FACHHOCHSCHULE
TECHNIKUM WIEN

INFORMATIK

0100 STUDIENBRIEF

Version 3.0

SOFTWARE - LIFE - CYCLE MODELLE

was gestern die formel für den erfolg war,
wird morgen das rezept für niederlagen sein.
Glasow Arnold

Dipl.-Ing. Reinhard Oeser

INHALTSVERZEICHNIS:

1 ÜBERSICHT

1.1 LEHRSTOFF

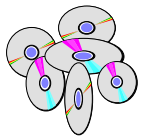
Die Anforderungen die an eine benutzerfreundliche Software gestellt werden sind in den letzten Jahren enorm gestiegen. Kam früher ein PC mit einer 10 MB Harddisk, 14" Monochrombildschirm und 128 **KB** RAM, Diskettenstation aus, so sind moderne PC´s mit 100.000 MB Harddisk, 19" Farbmonitor, 256 **MB** RAM, Soundkarte, DVD - Laufwerk, CD-ROM-Brenner, Soundkarte ausgestattet.

Anfang der 80er bis Ende der 80er Jahre war das Entwickeln von Software meist auf eine Person beschränkt. Sie führte die Gespräche mit den Kunden und implementierte (programmierte) danach die gesamte Anwendung.

➢ **Gestern**

Als ich 1989 meinen ersten Compiler (= "Software" zur Herstellung von Programmen) kaufte, so bestand das Handbuch von Turbo Pascal 4.0 aus dem Benutzerhandbuch (400 Seiten) und dem Referenzhandbuch (392 Seiten) und diesen Handbüchern waren 4 Disketten á 360 KB mit den Programmen beigefügt.

➢ **Und heute?**



Der Visual C++[1] Compiler Version 6.0 Standard wird auf einer CD ausgeliefert. Die Dokumentation ist auf zwei weiteren CD´s. (eine CD hat rund 700 MB Speichervolumen)

Planen, dann bauen - Erst denken dann handeln!

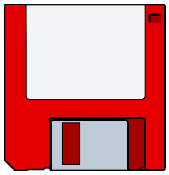
Wenn Sie ein Haus bauen, dann gehen Sie zu einem Architekten, der mit Ihnen gemeinsam das Haus plant und die Planung optimal auf Ihre Bedürfnisse und Anforderungen abstimmt. Der Architekt berät Sie fachlich und unabhängig. Aufgrund seiner Planung führt er für Sie eine Ausschreibung durch. Er beantwortet alle Fragen der Bieter und bewertet die Offerte. Sie beauftragen den Bestbieter mit der Bauausführung. Bei der Beauftragung liegt jedoch bereits eine entsprechende Leistungsbeschreibung vor. - Somit ist bei der Auftragsvergabe der Leistungsumfang klar definiert. Der Baumeister ist für die Durchführung der einzelnen Aufgaben verantwortlich. Während das Haus gebaut wird, überwacht der Architekt den Baufortschritt und gewährleistet, daß die Baufirma vertragskonform das Haus errichtet.

Für jeden Fachbereich ist es notwendig, die entsprechenden Spezialisten zu haben. Die Erstellung des Pflichtenheftes erfordert Kenntnisse der Analyse und des Designs. Die Erstellung eines Datenbankmodells setzt voraus, daß der Planer und Umsetzer über genaue Kenntnisse der

[1] Anmerkung: Fa. Microsoft hat den Nachfolger von C++ ist C# (sprich C sharp) bereits vorgestellt.

entsprechenden Datenbank verfügt. Die Programmierer müssen mit den entsprechenden Softwareentwicklungstools vertraut sein und der der die Software testet hat entsprechend systematisches Vorgehen zu wählen um möglichst viele Fehler zu finden. - Also den Nachweis zu erbringen, daß die Software Fehler enthält.

Ziel ist es Einblick in das ingenieurmäßige Vorgehen der Software - Entwicklung zu gewinnen. Sie sollen sich in die Lage eines Software- Engineers versetzen können und so nachvollziehen können, wie Software entwickelt wird und unter welchen Rahmenbedingungen dies geschieht.



Ausgehend vom Software - Life - Cycle werden wir die Konzepte der objektorientierten Software- Entwicklung kennenlernen.

Danach werden wir uns mit den Datenbankmodellen und Systemen beschäftigen und dann einen Ausflug in die Betriebssysteme, n-tier Architekturen, Total Cost of Owner Ship, Datensicherheit und Datenschutz unternehmen.

Abschließend werden wir uns beschäftigen, wie man Softwareprojekte managen kann. Der Gegenstand Informatik wird dann mit der Betrachtung von Entwicklungssystemen und einem Anwendungsbeispiel abgerundet.

1.2 ÜBUNGEN

In der Übung werden verschiedene Software - Entwicklungsmodelle miteinander verglichen und die Unterschiede herausgearbeitet.

1.3 LERNZIELE

- wissen
  - Was ist der Systemansatz.
  - Wissen was ist ein Software-Life-Cycle.
- verstehen
  - Verstehen wie Phasenmodelle funktionieren.
  - Verstehen wie Software professionell entwickelt wird.
- anwenden
  - Auswahl eines entsprechenden Software-Life-Cycles für die jeweilige Aufgabenstellung.
- beurteilen
  - Beurteilen, ob der verwendete Software-Life-Cycle problemadäquat ist.

1.4 ZIELGRUPPE

- Studiengang Elektronik und Wirtschaft, 1.Semester

1.5 VORAUSSETZUNGEN

- 0000Studienbrief
- technisches Verständnis, einfache Grundlagen der Informatik;

2 LERNWEGEMPFEHLUNG

2.1 WIE GEHEN SIE VOR

Nach Studium des Studienbriefes überlegen Sie welche Unterschiede zwischen den Softwareentwicklungen in verschiedenen Bereichen zu berücksichtigen sind.

Es ist ein Unterschied ob Sie Software für eine Bank, für die Steuerung eines Flugzeugs, eine Küchenmaschine oder ein Spiel entwickeln. Was haben Sie zu berücksichtigen?

Wenn Sie in der Übung die verschiedenen Software-Life-Cycle betrachten, über legen Sie sich, bei welchem Projekt welches Modell am effizientesten zum gewünschten Ziel (Softwareprodukt) führt.

3 DEFINITION DES BEGRIFFES SOFTWARE-TECHNIK [SE92]

Softwaretechnik = Software Engineering (engl.)

- *the systematic approach to the development, operation, maintenance and retirement of software [IEEE 83]*

  Die Techniken des Software-Engineerings beziehen sich also nicht nur auf die Entwicklung, sondern auf das Durchziehen alle Lebensphasen von der Entwicklung über Betrieb, Wartung bis hin zur Stillegung. Alles muß systematisch bearbeitet werden.

- *the technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost estimates*
  [Fairley 1985]

  Hier werden darüber hinaus weitere wichtige Managementaspekte ins Spiel gebracht. Die Softwareentwicklung muß nach wirtschaftlichen Gesichtspunkten durchgeführt werden.

- *the practical application of scientific knowledge for the economical employment of reliable and efficient software.*
  [Pomberger 1986]

  Weitere wichtige Gesichtspunkte sind Qualitätskriterien wie Zuverlässigkeit und Effizienz und eine wissenschaftliche Grundlage.

