AMEISE Workshop Emden 2007

Einleitung

A. Bollin, R. Mittermeir

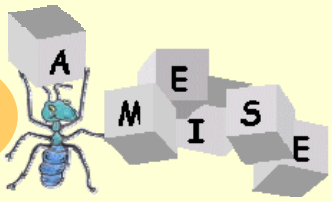
Software Engineering und Soft Computing

Alpen-Adria Universität Klagenfurt

Eckdaten

-
-

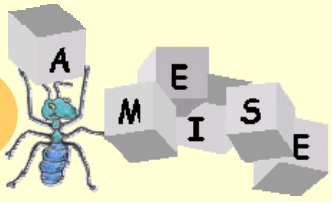
Zeitplan



- 5. Juli 2007 (Do.)
  - 09:00 – 10:45 Motivation/Theorie
  - 11:15 – 12:45 AMEISE Umgebung
  - 14:15 – 15:45 Projektplanung
- 6. Juli 2007 (Fr.)
  - 08:30 – 12:45 AMEISE Simulation I
- 7. Juli 2007 (Sa.)
  - 09:00 – 11:00 Assessment/Nachbesprechung

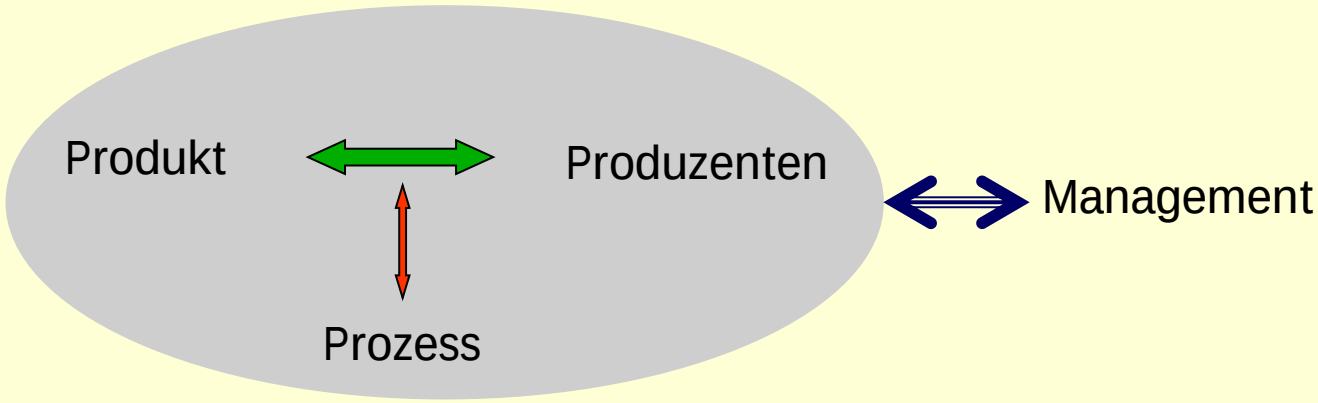
2te AMEISE Simulation (unbetreut)

Inhalt (Teil I)



- Projektmanagement (basierend auf dem Foliensatz der VK „IT Projektmanagement and Change“, © R. Mittermeir, 2006/7)
  - Allgemeine Aspekte
    - Planung
    - Produkt
    - Prozess
    - SW-Projekt
  - Kostenschätzung
    - Problembereiche
    - Kosten
    - Schätzung / Modelle
    - Function Points
  - Daumenregeln

Aspekte (1/3)



Aspekte (2/3)

- Basis für Planung
  - Eckpunkte des Projektmanagements
  - Prozessmodelle
- Projektplanung
  - **Ablaufplanung**
    - Vorgehensmodelle (Wasserfallmodell, Spiralmodell, Rapid Application Development/X-Prog., …)
  - **Aufwandsplanung**
    - Grundfragen der Aufwandsschätzung, Schätzgrößen und Metriken
    - Aufwandsmodelle (Function Point Methode, COCOMO, …)

Planung (1/4)

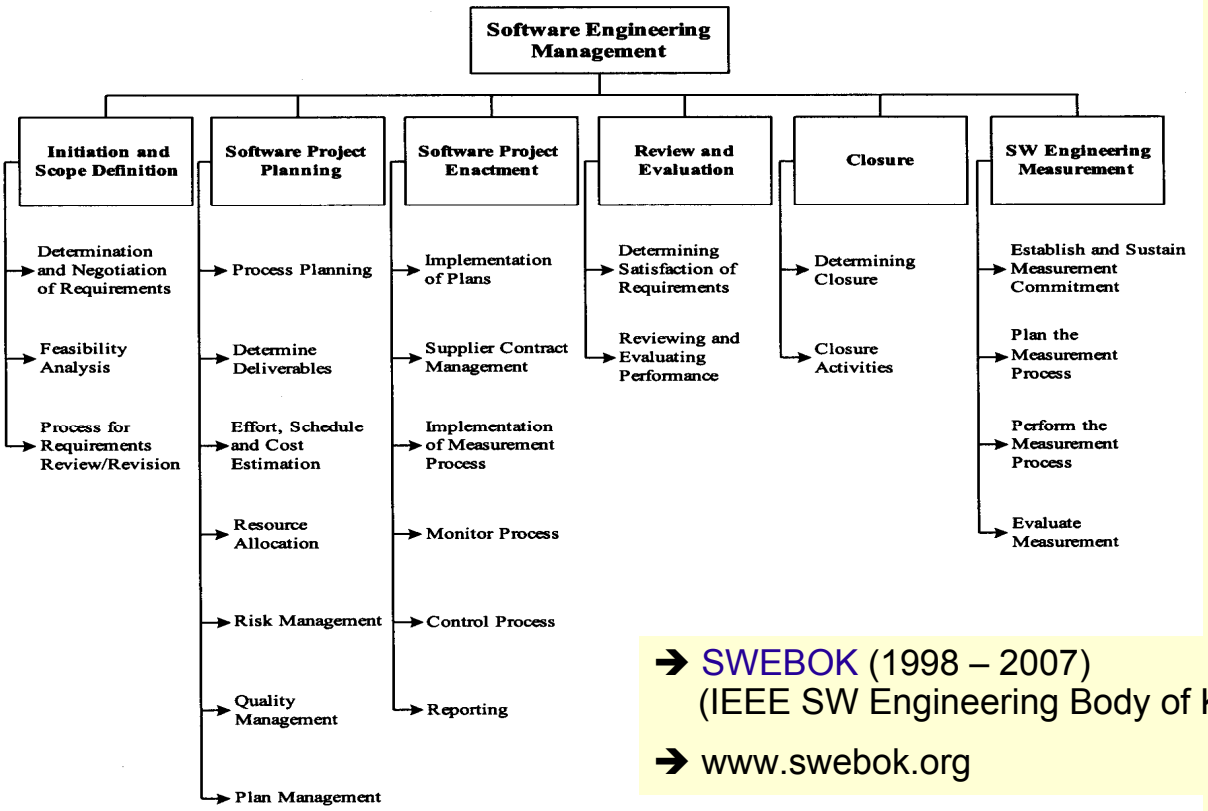
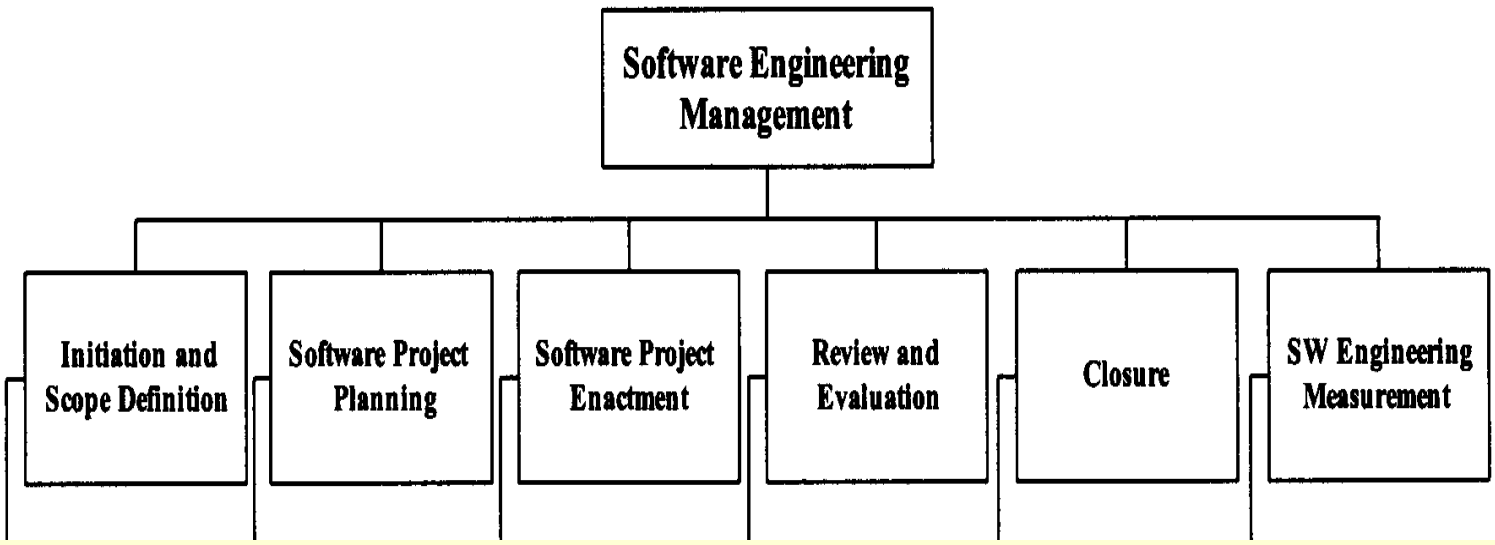


