..
   - Zum anderen *Welche Dauer ist für das Projekt zu erwarten?* Die einfache Rechnung, Anzahl der Mitarbeiter durch die geschätzte Dauer des Projektes geht hier allerdings nicht auf. Dafür gibt es spezielle Verfahren die es ermöglichen zu jeden Zeitpunkt des Projektes die optimale Mitarbeiterzahl zu finden.

3. [**Q**]Was versteht man unter Meilensteinen bzw. Lieferschritten in einem Projekt?
   [**A**]Die Definition von Meilensteinen dient der Projektüberwachung. Wird im Projektablauf ein Meilenstein erreicht, so bedeutet dies, dass bestimmte Ergebnisse in einer definierten Qualität vorliegen. Nach erreichen eines jeden Meilensteines sollte eine formelle Erklärung, zB in Form eines Berichts, abgegeben werden, welcher, wiederum der Projektleitung vorgelegt werden kann. Ein Lieferschritt ist ein Projektresultat, das an den Kunden ausgeliefert wird.

4. [**Q**]Auf was sollte man beim Aufstellen eines Projektzeitplans achten?
   [**A**]Man schätzt dabei die benötigte Zeit und die benötigten Ressourcen zum Ab-

schließen der Aufgaben ab und bringt diese in eine zusammenhängende Abfolge. Besonders wichtig ist es, beim Auftreten neuer, besserer Informationen den Zeitplan anzugleichen. Die erstellten Zeitpläne shließt die Aufteilung des Gesamtprojektes in verschiedene Aufgaben ein, ebenso die Beurteilung der benötigten Zeit zur Fertigstellung dieser Aufgaben. Bei möglicherweise parallel auführbaren Aufgaben muss darauf geachtet werden, dass diese koordiniert ablaufen um unnötige Verzögerungen vorzubeugen.

5. [**Q**]Nenne die Komponenten des Risikomanagements
   [**A**]

   - Risikoidentifizierung
     Aufdecken von potentiellen Risiken für das Projekt.
   - Risikobewertung
     Ermitteln des möglichen Schadens und der Eintrittswahrscheinlichkeit der einzelnen Risiken.
   - Risikovermeidung/Risikoverminderung Finden von Gegenmaßnahmen um potentiellen Risiken aus dem Weg zu gehen, zB durch Versicherung.

6. [**Q**]Was ist die Bedürfnisshierarchie von Maslow?
   [**A**]Die Hierarchie besteht aus 5 Stufen:

   - Bedürfnis nach Selbstverwirklichung
   - Ich-Bedürfnisse (Wertschätzung, Prestige, ...)
   - Soziale Bedürfnisse (Familie, Gruppenzugehörigkeit, ...)
   - Sicherheitsbedürfnisse (Rechtsgehalt, Versicherungen, ...)
   - Physiologische Bedürfnisse (Essen, Trinken, Wohnung, ...)

   Die Hierarchie der Bedürfnisse bringt dabei zum Ausdruck, dass die Bedürfnisse der jeweils höheren Ebene erst dann zum Tragen kommen, wenn die Bedürfnisse der darunter liegenden Ebenen hinreichend befriedigt sind. Man geht davon aus, dass der einzelne Mitarbeiter nur dann sein volles Leistungspotential einbringt, wenn er alle in der obigen Hierarchie dargestellten Bedürfnisse im Rahmen seiner Tätigkeit befriedigen kann.

7. [**Q**]Welche Vorteile hat Gruppenarbeit (mindestens 5)?
   [**A**]

   - die Gruppe vermag Leistungen zu erbringen, die ein einzelner Arbeiter nicht imstande wäre zu verrichten.
   - besseres Urteilsvermögen durch die Berücksichtigung unterschiedlicher Perspektiven

- verbesserte Möglichkeiten der Informationsübermittlung und gesicherten Informationsfluss
- größere Kontaktintensität
- es kommen mehrere Arten von Geschicklichkeit und verschiedenes Sachwissen zusammen (Know How Vorteil)
- eine Gruppe kann eine größere Menge von Informationen verarbeiten
- größere Zieleinhaltungskontrolle
- bessere Lernfähigkeit
- bessere Ausnutzung von technischen Installationen
- Anreicherung der Phantasie und Kreativität
- Risiko wird geteilt
- Emotionale Faktoren der Arbeitsgruppe wie: Kontakt, Geborgenheit, Sicherheit, Solidarität, Identität usw. mit der möglichen Folge einer erhöhten Motivation.