Ingenieurmäßiges vorgehen, wie es auch in anderen technischen Disziplinen (z.B.: Architektur) üblich ist, sollte das Problem der immer komplexer werdenden Software, die nicht mehr wartbar und kaum mehr zuverlässig ist, lösen. Software Engineering war ursprünglich die Antwort auf die Softwarekrise in den 60er Jahren.

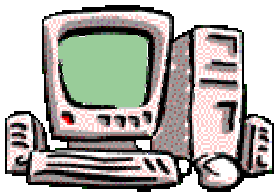
4 ZIELE DER SOFTWARETECHNIK [MODF. SE92]

Folgende Ziele verfolgt die Softwaretechnik:

4.1 ZIELE AUS SICHT DER ANWENDER:

- exakte Erfüllung der Benutzeranforderungen
- bessere Benutzeroberflächen
- Software gliedert sich in den Geschäftsprozeß ein

4.2 ANFORDERUNGEN AN DIE TECHNIK



- mehr Zuverlässigkeit und Robustheit
- mehr Modularisierung (des Produktes und der Entwicklung)
- leichtere Wartbarkeit und Erweiterbarkeit
- Wiederverwendbarkeit von Software- Komponenten
- Unabhängigkeit von Betriebssystemen
- Gewährleistung des Datenschutzes und der Datensicherheit

4.3 BETRIEBSWIRTSCHAFTLICHE ANFORDERUNGEN

- Senkung der Kosten (Herstellung, Betrieb, Wartung)

5 DER GESTALTUNGSANSATZ

Jeder Problemlösungsprozeß erfordert den Einsatz von zwei voneinander abgrenzbaren „Komponenten zur Projektlösung“ [Dae77], nämlich den Einsatz von zwei grundsätzlich verschiedenen Gruppen von Methoden und Techniken:

- Methoden und Techniken für die Systemgestaltung als eigentliche konstruktive Arbeit den Einsatz von **Problemlösungsmethoden und -techniken**

- Methoden und Techniken für die Projektführung, d.h. die Fragen der Organisation und Koordination des Problemlösungsprozesses, den Einsatz von **Projektführungsmethoden und -techniken** zur Unterstützung der inhaltlichen Teilaufgaben

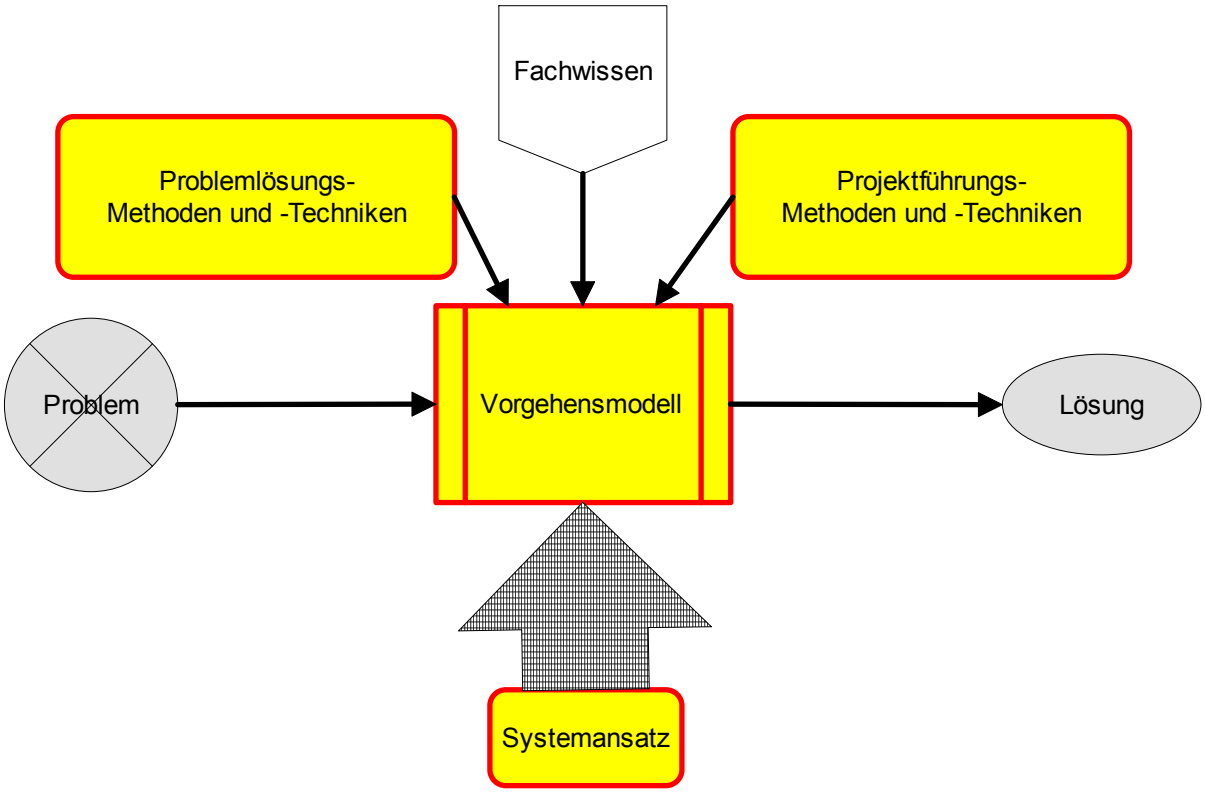


Abbildung 1 Schematisches Modell zur „Gestaltungsphilosophie systemorientierten Planens“ („Gestaltungsansatz“); [modifiziert Woj94, Seite GG1 ff]

Um ein Problem lösen zu können sind zwei Standpunkte festzulegen:

- systematische Betrachtung der Systeme - „Statische Situation“
- systematisches Vorgehen bei der Problemlösung - „Dynamischer Prozeß“

5.1 Grundlegende Anforderungen an die Gestaltung

Um ein erfolgreiches Vorgehen zu gewährleisten müssen folgende Anforderungen bei der Gestaltung von Prozessen („Gestaltungsansatz“) erfüllt sein:

5.1.1 Ganzheitlich / Systemisch

Die Problemlösung erfolgt unter Einbeziehung sämtlicher relevanter Faktoren, die auf das Projekt einen Einfluß ausüben. All diese Faktoren sind bereits in der Analysephase zu berücksichtigen.

5.1.2 Situativ

Das Vorgehen muß den individuellen Gegebenheiten und Erfordernissen in der jeweiligen Planungssituation angepaßt werden. Softwareentwicklung reicht von der Programmierung einer Steuerung einer Waschmaschine bis hin zur elektronischen Krankengeschichte. Die Planung und Realisierung ist jedoch entsprechend den Anforderungen anzupassen. Weiter ist auch das Umfeld in dem die Anwendung eingesetzt wird entsprechend einzubinden und zu berücksichtigen. In der Planungsphase ist es unumgänglich die Bildung von verschiedenen Alternativen durchzuführen und aus denen dann für das Problem die beste auszuwählen.

5.1.3 Strategisch

Berücksichtigung der Unternehmensziele, Produkt bzw. Dienstleistungspalette, interne und externe Randbedingungen, die auf die Gesamtstruktur eines Unternehmens ausgerichtet sind. Software ist ein Betriebsmittel, daß die Ablauf und Aufbauorganisation maßgeblich beeinflußt oder so zu gestalten ist, daß die Ablauf- und Aufbauorganisation optimal den geforderten Rahmenbedingungen entspricht.