Figure 1 Breakdown of topics for the Software Engineering Management KA

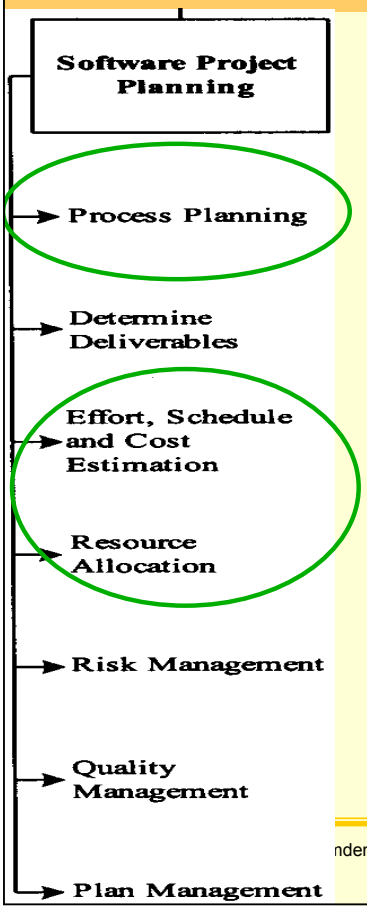
➔ SWEBOK (1998 – 2007)
(IEEE SW Engineering Body of Knowledge)

➔ www.swebok.org

Planung (2/4)

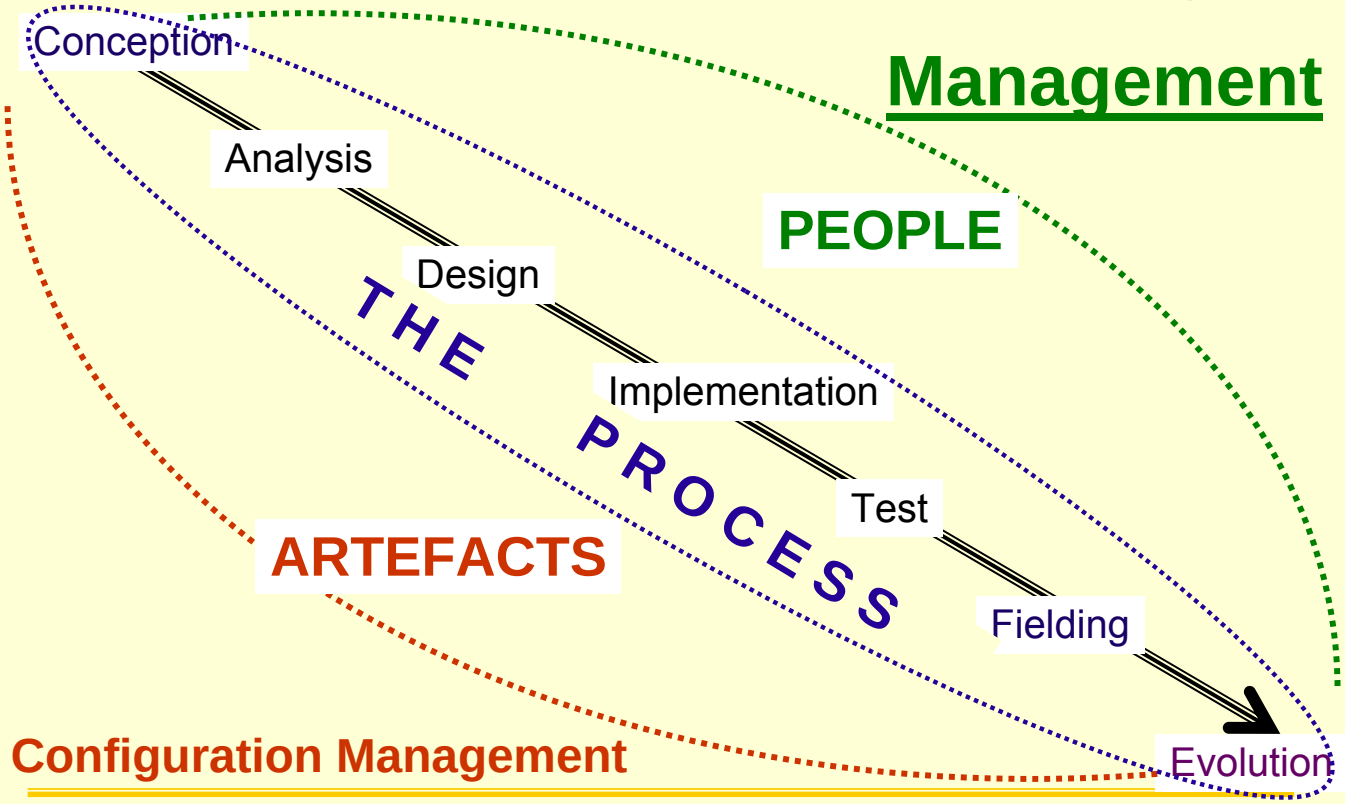


Planung (3/4)