8. [**Q**]Welchen Zweck hat ein Vorgehensmodell im Softwareprojekt?
[**A**]Die Grundidee der Vorgehensmodelle verfolgt das Ziel einer hierarchischen Gliederung der Gesamtaufgabe, die ein- oder mehrstufig sein kann. Die Darstellung dieser Gliederung kann sowohl zerlegungsorientiert - als Baum oder Hierarchie - als auch ablauforientiert - als Graph - erfolgen. Man möchte dadurch einen genauen Überblick über das Gesamtprojekt gewinnen und Plan- und Kontrollierbarkeit fördern. Letztlich dienen Vorgehensmodelle damit dem Zweck, Projekte einem sinnvollen Management zugänglich zu machen.

9. [**Q**]Welche zwei Dimensionen werden im RUP Rational Unified Process betrachtet?
[**A**]Das eigentliche Vorgehensmodell betrachtet den Ablauf des Projektes in zwei Dimensionen. Zum einen ist da natürlich die zeitliche Dimension, die in vier Phasen *Inception phase, Elaboration phase, Construction phase, Transition phase* unterteilt wird. Neben diesem dynamischen Ablauf werden in der zweiten Dimension die statischen Aspekte des Projektes betrachtet. Hierzu zählen die Aktivitäten, die erstellten Dokumente, die am Projekt beteiligten Personen und die statischen Arbeitsabläufe.

10. [**Q**]Es gibt 2 wichtige Aspekte der Planung von Softwareprojekten. Welche sind dies?
[**A**]

    (a) Erstellen von Plänen ist insbesondere bei Softwareprojekten ein kreativer Prozess. Es ist zwar sinnvoll, diesen Prozess durch die Vorgabe eines Vorgehensmodell und durch andere Planungshilfen zu strukturieren und zu leiten. Allgemein kann man lediglich festhalten, dass bei der Anwendung von Planungsvorgaben stets auf den Sinn der Vorgaben geachtet werden sollte.

    (b) Planung muss rollierend erfolgen. Das bedeutet, dass eine einmal erstellte Planung immer wieder verfeinert und hinterfragt werden muss.

Kapitel 11

Thema 16 - Aufwandsschätzungen

1. [**Q**] Was ist Aufwandsschätzung im Allgemeinen
   [**A**] Aufwandsschätzung ist der Versuch, den für ein vorgegebenes Ergebnis erforderlichen Aufwand zu bestimmen.

2. [**Q**] Warum schätzt man den Aufwand eines Projektes
   [**A**] Um sich eine Vorstellung über den für ein Projekt erforderlichen Aufwand zu verschaffen.

3. [**Q**] Wie wirkt sich eine bei der Systementwicklung beachtete Möglichkeit zur Wiederverwendbarkeit des Systems in Bezug auf die Aufwandsschätzung aus
   [**A**] Sie steigert selbstverständlich den Aufwand. Erst spätere Projekte, die dieses System wiederverwenden, profitieren davon.

4. [**Q**] Was ist der Unterschied zwischen einer Methode zur Aufwandsschätzung und einem Verfahren zur Aufwandsschätzung
   [**A**] Als Methode wird ein Prinzip zum Vorgehen bei einer Aufwandsschätzung bezeichnet, Bsp. Analogiemethode. Ein Verfahren kann bereits zur Aufwandsschätzung benutzt werden - man führt es aus. Bsp. Function-Point-Verfahren.

5. [**Q**] Welche Methoden zur Aufwandsschätzung kennen Sie
   [**A**] Analogiemethode, Relationsmethode, Multiplikatormethode, Gewichtungsmethode, usw.

6. [**Q**] Wie funktioniert die Methode der parametrischen Schätzgleichungen
   [**A**] Mittels Korrelationsanalysen werden Einflussfaktoren gesucht, deren wertmässige Ausprägung in einem engen Zusammenhang mit dem angefallenen Aufwand steht. Dazu sind viele Untersuchungen nötig, um ein möglichst genaues Bild des zu erwartenden Aufwandes zu bekommen.

7. [**Q**] Was ist INVAS
   [**A**] INVAS bedeutet Integriertes Verfahren zur Aufwandsschätzung und ist ein neues Schätzverfahren, das auf der Analogiemethode basiert und zum Vergleich die Mustererkennungstheorie verwendet.