5.1.4 Inhalts- und vorgehensorientiert

Der Gestaltungsansatz ist dann als inhalts- und vorgehensorientiert zu bezeichnen, wenn

- für eine Um- oder Neugestaltung sowohl der Inhalt der Veränderung,
- als auch das Vorgehen bei der Gestaltung,
- sowie deren Wechselwirkung in Bezug auf das Gestaltungsergebnis

methodisch berücksichtigt werden.

Für komplexe Systeme - wie es Softwaresysteme sind - ist es unumgänglich, daß die Planung und Realisierung methodisch erfolgt.

5.2 Besondere Merkmale des Gestaltungsansatzes

Der Gestaltungsansatz zeichnet sich durch folgende Aspekte besonders aus

- Verwendung des Systemansatzes
  - Problemlösungsprozeß erfolgt durch ein Vorgehensmodell
  - paralleler Einsatz von
    - Problemlösungsmethoden und Techniken, sowie
    - Projektführungsmethoden und Techniken
  - Fachwissen das von den Betroffenen in das Projekt einfließt

Im folgenden werden die Aspekte im Bereich der Softwareentwicklung entsprechend beleuchtet:

5.2.1 Systemansatz

Forderungen des Systemansatzes [modifiziert nach Pat82]

Der Systemansatz stellt die Basis für die Erfassung und Gestaltung komplexer Systeme sicher, dabei sind folgende Eigenschaften zu erfüllen

- inhaltliche Abstraktheit
- strukturierende Wirkung
- Möglichkeit zum interdisziplinärem Wissensaustausch und somit für alle Beteiligten verständlich abzufassen.

Systemdenken garantiert ein *„formal-abstraktes, objektunabhängiges, technologieunabhängiges, umfassendes Behandeln beliebiger Problemsachverhalte“*, durch systemorientierte, ganzheitliche Betrachtungsweise.

5.2.2 VORGEHENSMODELL

Bei der Software - Entwicklung haben sich mehrere Vorgehensmodell entwickelt. Das hier vorgestellte beschreibt einen systematischen nahezu sequentiellen Ablauf. Weitere Modelle werden Sie in der Übung kennenlernen.

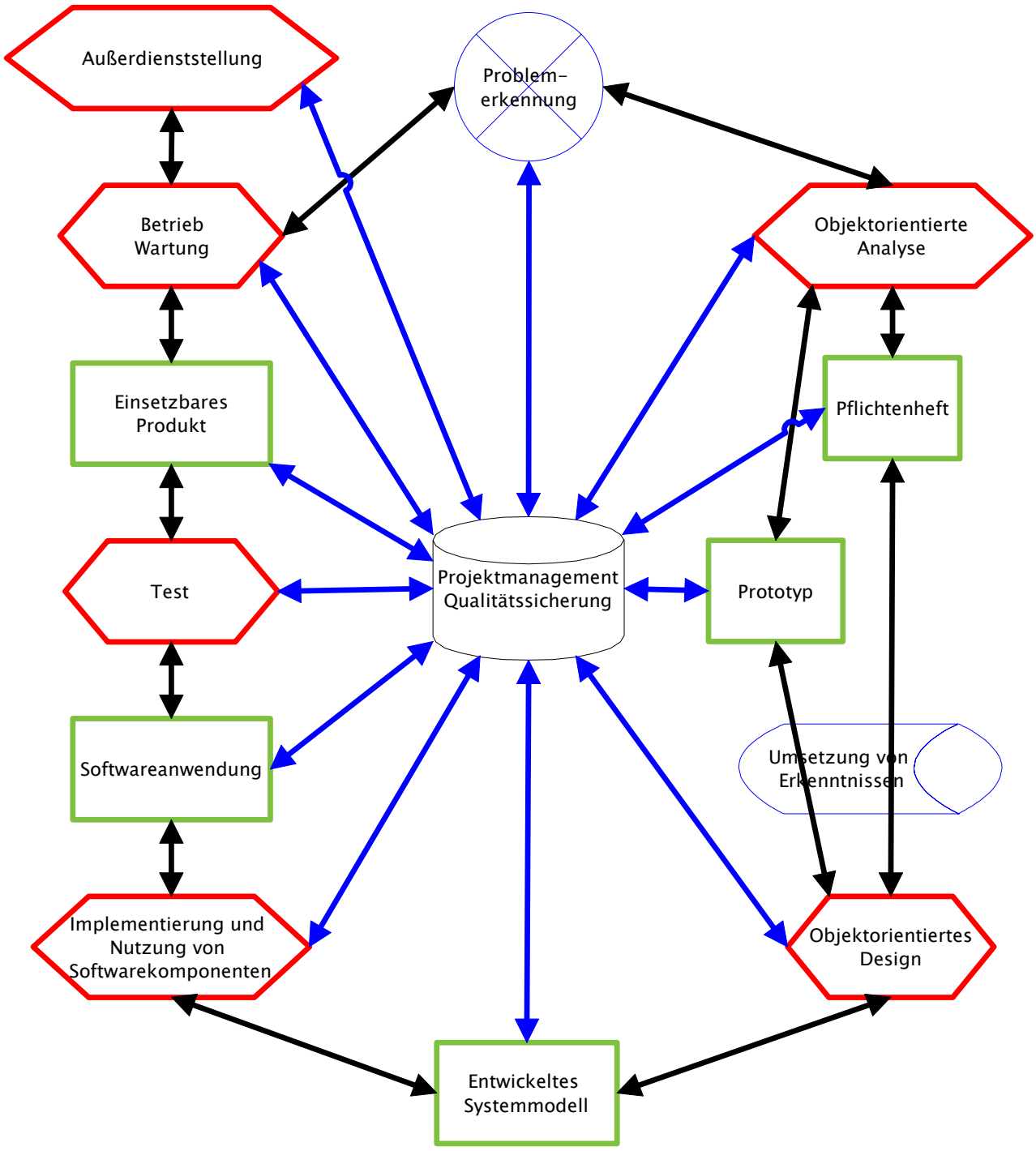


Abbildung 2 Software - Life - Cycle

DER SOFTWARE - LIFE - CYCLE [MODIFIZIERT NACH OE99.1/.2]

Die verschiedenen Vorgehensmodell bei der Softwareentwicklung (Software-Life-Cycle) beschreiben, welche Aktivitäten in welcher Reihenfolge in welcher Zeit durch die entsprechenden Fachleute zu realisieren sind.

Jedes Modell setzt gewisse Schwerpunkte. Es wird sofort ersichtlich, daß ein internetbasiertes Fahrplanauskunftsprogramm anderes zu entwickeln ist, als jene Software, die in einem Auto das Antiblockiersystem (ABS) steuert. Entsprechend dieser Anforderungen werden die Vorgangsmodelle entsprechend modifiziert.

Das in Abbildung 2 beschriebene Modell stellt einen idealisierten Ablauf im Bereich der kommerziellen Datenverarbeitung vor.

Im Zentrum des Geschehens begleitet das Projektmanagement und die Qualitätssicherung (siehe 3. bzw. 4. Semester) den gesamten Softwareentwicklungsprozeß. Unter Projektmanagement wird die Gesamtheit von Führungsaufgaben, -organisation, -techniken und -mittel für die Abwicklung eines Projektes verstanden (DIN 69901). Die Qualitätssicherung stellt auf der einen Seite sicher, daß das richtige Produkt gemacht und auf der anderen Seite, daß das Produkt richtig gemacht wird.