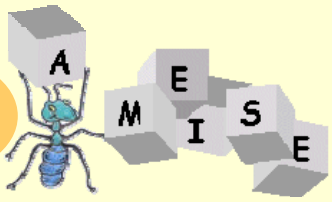


- SWEBOK: "Balanced amalgam of technical issues and peoples related issues"
- Leistungen definieren!!!
  - Arbeiten?
    - WANN, in WELCHER Qualität, von WEM
  - Wer definiert diese?
    - Entwicklungsprozess (ISO 9000, CMM, SPICE, …)
    - Kunde
- Planung, Schätzung, Zuordnung von Betriebsmittel:
  - Ein Netz an komplexen Zusammenhängen!
- Zum Schluss: Entscheidung über das Prozessmodell

Planung (4/5)



Produkt (1/2)



Auf dem Weg zum Software Produkt: Entwickler

spezifische Basisleistung von Entwickler(n)

- andere Aufgaben / Tätigkeiten

sachbezogene Basisleistung

- Fehler

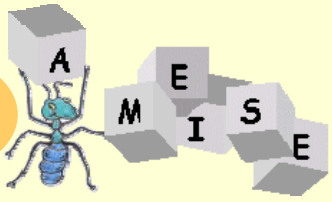
sachbezogene Nettoleistung

- Fehlerkorrektur

==========

sachbezogene Effektivleistung

Produkt (2/2)



- Lieferbare Ergebnisse: („deliverable“)

Code, in Qualität ... ?

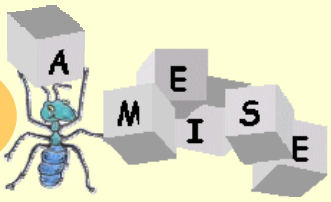
Qualitätsattribute

% vollständig
# Fehler / KLOC

Dokumentation

% vollständig
# Fehler / KLOC

Ablaufplanung (1/4)

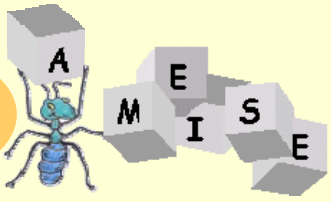


- Prozess-Grobstruktur:

  WAS
  wird WANN
  von WEM
  gemacht

- U-level Modelle [Humphrey 89]:
  - Lineare Phasenmodelle (Wasserfall)
  - V-model
  - RUP
  - Incrementelle Modelle
  - …

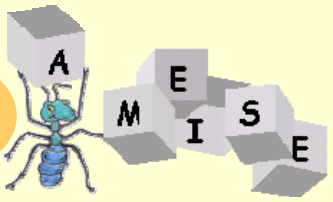
Ablaufplanung (2/4)



- Welche Kriterien bestimmen den Phasenübergang?
  - Welche „deliverables“ …
  - … in welchem Zustand?
    - Was muss vorhanden sein?
    - In welchem Zustand muss es vorliegen?
      - wie „fertig“? ↔ Def. v. „fertig“?
      - wie „richtig“ ↔ Verfahren f. Def v. „richtig“?
        . wie geprüft?

- Aus welchen Schritten setzt sich die Phase zusammen?

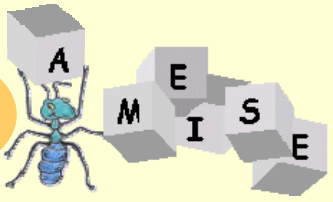
Ablaufplanung (3/4)



Kapazitäts-planung ⟷ Aufwands-schätzung

↳ „Arbeitsvorbereitung“

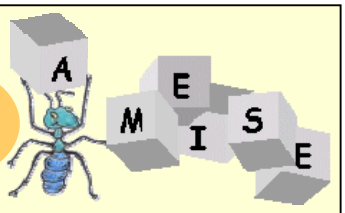
Ablaufplanung (4/4)



- **Gliederung des Projekts in eine Hierarchie von Arbeitspaketen**
  - Work Breakdown Structure
    - Funktions- bzw. Aufgabengliederung als bestimmendes Element
  - Product Breakdown Structure
    - Produktgliederung als bestimmendes Element
    - PRINCE
- Schlüsselfrage: Abhängigkeiten
- Methoden:
  - Balken-Diagramme: zeitlicher Verlauf, ohne Abhängigkeitsdarstellung
  - CPM (like) Methoden: Aktivitäten als Zustandsübergänge
    PERT (like) Methoden: Teilaufgaben als zeitlich verschiebbare Knoten

Fragen
?

Kostenschätzung



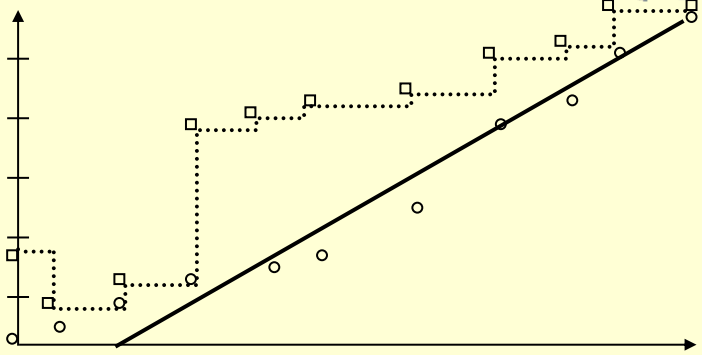
- Was kostet „das Ganze“?
  - zuerst: Was ist „das Ganze“?
- Wieviel kostet „ein bisschen Mehr“?
  - warum?
- Wann stellen wir diese Frage?
- Wozu stellen wir diese Frage?
- Wie oft stellen wir diese Frage?

Problembereiche (1/2)

- Warum unterschätzen wir Software-Projekte (und was können wir dagegen tun)
  - Schätzen oder Raten
  - Schätzgrößen und Metriken
  - Aufwandsmodelle
    * Basisüberlegungen
    * Mikro-Modelle
    * Makro-Modelle

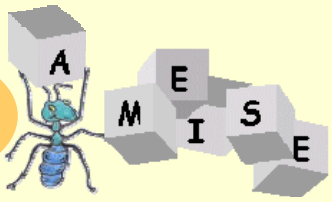
Problembereiche (2/2)

- ▫ geschätzte Kosten
- ○ tatsächliche Kosten



Probleme für
- Kunde:
  * Umstellungsplanung
  * Kauf / Verkauf von Geräten
  * Abschluss / Ablauf von Verträgen
  * ...
- Entwickler:
  * Überstunden
  * Ausbrennen
  * ...
- Finanzierung
- andere Verträge => Domino-Effekt

Kostenschätzung (1/3)



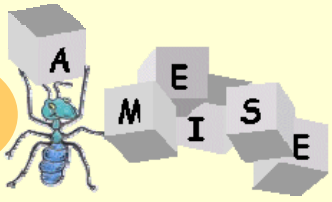
- Warum unterschätzen wir (SW-) Entwicklungsaufwand?