8. [**Q**] Was ist ein Function-Point
   [**A**] Function-Points werden beim Function-Point-Verfahren verwendet und entsprechen jeweils einer elementaren Endbenutzerfunktion.

9. [**Q**] Nennen Sie einige Faktoren der Allgemeinen Anwendungscharakteristik
[**A**] Kommunikation, verteilter Einsatz, Änderbarkeit, Online-Abfragen, Performance, usw.

10. [**Q**] Nennen Sie eine wichtige Voraussetzung für die Anwendung von Verfahren zur Aufwandsschätzung
[**A**] Der Schätzende sollte keine Detailkenntnisse des Schätzverfahrens besitzen.

11. [**Q**] Geben Sie 3 Gründe an, warum Aufwandsschätzungen gemacht werden
[**A**] Für die Einteilung von Geld und Ressourcen, für zeitliche Steuerung des Projekts, Beurteilung des Aufwands für technische Änderungen

12. [**Q**] Nennen Sie 3 Schätzmethoden
[**A**] Analogiemethode, Relationsmethode, Gewichtungsmethode

13. [**Q**] Wie ist das grundsätzliche Vorgehen bei den meisten Schätzmethoden
[**A**] Vergangene Projekte werden analysiert und aus diesen Daten wird der Aufwand für das neue Projekt geschätzt

14. [**Q**] Wie funktioniert die Analogiemethode
[**A**] Es wird ein vergangenes ähnliches Projekt gesucht. Abweichungen zum neuer Projekt werden von Experten mit viel Erfahrung ausgeglichen. Dann wird direkt auf den Gesamtaufwand des neuen Projekts geschlossen.

15. [**Q**] Nennen Sie zwei bekannte Schätzverfahren
[**A**] Function-Point-Verfahren, COCOMO-Verfahren

16. [**Q**] Was sind Function Points
[**A**] Function-Points sind Punkte (eigentlich eine Zahl), die aus allen Funktionen aus dem zu schätzenden Projekt berechnet werden. Diese Funktionen werden auch gewichtet. Durch die Anzahl der Function-Points lässt sich dann auf den Projektaufwand schließen.

17. [**Q**] Wie erfolgt beim Function-Point-Verfahren nach der Ermittlung der Function-Points die Ermittlung des Projektaufwands
[**A**] Die Anzahl der Function-Points und der dazugehörige Aufwand vergangener Projekte werden in einer Tabelle oder Graphik eingetragen. Durch Eintragen der neu ermittelten Function-Points lsst sich dann der neue Projektaufwand ablesen oder durch Interpolation ermitteln.

18. [**Q**] Warum werden die Funktionen beim Function-Point-Verfahren gewichtet
[**A**] Um zwischen dem Aufwand von verschiedenen z.B. Input Funktionen zu unterscheiden, können diese unterschiedlich gewichtet werden. Leicht, Mittel oder Schwer.

19. [**Q**] Was sind UCP
[**A**] Use Case Points

20. [**Q**] Warum ist es wichtig, das Schätzverfahren auch während des Projekts mehrfach zu wiederholen
[**A**] Durch zusätzliche Anforderungen während des Projekts, durch neu erkannte Probleme oder ähnliches kann sich der Projektaufwand verändern.

21. **[Q]** Aus welchen Komponenten setzt sich der Gesamtpreis eines Softwarepakets zusammen?
    [A] Aus den

    - Personalkosten, (worunter neben den Gehältern für Programmierer auch die für unterstützendes Personal, die Mieten, Lohnnebenkosten sowie die Kosten für andere Einrichtungen fallen)
    - Soft- und Hardwarekosten plus Wartung
    - Schulungs- und Reisekosten für Mitarbeiter

22. **[Q]** Nennen Sie mindestens vier preiserhöhende oder -senkende Gründe bei der Festlegung eines Softwarepreises!
    [A] Preiserhöhend:

    - Unsicherheiten bei der Aufwandsschätzung
    - Monopolstellung des Herstellers

    Preissenkend:

    - Erzieltes Eindringen des Unternehmens in ein neues Softwaregebiet - mit Aussicht auf einen späteren Gewinn.
    - Quellcode bleibt im Besitz des Herstellers
    - große Wahrscheinlichkeit auf Anforderungsänderungen. Preise für Anforderungsänderungen steigen entsprechend.
    - Firma ist finanziell angeschlagen; Billige Angebote scheinen oft besser als der Konkurs.