Um ein solches Projekt erfolgreich abwickeln zu können, bedarf es eines Projektleiters der sowohl die Problemlösungsmethoden, -techniken, Projektführungsmethoden und -techniken beherrscht.

Problemerkennung und formale Beschreibung

Der Projektausgangspunkt bildet ein Problem, das objektiv oder subjektiv wahrgenommen wird. Durch die formale Beschreibung erfolgt eine systematische Auseinandersetzung und Analyse des Problems. Folgende Projektcharakteristiken sind zu definieren:

- Welche Ziele und welcher Nutzen sollen nach der Softwareimplementierung erreicht werden
- Projektumfang (=Was soll realisiert werden?)
- Frage nach der Größe und Umfang des Projektes (z.B.: Abteilungsebene, gesamtes Unternehmen)
- welche Schnittstellen (Lieferanten, Kunden, Internetbestellungen werden direkt in das Bestellwesenprogramm übertragen) bzw. sind zu berücksichtigen
- externen Projektinteraktionen (Welche Auswirkungen hat das Projekt auf andere Abteilungen bzw. externe Einrichtungen),
- Rahmenbedingungen (z.B.: Technologien, Gesetze, Datenschutz und Datensicherheit, Normen),
- mit welchen Projektrisiken zu rechnen ist und wie diese minimiert werden können.

Nach Evaluierung der formalen Beschreibung führt man eine Projektbewertung durch. Wenn das Projekt immer noch als sinnvoll erachtet wird, beginnt der wichtigste Projektabschnitt, die Analyse.

Objektorientierte Analyse

Die Analyse ist die wichtigste Phase im Projektverlauf. In diesem Projektabschnitt wird die Funktionalität der Software vollständig und exakt beschrieben. Hier geht es um das **WAS**, das die Softwarelösung können soll und nicht **WIE** es realisiert werden kann. Die Kunst dabei ist es die Beschreibung der Lösung so abzufassen, daß sowohl die zukünftigen Nutzer als auch die Systementwickler den Inhalt verstehen und diesen gleich interpretieren können. Gemeinsam mit den zukünftigen Anwendern wird eine IST-SOLL Analyse durchgeführt und jene Abläufe beschrieben, die automatisationsunterstützt ausgeführt werden sollen und es werden die Informationsbedürfnisse erhoben – welche Daten soll das System speichern, verarbeiten und auswerten. Mit Hilfe von Maskengeneratoren wird mit den Usern die Benutzeroberfläche gestaltet. Heute ist es bereits möglich auch die Datenerfassung mit Hilfe von Prototypen auszuprobieren und so den Einsatz und die Brauchbarkeit der Lösung zu simulieren und zu optimieren. Die Anforderungen an die Softwaredokumentation (Entwicklungsdokumentation, Anwenderdokumentation) werden festgelegt. In Hinblick auf zukünftige Entwicklungen werden die Wartungs- und Weiterentwicklungsanforderungen, sowie die Wachstumsprognosen formuliert. Dies ist insofern wichtig, da man die Entwicklungskosten nicht isoliert von den Betriebskosten sehen darf. Weiters sind folgende Parameter festzulegen: Zuverlässigkeit (max. Stillstandszeit pro Jahr), die Betriebszeiten (z.B.: Notfallaufnahme rund um die Uhr, Ambulanzen jeden Werktag von 8 – 17 Uhr) und andere Parameter - wie Anzahl der User, Infrastruktur etc. .

Das Ergebnis dieser Problembetrachtung wird im sogenannten Pflichtenheft zusammengefaßt. Wichtig in diesem Zusammenhang ist, daß auch ein Konzept erstellt wird, das einen Notbetrieb ohne EDV jederzeit ermöglicht. Gewisse Grundinformationen sind daher stets auch auf Papier in Evidenz zu halten.

Als Instrumente der Visualisierung wird hier die graphische Beschreibung UML vorgestellt.

UML steht für "Unified Modeling Language" und ist eine Notation (hauptsächlich diagrammorientiert) für die objektorientierte Modellierung, die auf diverse OO-Konzepte aufsetzt und diese zusammenfaßt. Gemäß dem Grundsatz "Ein Hammer macht noch keinen Architekten", ist auch die Notation UML nur als ein Hilfsmittel für den OO-Entwicklungsprozeß zu sehen. Sie definiert keine Vorgehensweise und keine Methodik der OO-Modellierung, sondern bietet für die verschiedenen ursachengetriebenen Ansätze und Methoden die passenden Diagramme und Notationen für eine einheitlichen Darstellung. [ISB02/99]

Die Analyse wird mit Hilfe der Methode von Peter Coad and Yourdan durchgeführt.

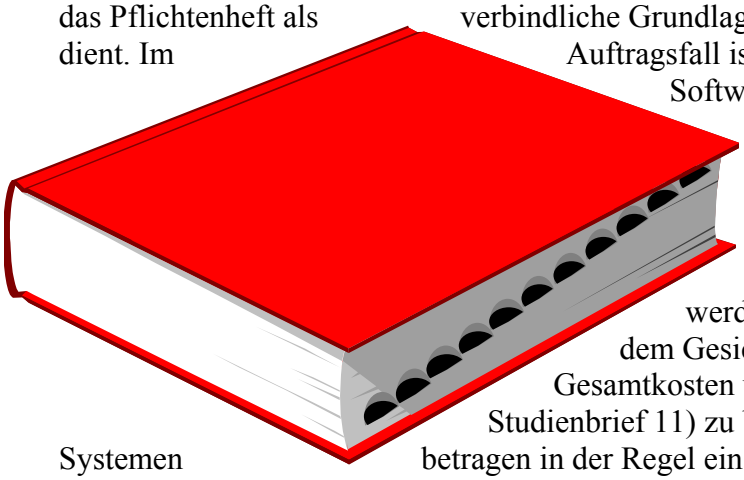
Für interessierte Leser werden folgende Bücher empfohlen:

[Coad91/1] Coad P., Yourdon E.; *Object-Oriented Analysis (Second Edition)*; YourdonPress, 1991;

[Coad91/2] Coad P., Yourdon E.; *Object-Oriented Design*; YourdonPress, 1991;

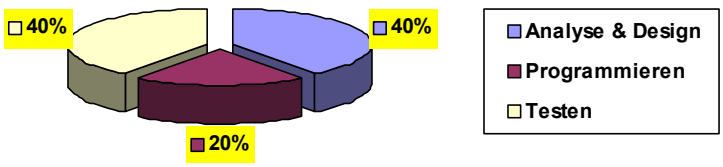
Pflichtenheft – Prototyp

Das Pflichtenheft beschreibt nun die funktionalilität der Softwarelösung und die Anforderungen (durch Leistungsangaben z.B.: Ausfallszeit, maximale Antwortzeiten, Erweiterbarkeit etc.) an die technische Realisierung. Das Pflichtenheft erfüllt nun zwei Aufgaben. Die erste Aufgabe ist es, daß das Pflichtenheft als verbindliche Grundlage für ein Angebot eines Softwareherstellers dient. Im Auftragsfall ist das Pflichtenheft der Leistungskatalog der Software. Die zweite Aufgabe ist es, Erkenntnisse die bei der Erstellung des Pflichtenheftes gewonnen wurden sind und direkt realisierbar sind auch sofort umzusetzen. Vor der Ausschreibung sollte der Bewertungsvorgang bereits festgelegt werden. Die Bewertung der Angebote ist unter dem Gesichtspunkt der TCO (Total Cost of Ownership, Gesamtkosten über den Nutzungszeitraum - siehe Studienbrief 11) zu betrachten. Die Betriebskosten von EDV Systemen betragen in der Regel ein Vielfaches der Entwicklungs- bzw. Anschaffungskosten.