Wie lange benötigen Sie, um

- mit der Bahn nach Hamburg zu fahren .....
- mit dem Auto nach Hamburg zu fahren .....
- zu Fuß zum Hauptbahnhof zu gehen .....
- eine Seminararbeit in ____ zu verfassen .....
  ....
  ....
  ....
  ....
- sich für den AMEISE Workshop vorzubereiten .....
- ein Programm zur Verwaltung Ihres Terminkalenders zu entwickeln

  .....

Kostenschätzung (2/3)



- Schätzung des Inhalts/Umfangs der Aufgabe
- Schätzung des Schwierigkeitsgrades der Aufgabe

} Interaktion!!!

- Schätzung unserer Problemlösungsfähigkeit
- Schätzung unserer Verfügbarkeit

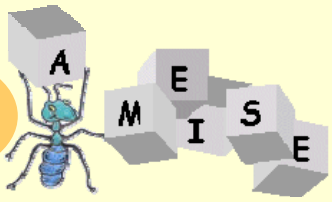
- Berücksichtigung des eingesetzten "Materials"
- Berücksichtigung der eingesetzten Instrumente / Werkzeuge
- ...
- Berücksichtigung von Randbedingungen der Entwicklung

(z.B. Anforderungsvolatilität, turnover, ..., Jahreszeit)

Pseudo-Paradoxon:

Aktivität	Assembler-Prog.	3-GL-Programm in Wochen
Entwurf	4	4
Codierung	4	2
Test	4	2
Dokumentierung	2	2
Management / Support	2	2
Aufwand	16 (4M)	12 (3M)
Lines of Code	2000	500
LOC p. PM´	**500**	**167**

Kostenschätzung (3/3)



- Wie viel Tage hat ein Monat/ein Jahr?

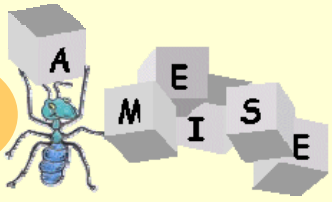
  Das Jahr gliedert sich in:
  - Feiertage,
  - Urlaub
  - Krankheit
  - dringende Aushilfstätigkeiten
  - Schulungen
  - ...
  - **Arbeit am Projekt**

  Die Arbeit (von Programmierern) am Projekt gliedert sich in:

  [Bell Lab time and motion study]

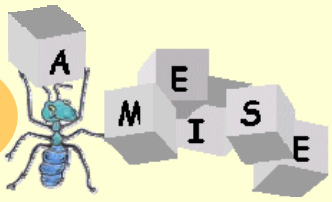
32 %	job communication
16 %	read programs, manuals, ...
13 %	write programs
6 %	training
5 %	mail, misc. documentation
15 %	misc. (walking offsite, ...)
13 %	personal

Schätzgrößen und Metriken (1/4)



- Wieviel kostet die Projektdurchführung? <S, e, $, ...>
  - von welcher bis zu welcher Phase?
  - wessen Kosten?
- Wie lange benötigen wir für das Projekt? <Tage, Wochen, Jahre>
- Wieviel Aufwand ist mit dem Projekt verbunden? <Mann-/Personal--Tage,-Monate>
- Wieviel Mitarbeiter benötigt / verträgt das Projekt (in Phase ___)? <# Personen>
- Wie groß wird das Programm? <# Dateien, # Funktionen> <Lines of Code>
- Wie wirkt sich "_diese_" Änderung auf den Gesamtaufwand aus?
- *"I have one lamp by which my feet are guided and that is the lamp of experience; I know of no way of judging the future but by the past."* *[Patrick Henry]*

Schätzgrößen und Metriken (2/4)



direkte Zeitschätzung:

Bei Aufgaben, die unsere geistigen Fähigkeiten involvieren zu gefährlich!

=> Ziel: **Vermeiden von emotionsgeladenen Primär-Schätzgrößen**

Lines of Code (LOC, KLOC; KDSI)

Vorsicht: Exakte Definition konstant halten ? written / delivered ?

* Programmiersprache (insb. Sprachgeneration)
* Zählen von Kommentaren
* Zählen von Deklarationen
* Zählen von Includes
* Zählen von Reused Components
* Beurteilung von Vererbung

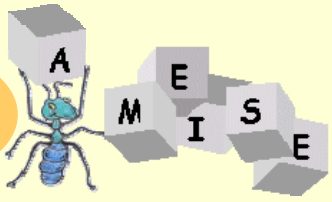
Personenmonate, Mannmonate (PM, MM)

(ManMonth, ProgrammerMonth, SoftwarePersonellMonth, ..)

Monat

Vorsicht: Exakte Definition konstant halten;
Definitionen sind länder- und kulturspezifisch!

Schätzgrößen und Metriken (3/4)

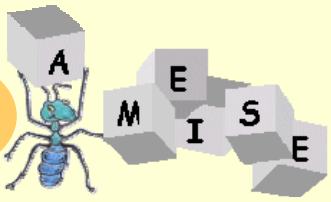


Methoden der Praxis:

- Top-Down Schätzung — "raten" auf der Basis globaler Erfahrungen
- Verhältnisschätzung — VORSICHT: diseconomies of scale
- Gleichheit / Unterschiedsschätzung — Basis: Zerlegung und Klassifikation
- Standard-Werte — "Preisliste" vgl. mit klassischem Ingenierswesen
- Was der Kunde verträgt — *no comment !*
- Damit wir den Auftrag noch „ergattern" — *no comment !*
- Schätzung + Überziehung (z.B.: Schätzung * 2) — *no comment !*
- Schätzung - Nachforderung — *no comment !*
- Bottom-Up Schätzung
- modellbasierte Schätzungen

VORSICHT:
Schätzung von Wartungsaufwand =|= Schätzung für Neuentwicklung

Schätzgrößen und Metriken (4/4)



Grundannahme: Man verschätzt sich, aber wenn man das Problem konsequent durchdenkt, schätzt man besser

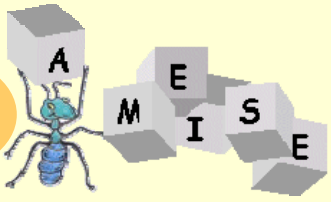
3 Schätzwerte:

* Untergrenze u
* Modus m
* Obergrenze o

Erwartungswert$_{KDSI}$: μ_M := (u + 4m + o) / 6

Streuung$_{KDSI}$: σ_M := (| o – u |) / 6

Aufwandsmodelle



- Mikro-Modelle
  - Halstead Methode
  - McCabes Komplexitätsmodell
  - Objektorientierte Mikro-Modelle
  - ...
- Makro-Modelle
  - Regression aus Analyseparametern
  - Function Point Methode
  - Putnam's Rayleigh Curve Model
  - COCOMO
  - ...