23. **[Q]** Welche Faktoren beeinflussen die "Produktivität" eines Unternehmens?
    [A]

    - Kenntnisse der Entwickler im jeweiligen Anwendungsgebiet
    - Qualität der Software wird nicht in der Produktivität gemessen. Qualitativ hochwertige Software ist demnach "unproduktiv".
    - Größere Teams brauchen mehr Zeit für die Kommunikation.
    - Entwicklungsumgebungen in einem Unternehmen
    - Arbeitsumgebung

24. **[Q]** Was sind die SLOC?
    [A] Source Lines Of Code. Maßeinheit für die Messung der Codegröße. Es existieren verschiedene Zählweisen dafür:

    - nur die ausführbaren Anweisungen
    - ausführbare Anweisungen plus Deklarationen
    - alle Programmzeilen außer Leerzeilen

25. **[Q]** Welche Faktoren werden bei der Zählung von *Object Points* in Betracht gezogen und wieviele Punkte werden jeweils dafür vergeben?
[A]

   - Bilschirmmasken: 1, 2 oder 3 Punkte
   - Berichte: 2, 5 oder 8 Punkte
   - 3 GL-Module: 10 Punkte

26. **[Q]** Beschreiben Sie das Vorgehen bei der Analogiemethode!
[A] Zuerst Faktoren suchen, die das Projekt beeinflussen könnten. Danach mit Projekten vergleichen, die unter ähnlichen Bedingungen realisiert worden sind. Abschließend wird das Projekt noch *subjektive* an an die neuen Verhältnisse angepasst. Siehe auch Kapitel 3.1

27. **[Q]** Wie könnte man die Prozentsatzmethode an folgendem Beispiel praktisch anwenden? Welche Daten werden benötigt, um eine vernünftige Schätzung durchführen zu können?

Phase	Durschnittlicher Aufwand einer Firma
Analyse	20 Stuff-Months
Design	19 Stuff Months
Implementierung	15 Stuff-Months
Test	26 Stuff-Months

[A] Es gibt zwei praktizierte Möglichkeiten:

   (a) Bei einem Projekt wurde bereits die erste Phase beendet. Dann kann aufgrund der Tabelle abgeschätzt werden, dass das Projekt noch etwa 3 mal den bisher benötigten Aufwand brauchen wird.

   (b) Es wird mit möglichst großer Genauigkeit die Analysephase abgeschätzt. Der Gesamtaufwand ist das Vierfache des Ergebnisses.

28. **[Q]** Worin besteht der Unterschied zwischen den Stuff-Months (dem Aufwand) und der Time of Develop (Entwicklungszeit)?
[A] Grundsätzlich ermittelt man zunächst die Stuff-Months nach der Formel $SM = a * (KDSI)^b$. a und b sind Parameter, die von der Komplexität des Projektes abhängen. Dann kann man mit einer Formel $TDEV = c * (SM)^d$ daraus die Entwicklungszeit errechnen. c und d sind wieder anzupassende Parameter.

29. **[Q]** Worin unterscheidet sich COCOMO II von Basic COCOMO?
[A]

   (a) Durch einen Skalierungsfaktor, der exponentiellen Einfluss auf das Ergebnis ausübt und sich aus 5 einzelnen Faktoren zusammensetzt.

   (b) Durch einen Aufwandsmultiplikator, der lineare Auswirkungen auf das Ergebnis hat und aus 17 Faktoren besteht.

   Der Aufwandsmultiplikator wirkt sich auf die Stuff-Months aus, der Skalierungsfaktor wirkt sich sowohl auf die Stuff-Months als auch auf die Time of Develop aus.

30. **[Q]** Welche Skalierungsfaktoren gibt es und wie muss man diese miteinander verknüpfen, um zu einem Skalierungsfaktor zu gelangen?
    [A]

    - Vertrautheit der Organisation mit dem Produktbereich
    - Freiräume bei Produktanforderungen und Vorgehensweisen
    - Ausgereifter, risikofreier Produktentwurf
    - Zusammenhalt zwischen den Projektbeteiligten
    - Prozessreife

    $SF = 0,01 * \sum SF_{i(i=1..5)}$

31. **[Q]** Wie wird der Aufwandsmultiplikator aus den einzelnen Komponenten berechnet und in welchem Wertebereich liegt das Ergebniss für den Aufwandsmultiplikator in COCOMO II?
    [A] Die 17 Kostentreiber befinden sich in Kategorien Produkt (5), Plattform (3), Personal (6) und Projekt (3). Nach dem Zuweisen eines Wertes zu jedem Kostentreiber werden die Werte einfach multipliziert ($AM = \prod AM_{i(i=1..17)}$). Das Ergebnis liegt zwischen 0,05 und 115.