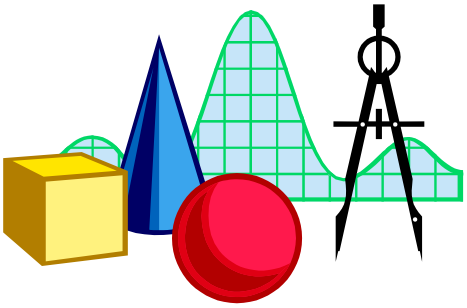
Nach Einholung der Offerte muß das Projekt nochmals kritisch hinterfragt werden. Es müssen die Investitionskosten im Verhältnis zum erwartenden Nutzen gesetzt werden. Bei einer positiven Projektentscheidung und nach Auswahl des Bieters und abschließenden Verhandlungen, werden die Verträge geschlossen und man beginnt mit der tatsächlichen Umsetzung.

Nach der 40:20:40 Regel liegen 40% des Aufwandes bei der Analyse und dem Design, nur 20% bei der klassischen Programmierung und 40% beim Testen.



Design

Als Ausgangspunkt des Designs dient das Pflichtenheft. Dieses Dokument beschreibt formal, **WAS** die Softwarelösung können soll. Nun soll festgelegt werden, **WIE** die Software tatsächlich realisiert werden kann. Jetzt entscheiden die Softwareentwickler mit welchen Softwareentwicklungstools die Software entwickelt, welche Datenbank gewählt und auf welcher Computerplattform das System realisiert wird und mit welcher Technologie das Netzwerk aufgebaut wird. Diese Entscheidung wird aufgrund der Anforderungen des Pflichtenheftes getroffen. - Echzeitsysteme können nur auf einer Plattform laufen, die auch tatsächlich dafür konzipiert sind. Aus heutiger Sicht scheint die beste Lösung - bei Projekten im der kommerziellen Datenverarbeitung - sich möglichst jener Technologien zu bedienen, die auch im Internet zur Anwendung kommen.



Ziel dieser Technologie ist es die Softwarekomplexität vom Anwender auf einen zentralen Produktionsrechner zu verlagern und der Anwender bedient die Software über einen Browser wie im Internet. Der Vorteil liegt auf der Hand. Der lokale Computer dient lediglich als Anzeigegerät und die Daten werden in einem zentralen Produktionsrechner gespeichert und gesichert. Fällt der lokale Rechner aus, so kann man ihn ohne Probleme ersetzen. Der Produktionsrechner ist meist redundant ausgelegt und somit sehr ausfallssicher. Das Prinzip ist wie die Energie aus der Steckdose. Wenn Sie ein Gerät an die Steckdose anschließen, denken Sie in dem Moment nicht darüber nach, wie die Energie erzeugt wird und zu Ihnen kommt. Die Komplexität der Energie - Erzeugung bleibt ihnen verborgen.
Zwischen den Softwaredesignern und den Anwendern muß es einen Dialog geben, der sicherstellt, daß das Pflichtenheft richtig interpretiert und umgesetzt wird.

Viele Softwarehersteller haben diesen Trend erkannt und setzen voll auf diese Technologie. In Zukunft wird man Software "nicht mehr kaufen" sondern nur noch mieten können. Im Prinzip stehen wir derzeit bei der dritten Generation:

- **1. Generation**: zeichenorientierte Betriebssysteme / Programme (DOS, Unix ...)
- **2. Generation**: graphikorientierte Systeme (Windows Familie (3.x, 95, 98, ME, Windows NT, Windows 2000, Windows XP, OS/2, Wrap, Apple OS, X-Motif etc.)
- **3. Generation:** webbasierte Systeme - Programme laufen in einem Browser - plattformunabhängig

Implementation

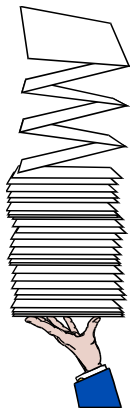
Aufgrund des Designs wird nun tatsächlich das Programm geschrieben. Viele Programmabschnitte werden jedoch automatisch erstellt und brauchen nicht mehr codiert zu werden. Darunter fallen die Eingabemasken (Erstellung mit einem Maskengenerator), Datenbankstruktur (Erstellung mit einem Datenbankdesigner), die Druckerausgaben werden mit einem Reportgenerator erstellt. Dies ermöglicht eine raschere und kostengünstigere Entwicklung und der Wartungsaufwand läßt sich dadurch senken. Spezielle Anforderungen müssen jedoch weiterhin programmiert werden.
Es gibt immer mehr Tools die die Analyse das Design, die Codierung und den Softwaretest unterstützen.

Test

One should start with the assumption that the program contains errors and then test the program to find as many of the errors as possible. .... Testing is the process of executing a program with the intent of finding errors [Meye79].

Testen bedeutet nicht nachzuweisen, daß die Software die Anforderungen erfüllt. Testen heißt methodisch nachzuweisen, daß das Programm Fehler enthält.

Nicht nur Programme können getestet, sondern auch alle Dokumente, die während des Software - Entwicklungsprozesses erstellt werden, werden mittels eines Reviews („Test“ der Dokumente) geprüft. Bei einem Review wird das Dokument an alle Projektteilnehmer versendet, wobei jeder einen Prüfschwerpunkt zugeteilt bekommt. (Ärzte: Prüfung der Diagnoseerfassung, Pflege: Pflegedokumentationssteil, Finanzchef prüft, ob alle Statistiken erstellt werden u.s.w.). In einer gemeinsamen Sitzung werden die Unklarheiten und Fehler besprochen und korrigiert. Diese Aufgabe ist ein Teil der Qualitätssicherung.



Der Programmcode wird einem White – Box Test und einem Black – Box Test unterzogen.

- Beim White – Box Test wird das Programm mit Testdaten – welche systematisch erstellt werden – so ausgeführt, daß mindestens jedes Programmstatement ausgeführt wird und ebenso wird verglichen, ob der Programmteil das gewünschte Ergebnis liefert.
- Der Black – Box Test wird ohne Kenntnis der Programmstruktur ausgeführt. Die Testfälle werden aus dem Pflichtenheft mit den Anwendern gemeinsam abgeleitet. Hier ist wieder die Kooperation zwischen Software – Entwickler und Anwender gefragt. Aber auch beim Testen besteht die Möglichkeit mittels spezieller Software die Tests teilweise automatisch durchzuführen.