Function Points (1/6)

Function Points Methode

Heute gängigstes Verfahren zur Aufwandsschätzung in frühen Projektphasen

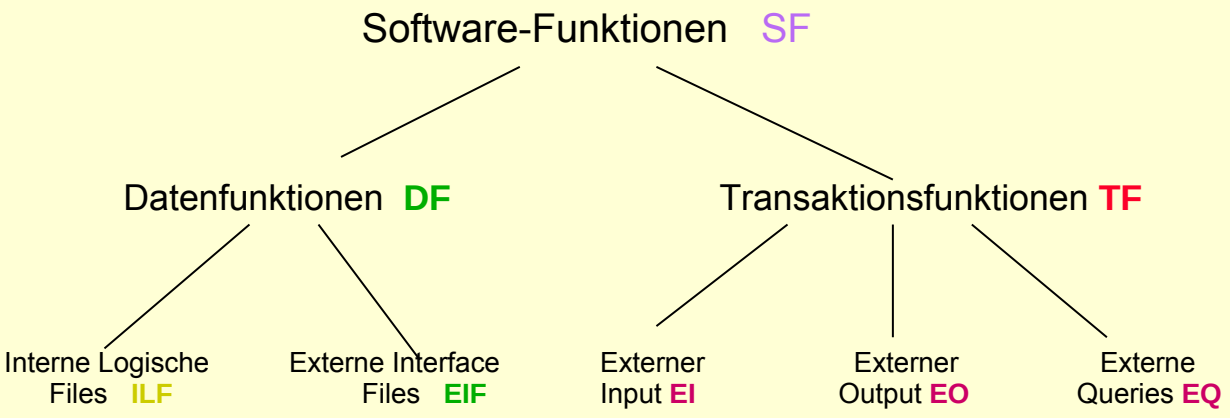
* Entwickelt bei IBM (1974 - 1979) [Albrecht, 79; Albrecht, 83]
* Weiterentwicklung und Standardisierung durch die IFPUG (Internat. Function Point User Group)
* Anwendungsbereich: klassische Datenverarbeitungsprojekte; übertragbar auf moderne Entwicklungen
* Benutzersicht als Ausgangsbasis (Analysemodell)
* Function Point ::= abstraktes Maß für die Funktionalität des Systems
* Einfach handhabbar, vielerorts beschrieben [Putnam 92, Garmus 96, ...]
* methoden- und technologie-unabhängig
* Für technische Anwendungen (Numerik, Algorithmik, Real-Time, ...) nicht geeignet (zu einseitig auf Daten bzw. I/O fokussiert)

Function Points (2/6)

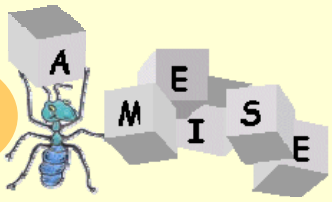
Ausgangsbasis für die Schätzung:

- „sichtbare“ Funktionalität des Systems
- Dateibehandlung (verallgemeinerbar auf: Schnittstellenbehandlung)

Aufbau:



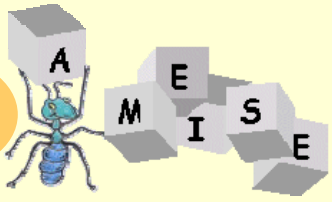
Function Points (3/6)



Ausgangsparameter

* # int. logische Files: Wesentliche interne Datenstrukturen.
* # ext. Interfaces: Typen unterschiedlicher Schnittstellen zu anderen Systemen.
* # ext. Inputs: Typen unterschiedlicher Benutzerdaten /-steuerungen die ins System gehen.
* # ext. Outputs: Typen unterschiedlicher Benutzerdaten /-steuerungen die das System verlassen.
* # Abfragen: Typen unterschiedlicher Benutzereingaben die eine unmittelbare Antwort verlassen.

Function Points (4/6)



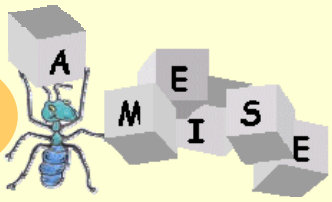
Vorgangsweise:

1. Zählart bestimmen
2. Anwendungsgrenze bestimmen
   Systemgrenze aus Sicht der/des Nutzers; implementierungsunabhängig
   ??use cases??
3. Datenfunktionen und deren Komplexität bestimmen
4. Transaktionsfunktionen und deren Komplexität bestimmen
5. Rohbewertung ermitteln (UFPs)
6. Wertanpassungsfaktor (WAF) bestimmen (AFPs)
7. Endbewertung, Umrechnungen

Zählarten:

- Entwicklungsprojekte
- Anpassungsprojekte
- Anwendungen eines großen Systems

Function Points (5/6)

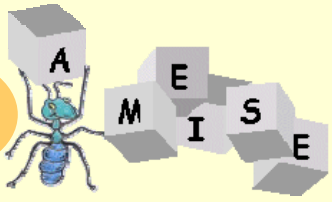


- Systemcharakteristika
  14 Anwendungs-Charakteristiken (siehe [Albrecht 83])
- Transformation in SLOC (programmiersprachenabhängig !)
  zB: (siehe [Capers 95])

Assembler	320	APL	32
C	150	SmallTalk	21
Pascal	91	Spreadsheet	6
Java	53	Ada	49
SQL	12		

- Transformation in Personalmonate (ebenfalls programmiersprachen- und umgebungsabhängig)

Function Points (6/6)

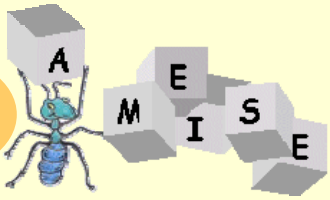


Zusammenfassung:

- Schätze Systemgröße (UFP)
- Korrigiere um techn. Komplexität (AFP)
- Berechne Zeitdauer und Aufwand mit sprachspezifischen Tabellen
- Verteile Dauer, Aufwand (Effort) über Phasen
- Adjustiere um Risikofaktoren
- Berücksichtige Personalverfügbarkeit

Fragen
?

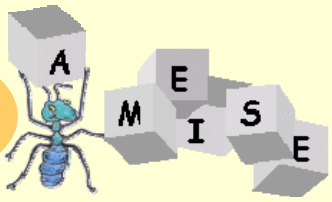
Daumenregeln (1/4)



VORSICHT: Diese Daumenregeln sind
keine Berechnungs- oder Planungsrichtlinien.

Sie können aber zur ersten Orientierung, insbesondere zur Plausibilitätsprüfung von Werten,
die von Planungswerkzeugen berechnet werden,
verwendet werden.

Daumenregeln (2/4)



D1: 1 UFP ~ 0,1 KDSI ~ 100 LOC

Range: (300 – 20)				
	ADA 83	71	ADA 95	49
	C	128	C++	55
	FORTRAN 77	107	FORTRAN 95	71
	UNIX Shell Script	107	Java	53

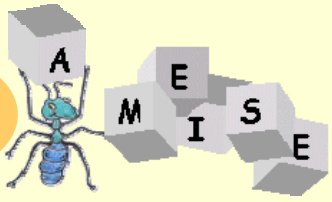
D2: $(\#\ UFP)^{1,15} \sim \#S$ Dokumentation

sehr kundenspezifisch – kann teuer kommen!