32. **[Q]** Wozu wird Aufwandschätzung für ein Softwareprojekt benötigt?
    [A] Es ist wichtig zu wissen, wie viel Aufwand und Zeit für die Erledigung einer Aufgabe erforderlich ist und wie hoch die anfallenden Gesamtkosten sind. Die Schätzungen sollten während der Bearbeitung eines Projektes regelmäßig aktualisiert werden, um den Planungsprozess zu unterstützen und den effektiven Einsatz von Ressourcen zu ermöglichen.

33. **[Q]** Es gilt *Softwarekosten=errechnete Kosten+erhoffter Gewinn*. Welche politische oder wirtschaftliche Faktoren bewegen eine Organisation dazu, den Preis der SW über oder unter dem berechneten Preis anzusetzen?
    [A]

    - Marktchancen: wenn man in ein Marktsegment eindringen möchte, setzt man den Preis niedriger an.
    - Unsicherheit der Aufwandsschätzung: Berücksichtigung möglicher unvorhersehbare Ausgaben, der Preis wird höher angesetzt.
    - Vertragsbedingungen: Preis wird niedriger angesetzt, wenn dem Auftragnehmer das Recht zur Wiederverwendung des Codes gewährt wird.
    - Unbeständigkeit: Preis niedrig ansetzen, um Auftrag zu erhalten; bei geänderten Anforderungen kann später mehr verlangt werden.
    - Wirtschaftliche Bonität: Preis senken, um überhaubt einen Auftrag zu erhalten und auf bessere Zeiten warten.

34. **[Q]** Welche Nachteile birgt die Verwendung der Source Lines Of Code (SLOC) als Maßfür die Produktivität?

[**A**] Die Anzahl der gelieferten Codezeilen wird durch die Gesamtanzahl der Monate geteilt, die für die Fertigstellung des Projektes benötigt werden. Der Zeitraum beinhaltet auch Zeit, die für Analyse, Entwurf, die Programmierung, das Testen und die Dokumentation erforderlich ist.
Höhere Programmiersprachen enthalten auch Deklarationen, ausführbare Anweisungen, Kommentare, Makroanweisungen die mehrere Codezeilen erzeugen usw. Je anschaulicher eine Programmiersprache, desto geringer die sichtbare Produktivität.
Keine weit verbreiteten Standards, daher sind Vergleiche zwischen verschiedenen Organisationen schwierig.

35. [**Q**] Was sind die *Unadjusted Funktion Points* (UFP)? Erkläre in groben Zügen wie sie berechnet werden.
[**A**] Es gilt allgemein: UFP=$\sum$(Anzahl der Elemente einer bestimmten Programmfunktion)$\times$(deren Gewicht)
Die Prozedur zum Zählen der Function Points beinhaltet vier Schritte: Bestimme die Anzahl der Funktionstypen, bestimme den Grad der Komplexität jeder gezählten Funktion, weise den Komplexitätswerten bezüglich der Art der Funktion die richtigen Werte zu und zähle zum Schluss alle Werte zusammen.

36. [**Q**] Nenne drei Faktoren, auf denen der *Value Adjustment Factor* (VAF) basiert.
[**A**] Der VAF basiert auf 14 generellen Systemeigenschaften, welche die allgemeine Funktionalität der Anwendung berücksichtigen. Beispiele: Reusability, Performance(response time, Durchsatz...), Komplexe logische oder mathematische Berechnungen, Handhabung von verteilten Daten, Anzahl der Hilfsmittel zum Austausch von Information mit dem System usw.

37. [**Q**] In welcher Phase des COCOMO II Models werden Object Points eingesetzt und warum?
[**A**] Die Methode der Object Point Zählung wird in der ersten der drei Stufen des COCOMO II Models eingesetzt. Experimente haben gezeigt, das man mit Hilfe der Object Points in einem frühen Stadium der Schätzung etwas bessere Ergebnisse erzielt als mit Function Points. Diese Methode soll auch leichter zu Verwenden sein.