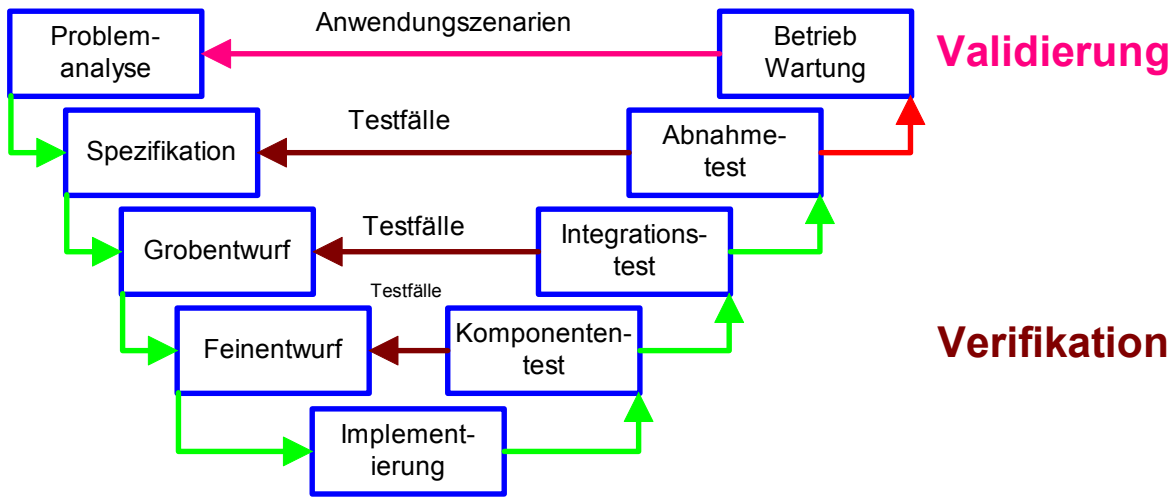
Begriff: Validierung- Verifikation [Bal962]

Unter Valierung wird die Eignung bzw. der Wert des Produktes bezogen auf seinen Einsatzzweck verstanden.

Wird das richtige Produkt entwickelt?

Unter Verifikation wird die Überprüfung der Übereinstimmung zwischen einem Software- Produkt und seiner Spezifikation (Pflichtenheft) verstanden.

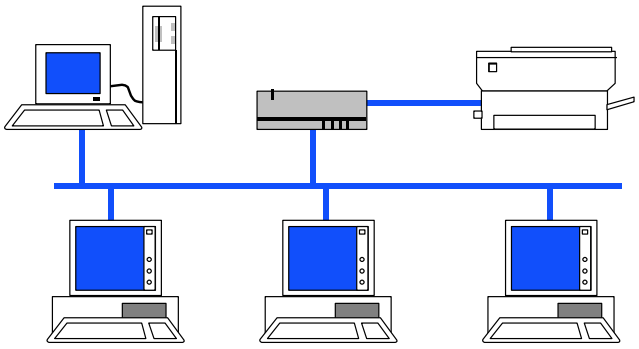
Wird ein korrektes Produkt entwickelt.



(Abbildung zeigt das V- Modell)

Betrieb und Wartung

Wenn das System im Test zeigt, daß es der geforderten Stabilität genügt und es den Anforderungen des Pflichtenheftes gerecht wird, kann der Vollbetrieb aufgenommen werden. Parallel zum Probebetrieb muß die Arbeitsorganisation an das EDV – System angepaßt werden und die Mitarbeiter in das neue System eingeschult werden. (Es sollte schon in der Analysephase die Organisation optimiert werden. Die Softwareanforderungen sollten basierend der optimierten Organisation gestellt werden.). Um so besser die Mitarbeiter mit dem System umgehen können, um so höher ist die Akzeptanz und umso weniger Probleme treten im Routinebetrieb auf. Die meisten Schwierigkeiten beim Betrieb entstehen durch Unkenntnis der Anwender. Ein Ausfall wegen eines Hardwaredefektes ist heute durch die redundante Auslegung kaum gegeben.
Im Produktionsbetrieb sammelt man Erfahrung mit dem EDV-System, welche man bei der Wartung und der Systemoptimierung berücksichtigt. Wie schon in der Systembetrachtung ausgeführt, ist es unbedingt notwendig bei der Investitionsentscheidung auch die laufenden Betriebskosten zu berücksichtigen und auf die Ausbaufähigkeit des Systems zu achten. Weiters bedarf ein EDV-System einer permanenten Betreuung. Unter diesen Agenden fallen zum Beispiel Schulung, die Datensicherung und Lagerung, Reorganisation der Festplatten, Virenschutz und deren Aktualisierung, Benutzeradministration, Systemtuning, Zugriffsschutz von außen, Softwareaktualisierung, usw.



Außerdienststellung

Warum sich eigentlich schon bei der Entwicklung Gedanken über die Außerdienststellung machen? – Daten die in einem EDV – System gespeichert sind müssen bei der Installation in das System importiert werden. Meistens sieht man jedoch keine Exportschnittstelle vor. Will man die Daten nach einigen Jahren doch in ein neues EDV – System übernehmen so ist dies nur mit erheblichem Aufwand möglich, da meist die Kenntnis über das bestehende System verlorengegangen ist und man meist nicht immer regelmäßig die neuesten Updates gekauft hat. Daher muß man bereits in der Planungsphase dies berücksichtigen und auch gleich bei der Entwicklung diese Schnittstelle implementieren und testen.

Resümee:

Softwareentwicklung kann man mit dem Bau eines Hauses vergleichen. Am Anfang muß alles genau geplant werden bevor man mit der tatsächlichen Umsetzung beginnt. Änderungen im Nachhinein sind mit hohen Kosten verbunden oder gar nicht durchführbar. (z.B.: nachträgliche Einbau eines Lifts, der in der Planung nicht vorgesehen war etc.)
Software – Entwicklung im medizinischen Bereich ist eine Herausforderung für die Software – Industrie und bedarf eines professionellen Consultings.

Wichtig ist, daß man die zwei Projektphasen Analyse und Design klar voneinander trennt werden.

1. In der ersten Phase muß analysiert werden, WAS das System können soll und unter welchen Rahmenbedingungen das Projekt durchgeführt werden soll.
2. In der zweiten Phase wird die Software nach den Plänen der Systembetrachtung entwickelt. Dabei handelt es sich um grundlegend andere Aufgaben. Durch eine unabhängige Analyse ist der Auftraggeber bis zum Entwicklungsauftrag technologisch ungebunden und kann vom Softwarehersteller ein Fixpreisangebot einholen.

Ausblick in die Zukunft:

Softwareprojekte können nur dann erfolgreich Realisiert werden, wenn man die Grundzüge des vorhin beschrieben Modells einhält.

Studien [Prin99] der Firmen IBM, GTE, TRW haben gezeigt, daß die Beseitigung von Fehlern im Produktionsbetrieb 200x mehr Kostet als wenn man den Fehler bereits in der Analyse behoben hätte. Bei einer Fehlerbehebung im Produktionsbetrieb sind alle Phasen des Software - life -cycles neu zu durchlaufen.