D3: „Appetit“: UFP(m+1) ~ UFP(m)*1,01

Umso länger die Entwicklung dauert, umso mehr zusätzliche Anforderungen fallen dem Kunden ein.

Daumenregeln (3/4)



D4: $\#\ Testfälle \sim (\#\ UFP)^{1,2}$

Nimmt man an, dass jeder Testfall n-mal (z.B. 4 mal) ausgeführt wird, lässt sich daraus die Testdauer schätzen.

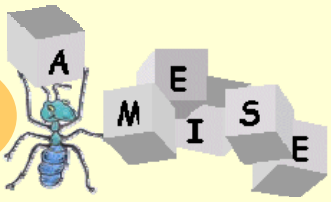
D5: $(\#\ UFP)^{1,25} \sim \#$vermutbarer Fehler

Bugs in „mayor deliverables“ (Requ, Design, Code, User-Docu).
Bei Wartungsprogrammierung ist‘s mehr: ~ 1,27

D6: Review/Inspektion findet 30 % d. Fehler

Dies begründet den Unterschied zwischen Fehler-Entdeckungsraten von 85 % (Industriestandard) und 99 % (Spitzen)

Daumenregeln (4/4)



D7: (# UFP)^{0,4} ~ #m Entwicklungsdauer

Dies ist hier die Zeitspanne zwischen Beginn der Anforderungsanalyse und erster Auslieferung (non-real-time SW).

D8: (# UFP)/150 ~ #Personal(Entwicklung)

Vorsicht: „Rechtecks-Staffing“ ist ineffizient

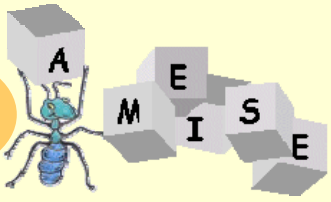
D9: (# UFP)/500 ~ #Personal(Wartung)

Vorsicht: „Rechtecks-Staffing“ ist ineffizient

D10: 2,4 x (KLOC)$^{1.05}$ ~ #MM

COCOMO (Basic)

Referenzen



[Albrecht 79] Albrecht A.J.: “Measuring Application development” Proc. of IBM Applications Development Joint SHARE/GUIDE Symposium, Moterey, CA, pp. 83 - 92, 1979

[Albrecht 83] Albrecht A.J., Gaffney J.: “Software function, source lines of code and development effort prediction” IEEE Trans. on Software Engineering, SE-9(6), pp. 639 - 648, 1983.

[Basili 80] Basili V.R.: Tutorial on Models and Metrics for Software Management and Engineering, IEEE CS Press, 1980

[Boehm 81] Boehm B.: Software Engineering Economics; Prentice-Hall, 1981

[Boehm 00] Boehm B. W. et al.: Software Cost Extimation with COCOMO II, Prentice Hall, 2000.

[Briand 01] Briand L.C., Wüst J: Modeling Development Effort in Object-Oriented Systems Usin g Design Properties; IEEE Trans on Software Eng. Vol. 27 (11), Nov. 2001, pp. 963-986.

[CK 94] Chidamber S.R-., Kemerer C.F.: A Metrics Suite fopr Object Oriented Design, IEEE Trans. on Software Eng., 20 (6), June 1994, pp. 476-493,

[Conte 86] Conte S., Dunsmore H., Shen V.: "Software Engineering Metrics and Models"; Benjamin/Cummings, 1986

[Costagliola 05] Costagliola G., Ferrucci F., Tortora G, Vitiello G.: Class Point: An Approach for the Size Estimation of Object-Orfiented Systems; IEEE Trans. on Software Eng., 31 (1), Jan 2005, pp 52 – 74.

[Fenton 97] Fenton N.E., Pfleeger S.L.: "Software Metrics: A Rigorous & Practical Approach"; 2nd ed, Thomson, 1997

[Garmus 96] Garmus D., Herron D.: “Measuring the Software Process - a practical guide to functional measurements”; Yourdon Press, Prentice Hall, 1996.

[Halstead 77] Halstead M.H.: "Elements of Software Science"; North Holland, 1977

[Jones 86] Jones Capers: “Programming Productivity”; McGraw-Hill, 1986

[McCabe 76] McCate T.: “A software complexity measure” IEEE Trans. on Software Engineering, SE-2(4), pp. 308-320, 1976.

[Parikz 84] Parikh G.: "Programmer Productivity"; Reston, 1984

[Putnam 78] Putnam L.: “A General Empirical Solution to the Macro Software Sizing and Estimating Problem”; IEEE Trans. On Software Engineering SE-4(4), pp. 345 - 361, 1978

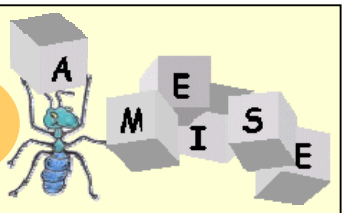
[Putnam 80] Putnam L.: “Tutorial: Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers”; IEEE-CS Press, 1980

[Putnam 92] Putnam L., Myers W.: "Measures for Excellence: Reliable Software on Time, within Budget"; Yourdon, 1992

[Wolverton 84] Wolverton R.W. "Software Costing" in Vick C.R., Ramamoorthy C.V.: "Handbook of Software Engineering"; van Nostrand Reinhold, 1984

Fragen
?

Inhalt (Teil II)



- AMEISE Simulation (basierend auf dem Foliensatz der VK „IT Projektmanagement and Change“, A. Bollin, 2006)
  - Ameise
    - Projektbeschreibung
    - Architektur
  - Simulationsmodelle
    - Qualitätssicherung
    - 200 AFP Projekt
  - Aufgabenstellung
  - Simulationshinweise
    - Besonderheiten
    - Start und Ende