38. [**Q**] Erkläre kurz die Begriffe Data Element Type (DET), Record Element Types (RET) und File Type Referenced (FTR) im Zusammenhang mit Funktion Points.
[**A**] FTR: Ist ein Typ von Daten, der bei einer Transaktion referenziert wird; muss auch eine interne logische Datei sein oder zu einer externen Schnittstelle gehören.
DET: Ist ein einheitliches, vom User erkennbares, nicht Rekursives (sich nicht wiederholendes) Datenfeld. Die Daten sind dynamisch. Ein DET kann von einer Datei gelesen werden oder wird von einem DET aus einem FTR erzeugt.
RET: Ein RET ist eine Untergruppe von Datenelementen innerhalb einer internen logischen Datei (ILF) oder eines externen Interfaces. Um sie identifizieren zu können, ist es am besten auf die logische Gruppierung der Daten zu achten.

39. [**Q**] Welche zwei Arten von Maßen zur Definition von Produktivität gibt es und nenne Beispiele davon.
[**A**]

(a) *Größenspezifische Maße:* Beziehen sich auf die Größe der Software. Eine der gebräuchlichsten Einheiten sind die gelieferten Source Lines of Code. Andere Beispiele sind die Anzahl der gelieferten Objektcodeanweisungen oder die Seiten der Systemdokumentation.

(b) *Funktionsspezifische Maße:* Beziehen sich auf die Gesamtfunktionalität der gelieferten Software. Produktivität ist die Summe der nutzbringenden Funktionen, die innerhalb eines Zeitraums produziert werden. Beispiele sind Function Points und Object Points.

40. [**Q**] Welche Programmfunktionen müssen gemessen bzw. geschätzt werden, um die Gesamtanzahl der Function Points eines Programms zu berechnen?
[**A**] Externe Ein- und Ausgaben (User Daten oder Kontrolldaten, die in das System kommen oder weggehen), Benutzerinteraktionen, externe Schnittstellen (Daten die zwischen Systemen weitergereicht oder gemeinsam genuzt werden) und vom System verwendete Dateien.

41. [**Q**] Wie viele Stufen gibt es bei COCOMO II und erkläre sie kurz.
[**A**] Es gibt drei Stufen:
Die frühe Prototypenstufe: unterstützt die Aufwandsschätzung für Prototypen und Projekte, bei denen Software aus einer Zusammensetzung vorhandener Komponenten entwickelt wurde. Basiert auf Schätzung von gewichteten Object Points.
Die frühe Entwurfsstufe: Basiert auf Standardformel algorithmischer Modelle. Die Systemgröße wird in Kilo Source Lines of Code angegeben. Diese erhält man durch Schätzung der Anzahl der Function Points, die mit Hilfe von Standardtabellen (unter Beachtung verschiedener Programmiersprachen) in KSLOC umgewandelt werden.
Die Stufe nach dem Architektenentwurf: Basiert auf der selben algorithmischen Grundformel, die in der frühen Entwurfsstufe verwendet wird. Hier sollte die Größeneinschätzung genauer sein. Zwei wichtige Projektfaktoren finden Berücksichtigung: Unbeständigkeit der Anforderung, Umfang einer möglichen Wiederverwendung;

Kapitel 12

Thema 18 - Extreme Programming

1. [**Q**] Auf welche Werte baut Extreme Programming?
   [**A**] Einfachheit, Courage, Feedback, Kommunikation.

2. [**Q**] Welche Praktiken verwendet Extreme Programming?
   [**A**] Planing Game, Kleine Releases, Systemmetapher, Einfaches Design, Testen, Refaktorisierung, Gemeinsames Code-Eigentum, Andauerende Integration, 40-Stunden Woche, Vor-Ort Kunde, Programmieren in Paaren, Programmierrichtlinien.

3. [**Q**] Welche Aufgaben hat der Vor-Ort Kunde im XP Projekt?
   [**A**] Dieser muss Stories schreiben und beim Planning Game entscheiden, welche Stories in die nächste Release kommt. Außerdem sind vom Kunden Tests zu schreiben, die den alltäglichen Gebrauch der Software simulieren.

4. [**Q**] Welchen Lebenszyklus hat ein XP Projekt?
   [**A**] Erkundung, Plannung, Iterationen zum ersten Release, Produktion, Wartung, Tod.