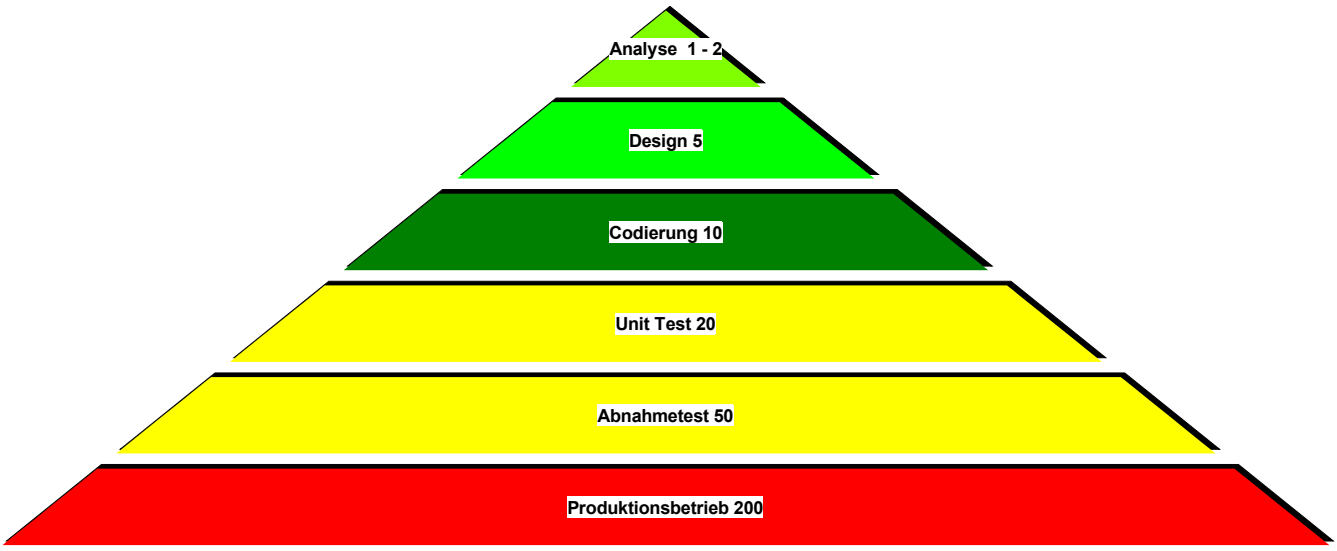


Abbildung 3 Phase in der Software-Entwicklung - Faktor der Kosten bei Fehlerbehebnung

5.2.3 PROBLEMLÖSUNGSMETHODEN UND TECHNIKEN

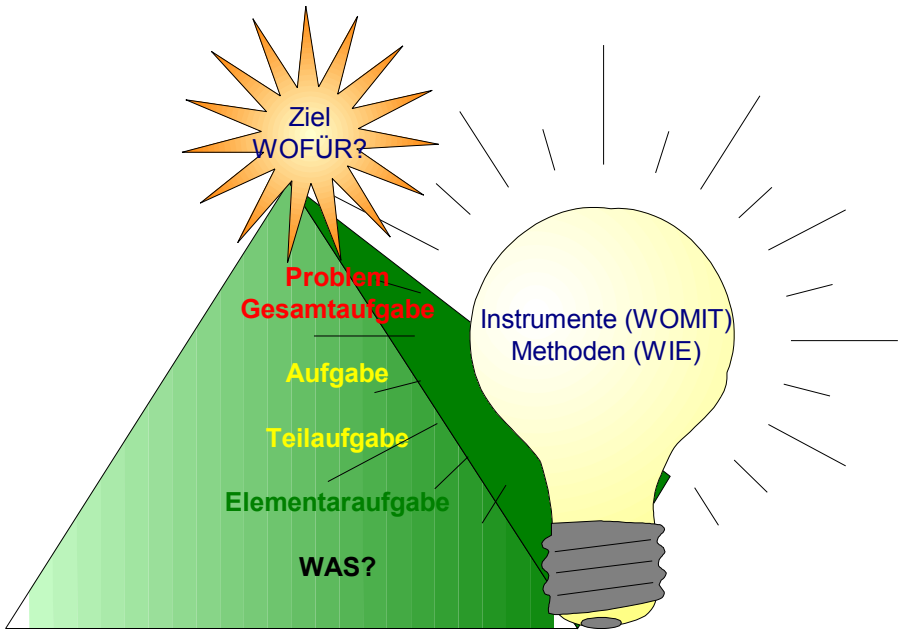


Abbildung 4 Zuordnung Problem, Aufgabe, Ziel, Instrumente, Methoden [modifiziert nach Wojda94]

Ausgehend von der Fragestellung "Wofür" (Definition der Ziele), erklärt das "WAS" was gemacht werden soll (Definition der Aufgaben). Die Frage muß nun geklärt werden, "WIE" man nun das Ziel erreichen kann.

Aufgaben: inhaltliche Beschreibung des Problemlösungsschrittes (Was soll gemacht werden?)
Methoden: Beschreibt den Weg zur Lösung der Aufgabe und legt in objektiver Weise, eine endliche, geordnete Anzahl von Vorschriften/Regeln fest. (Wie?)
Analysemethode von Coad P., Yourdon E - Objektorientierte Analyse und Objektorientiertes Design;
Instrumente: Dienen im Sinne von Hilfsmitteln zur Unterstützung der Methodenanwendung bei der Aufgabendurchführung. (Womit?) *z.B.: UML*

Planungsmethoden (lt. Bechmann 1981, S.117) sind ein Rezept, dienen zur Reduktion der Komplexität, beeinflussen das Lösungsverhalten und garantieren in formaler Hinsicht eine Lösung des Problems.

5.2.4 PROJEKTFÜHRUNGSMETHODEN UND TECHNIKEN [OES98]

Die generellen Projektmanagementaufgaben bilden die Basis, um Softwareprojekte durchzuführen.

Grundsätzlich besteht der Projektmanagement - Regelkreis aus:

- **Projektplanung** regelt
  - die Aufgaben-, Verantwortungs- und Kompetenzverteilung insbesondere die Zusammenarbeit zwischen den einzelnen Mitarbeitern und Teams
  - wer ist in welchen Phasen wie beteiligt
  - Entscheidungsregeln und Abläufe - Kommunikationsstrukturen
  - Aufgaben des Projektleiters

- **Projektsteuerung** sind jene Hilfsmittel, die das Projekt begleiten und Informationen über den Projektstatus liefern. Diese Daten werden zum Regeln der Abläufe verwendet, um Abweichungen vom IST-SOLL zu korrigieren. Sollten sich im Projektfortschritt projektrelevante Veränderungen ergeben, so sind diese in der Projektsteuerung zu berücksichtigen.

  Hier werden insbesonders
  - Ablauf- und Terminmanagement
  - Ressourcenmanagement
  - Budgetmanagement
  - Gesamtprojektmanagement

  durchgeführt

- **Projektdokumentation** dient zur raschen und effizienten Orientierung im Projekt. Sie hält die gewonnenen Erkenntnisse und Ergebnisse fest.

- **Berichtswesen** sorgt für den raschen Informationsfluß zwischen den Beteiligten.

5.2.5 FACHWISSEN

Schlußendlich muß man bei der Projektdurchführung über das nötige Fachwissen verfügen um ein Projekt durchführen zu können.

6 ÜBUNGSTEIL

Die Übung erfolgt in Präsenz (4 Stunden) und in der Fernlehre (6 Stunden) pro Lernwochenzyklus.

6.1 LERNKONTROLLE

Es stehen Ihnen im Fernlehrsystem automatisierte Tests zur Selbstkontrolle zur Verfügung.

6.2 ÜBUNGSBEISPIELE

Die Übungsbeispiele sind im Fernlehrsystem unter www.telefh.at. Entsprechend der Studienbriefnummer 0100 finden Sie die entsprechenden Übungsaufgaben.