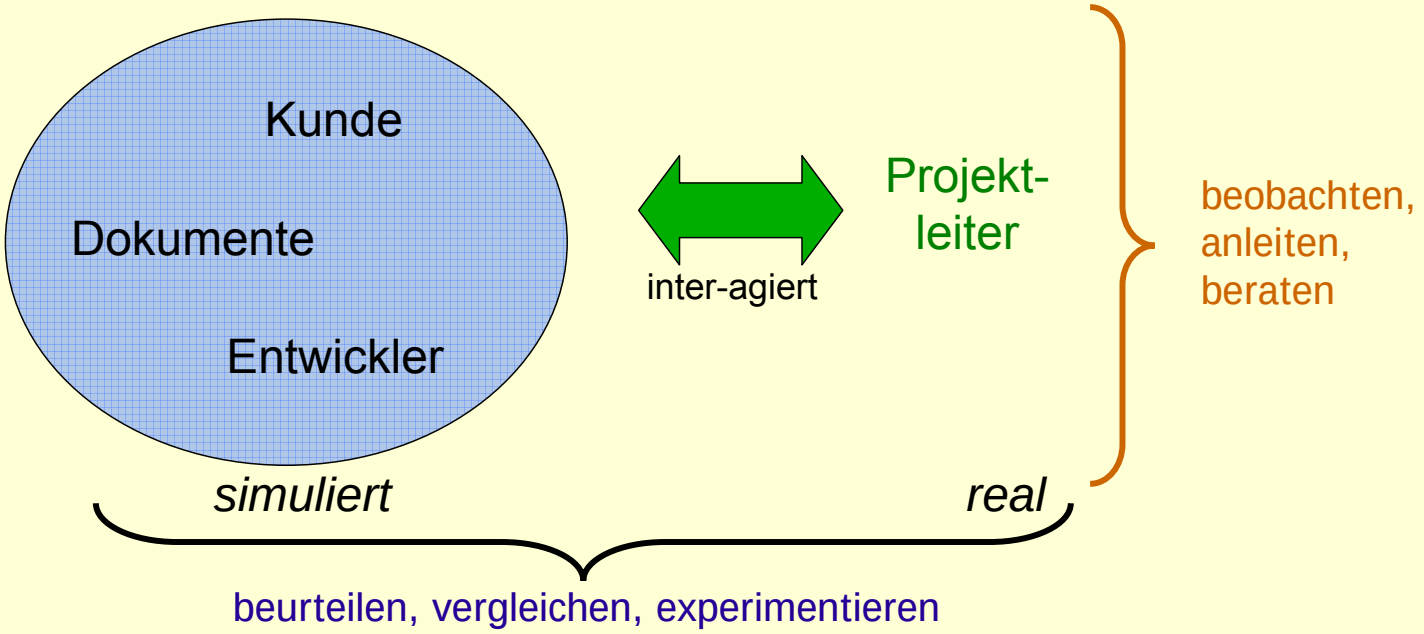
Projektbeschreibung

- **A** **M**edia **E**ducation **I**nitiative for **S**oftware **E**ngineering
- Motivation
  - Theoretisches Wissen über Projektmanagement erfahrbar machen, Erfahrungen für die Praxis sammeln
  - Prototyp „SESAM“ der Uni Stuttgart erlaubt experimentelles Lernen in Simulationsumgebung
- In interuniversitärer Zusammenarbeit (Stuttgart, Klagenfurt, Linz, Technikum) wurden folgende Punkte im Rahmen des Ameise-Projekts realisiert:

➔ ~150.000 LOC (Ada, Java, C)

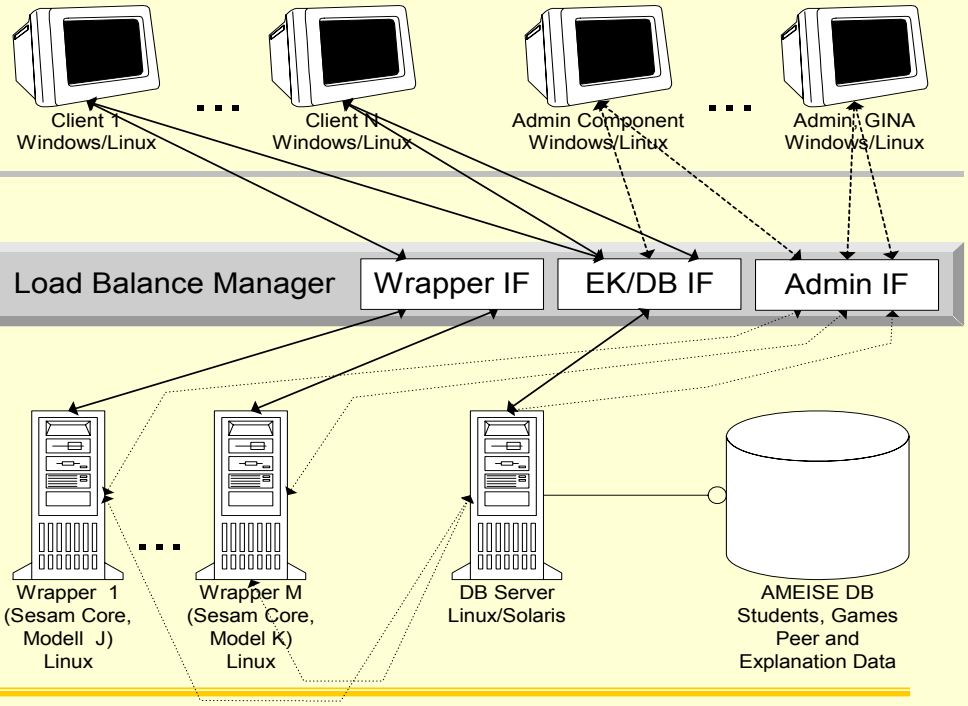
➔ 40 Mitarbeiter in KLU, 10 Jahre in Stuttgart, 6 Jahre in Klagenfurt

Architektur (1/4)



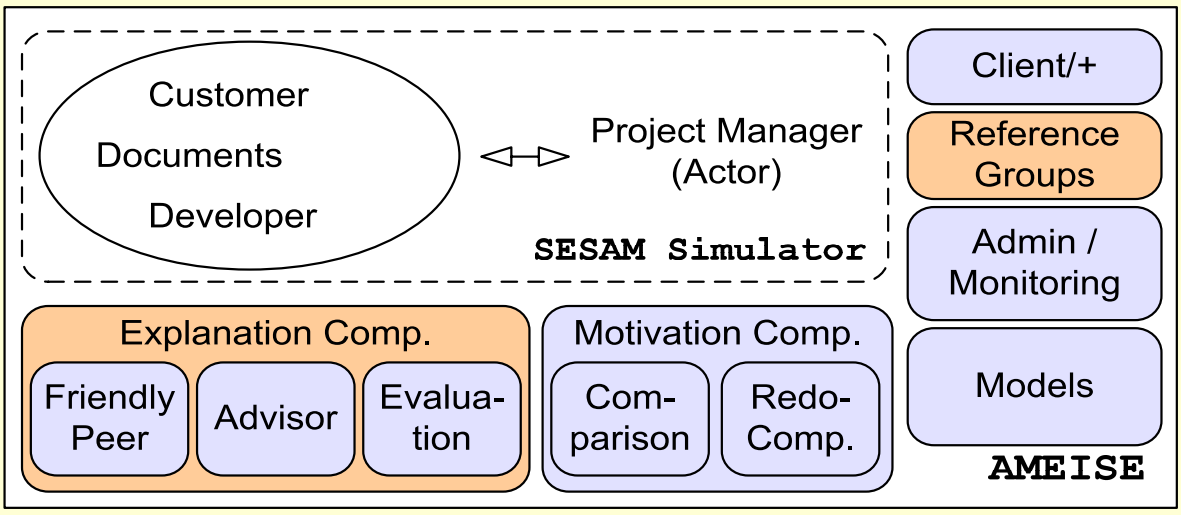
Architektur (2/4)