5. [**Q**] Welche Aufgaben haben die Programmierer in einem XP Projekt?
   [**A**] Sie schätzen die Zeit die eine Story/Task braucht, um implementiert zu werden. Auerdem wählen Sie Tasks, welche Sie implementieren wollen. Schreiben Tests. Schreiben Code. Erklären dem Kunden die Konsequenzen seiner Wahl der Stories.

6. [**Q**] Welche Aufgaben hat ein Berater in einem XP Projekt?
   [**A**] Ein Berater übernimmt die Rolle eines Lehrers für Probleme, die das Team alleine nicht lösen kann. Er sollte deshalb nie alleine gelassen werden, sondern von Mitgliedern beobachtet und gefragt werden über das Problem.

7. [**Q**] Warum muss das Testen in einem XP Projekt automatisch ablaufen?
   [**A**] Dies ist notwendig damit auch wirklich getestet wird. Wenn sich der jeweilige Programmierer selbst hinsetzen muss, um das Programm zu testen, geschieht dies viel seltener, als wenn die Test per Knopfdruck ablaufen.

8. [**Q**] Wie funktioniert XP wenn der Preis feststeht?
   [**A**] Es entsteht ein Vertrag in dem Preis und Termin feststehen, aber die Funktion des Produkts nur grob feststehen. Die Funktionen können nicht von Beginn an genau feststehen, da dies die Flexibilität sehr stark einschränken würde.

9. **[Q]** Was macht XP flexibel gegenüber Anforderungen?
[A] Die kurzen Release- und Iterationsphasen, bei denen jeweils zu Beginn neue Teile hinzukommen oder wegkommen können, ermöglichen rasche Änderung der Richtung in die sich das Projekt voranbewegt.

10. **[Q]** Nennen Sie 5 Rollen in einem XP Projekt?
[A] Programmierer, Kunde, Tester, Trainer, Berater.

11. **[Q]** Nennen sie ein Beispiel für ein Projekt, das mit Extreme Programming nur ineffizient realisierbar ist.
[A] Z.B.: Projekte, die nur einer spezifischen Spezifikation genügen sollen, wie eine Syntaxprüfung für eine Programmiersprache; Projekte mit zu großem Personalaufwand; Projekte mit zu hohen Integrationszeiten.

12. **[Q]** Was ist die Mindestteamgröße bei Extreme Programming?
[A] 2.

13. **[Q]** Was kann man sich unter der XP-Programmiertechnik *Pair Programming* vorstellen?
[A] Es wird immer zu zweit an einem Computer programmiert: Während einer codet, macht der andere eine Art "Code Review" und kritisiert den Programmierenden; zum Beispiel wenn ihm eine bessere oder einfachere Löung für das Problem einfällt, das gerade behandelt wird.

14. **[Q]** Was ist Extreme Programming?
[A] Ein leichtgewichtiger Softwareentwicklungsprozess.

15. **[Q]** Was wird bei XP als die wichtigste Aktivität bei der Entwicklung von Software angesehen?
[A] Programmieren.

16. **[Q]** Nennen sie eine der Aufgaben des Kunden beim XP-Softwareentwicklungsprozess.
[A] Z.B.: Schreiben der Akzeptanztests, Mitwirken am Planning Game, als On-Site Customer vor Ort zur Verfügung stehen, das Produkt einsetzen und Feedback geben.

17. **[Q]** Was versteht man unter einem *On-Site Customer*?
[A] Einen Vertreter des Kunden, der bei der Programmierung anwesend ist und jederzeit für fachspezifische Fragen bereitsteht.

18. **[Q]** Welche Technik kommt am Beginn eines XP-Projekts zum Einsatz?
[A] Das Planning-Game.

19. **[Q]** Nennen sie drei der 12 XP-Techniken!
[A] 3 aus folgenden Antworten: Planning Game, Short Releases, On-Site Customer, Metaphor, Testing, Collective Ownership, Simple Design, Refactoring, Pair Programming, Continuous Integration, Coding Standards, 40 Hour Week.

20. **[Q]** Nennen sie einen Nachteil von XP gegenüber konventionellen Softwareentwicklungsprozessen.
[A] Z.B.: Schlechte Dokumentation des Systems, schlechte Skalierbarkeit, schlechte Eignung für bestimmte Arten von Software.