7 LITERATUR


Wer sich mit dem Thema eingehender beschäftigen möchte findet in den folgenden Lehrbüchern die Möglichkeit sich weiter zu vertiefen:

[Bal96] Balzert Heide; *Methoden der objektorientierten Systemanalyse;* 2. Auflage; Spektrum Akademischer Verlag GmbH. (kurze Zusammenfassung - Überblick)

[Bal961] Balzert Helmut; *Lehrbuch der Software-Technik - Software-Entwicklung*; Spektrum Akademischer Verlag GmbH; 1996. (sehr detailliert !!)

[Bal962] Balzert Helmut; *Lehrbuch der Software- Technik - Software- Management, Software - Qualitätssicherung, Unternehmensmodellierung*; Spektrum Akademischer Verlag GmbH; 1996. (sehr detailliert !!)

[Bal99] Balzert Heide; *Lehrbuch der Objektmodellierung (Analyse und Entwurf);* Spektrum Akademischer Verlag GmbH, 1999. (sehr detailliert !!)

[Dae77] Daenzer (1977, S. 8) aus [Woj94].

[Pat82] Patzak G.; *Systemtechnik – Planung komplexer Systeme. Grundlagen; Methoden, Techniken; Berlin, Heidelberg,* New York: Springer 1982.

[Coad91/1] Coad P., Yourdon E.; *Object-Oriented Analysis (Second Edition)*; YourdonPress, 1991

[Coad91/2] Coad P., Yourdon E.; *Object-Oriented Design*; YourdonPress, 1991

[Oes98] Oeser; *Software Projektmanagement aus Auftraggebersicht zur Umsetzung telemedizinischer Konzepte*; Diplomarbeit 1998.

[Oe99.1] Oeser; *Softwarelösungen nach dem Bausteinprinzip Teil 1*; managemed Ausgabe 2-3/1999.

[Oe99.2] Oeser; *Softwarelösungen nach dem Bausteinprinzip Teil 2*; managemed Ausgabe 4-5/1999.

[SE92] Weitere Grundlage der Vorlesung ist das Software - *Engineering Skriptum (1992*) von Herrn Prof. DI Dr. Futschek, daß unter meiner Mitwirkung entstanden ist.

[Woj94] Wojda Franz; *Grundlagen der Arbeitswissenschaften*; Skriptum Institut für Betriebswissenschaften; Arbeitswissenschaften und Betriebswirtschaftslehre.

7.1 WEITERFÜHRENDE LITERATUR

[Balz84] Balzert H.;Die Entwicklung von Software-Systemen;B.I.-Wissenschaftsverlag, 1984

[Balz85] Balzert H.; Moderne Software-Entwicklungssysteme; B.I., 1985, Reihe Informatik / 44

[Borl901] Borland GmbH.; *Turbo C++*; 2. Auflage 1990

[Borl902] Borland GmbH.; *Turbo Pascal 6.0 mit Turbo Vision*; 1. Auflage 1990

[Dahl72] Dahl, Dijkstra, Hoare; *Structured Programming*; Academic Press, 1972

[Dene91] Denert E.; *Software-Engineering*; Springer-Verlag, 1991

[Enge88] Engesser H.; *Duden Informatik*; Dudenverlag, 1988

[Fair82] Fairley R.; *Software Engineering Concepts*; McGraw Hill, 1982

[Gilb88] Gilb T.; *Principles of Software Engineering Management*; Addison-Wesley, 1988

[Hals77] Halstead M.H.; *Elements of SW-Science*; Elsevier North-Holland, 1977

[Hans] Hansen H.R.; *Wirtschaftsinformatik*; UTB 803

[Heu92] Heuer, Andreas; *Objektorientierte Datenbanken / Konzepte, Modelle, Systeme;* Addison-Wesley Deutschland GmbH; 1992

[ISB2/99] http://www.isb-mz.de/zeitung/02-99/artikel2-1.htm ff.

[Jack75] Jackson M.A.; *Principles of Program Design*; Academic Press, London, 1975

[Jack85] Jackson M.A.; *Systementwicklung: Die JSD-Methode*; Hanser Verlag, 1985

[Kern74] Kernighan, Plauger; *The Elements of Programming Style*; McGraw-Hill, 1974

[Kimm79] Kimm, Koch, Simsonsmeier, Tontisch; *Einführung in Software Engineering*; de Gryter, 1979

[Kitc90] Kitchenham B.; *Software Development Metrics and Models*; ed. Paul Rock, Elsevier Applied Science,1990

[Marc88] Marca D.A.; *SADT: Structured Analysis and Design Technique*; McGraw-Hill, 1988

[McCa82] McCabe T.J.; *Structured Testing*; Silver Spring, MD:IEEE Computer Society Press,1982

[McCa91] McCall J., Markham D.; *The automated Measurement of Software Quality*; Compsac 81,pp. 52-58, 1991

[McCa77] McCall J.,P. Richards P., Walters G.; *Factors in Software Quality*; NTISAD-A049-014/015,AD-A049-055,Nov77,3 Volumes

[Meye88] Meyer B.; *Object-oriented Software Construction*; Prentice Hall, 1988

[Myer79] Myers G.J.; *The Art of Software Testing*; John Wiley & Sons, 1979

[Rumb91] Rumbaugh J. u. Andere; *Object - Oriented Modeling and Design*; Prentice - Hall 1991

[Sche89] Scheibl H.-J.; *Kommerzielle Software-Entwicklung;* expert verlag, 1989 Band 151

[Schu88] Schulz A.; *Software-Entwurf, Methoden, Werkzeuge*; Oldenburg, 1988

[Schu86] Schulze H.H.; *Computer Englisch, Fachwörterbuch*; rororo, 1986

[Schu89] Schulze H.H.; *Computer Enzyklopädie*; rororo, 1989

[Shne86] Shneiderman B.; *Designing the User Interface*; Addison-Wesley, 1986

[Shoo83] Shooman M.; *Software-Engineering* ; Addison-Wesley, 1983

[Somm92] Sommerville I.; *Software Engineering*; Addison-Wesley, 4_{th} Edition 1992

[Tuto80] Tutorial; *SW-Cost Estimating and SW-Life-Cycle Control*; IEEE-Sammelband, 1980

[Wein71] Weinberg G.; *The Psychology of Computer Programming*; Van Nostrand Reinhold, 1971

[Will84] Willmer H. / Balzert H.; *Fallstudie einer industriellen Software-Ent.*; B.I., 1984, Reihe Informatik / 39

[Your79] Yourdon E. Constantine L.; *Structured Design:Fundamentals of Discipline ...;* Prentice-Hall, 1979

[Zehn86] Zehnder C.A.; *Informatik-Projektentwicklung*; B.G. Teuber Stuttgart, 1986

8 UNTERSTÜTZUNG DER STUDENTEN

Sie können im Diskussionsforum: „Fragen“ und „Probleme“ in das Forum stellen. Sie sind im Gegenzug dazu aufgerufen, daß Sie ihre Kollegen ebenfalls unterstützen. Das Diskussionsforum wird regelmäßig von mir bearbeitet und „überwacht“. Sollten Probleme auftreten, die nicht in der Gruppe gelöst werden, können per Email an den Vortragenden weitergeleitet werden.