Architektur

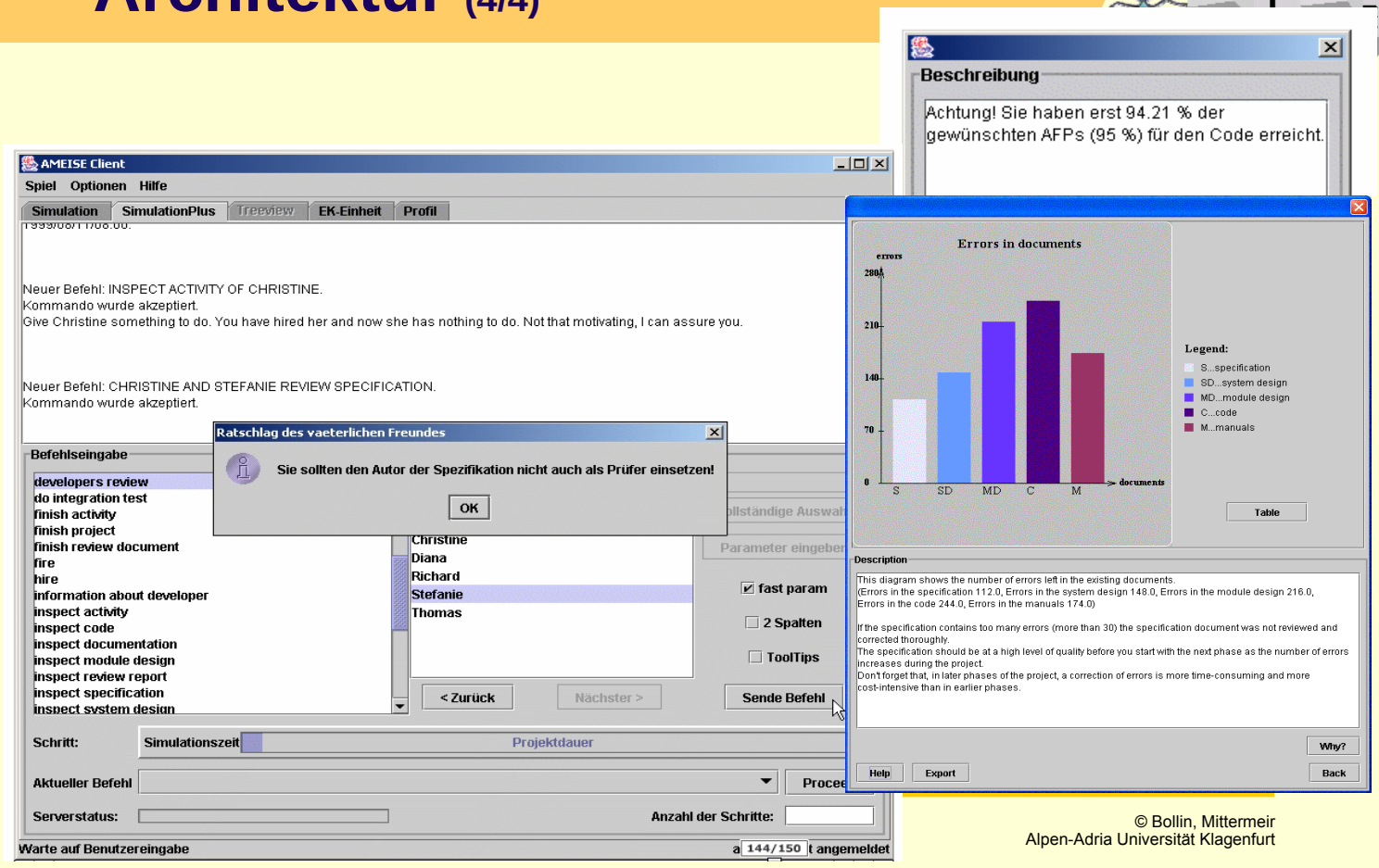


Architektur (3/4)

- Komponenten



Architektur (4/4)

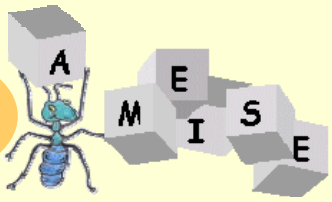


Modelle (1/2)

- SESAM ist ein Simulationssystem, das ein spezifisches Modell interpretiert
- Objekte und Beziehungen aus realen Projekten sind auf das Projektmodell von SESAM abgebildet
- Abbildung durch die Modellbeschreibungssprache (SESAM-2)

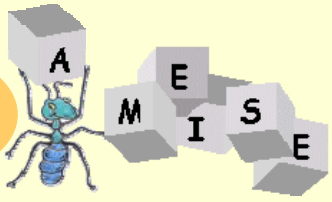
- Unser Modell: QS/QA-Modell
  - Software-Entwicklung mit spezifischem Fokus auf Qualitätssicherung (Quality Assurance)
  - Das QS-Modell konzentriert sich vor allem auf
    * konstruktive Maßnahmen
    * analytische Maßnahmen
    * korrigierende Maßnahmen

Modelle (2/2)



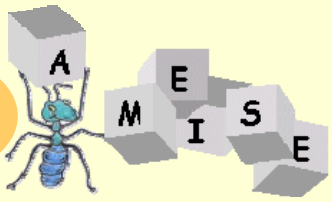
- Ziel: Nutzen verschiedener Maßnehmen erfahren und lernen, diese sinnvoll einzusetzen
- Mit Hilfe des QS- Modells sollen folgende Projektmanagement-Funktionen eingeübt werden
  - Projektplanung
  - Stellenbesetzung (d.h. richtige Mitarbeiter auswählen und entsprechend ihrer Qualifikation einsetzen)
  - Projektführung
- QS-Modell bildet kleine bis mittlere Auftragsprojekte mit einem Umfang von 200 (bis 1200) AFPs ab
- QS-Modell unterstützt aktivitäten-orientierte Prozessmodelle
  - positiv: Vorgehen nach dem Wasserfallmodell
  - negativ: Code-and-Fix

Aufgabenstellung (1/4)



- Ihre Aufgabe: Ein 200 AFP-Projekt soll
  - mit Auswahl aus verfügbarem Personal,
  - innerhalb der vereinbarten Bedingungen
    * Code
      - 95 % Fertigstellung,
      - < 12 Fehler pro 1000 LOC
    * Handbuch
      - 95 % vollständig
      - max. 1 Fehler auf jeder zweiten Seite
    * Dauer
      - Beginn 1.10.2007, Ende spätestens 5.7.2007
  - innerhalb des vorgegebenen Budgets (225.000 €) fertig gestellt werden

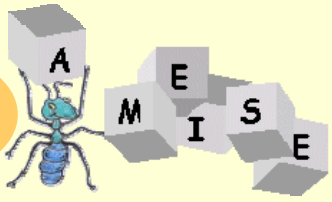
Aufgabenstellung (2/4)



„Spielregeln:“

- Jedes Zwischenprodukt will/kann geprüft werden
- Prüfergebnisse erfordern Aufmerksamkeit
- Aspekte:
  - Niemand ist sein eigener Prüfer
- Problem:
  - Reviews sind keine Vollzeit-Beschäftigung
  - Modell kennt keine Teilzeit-Arbeitskräfte

Aufgabenstellung (3/4)

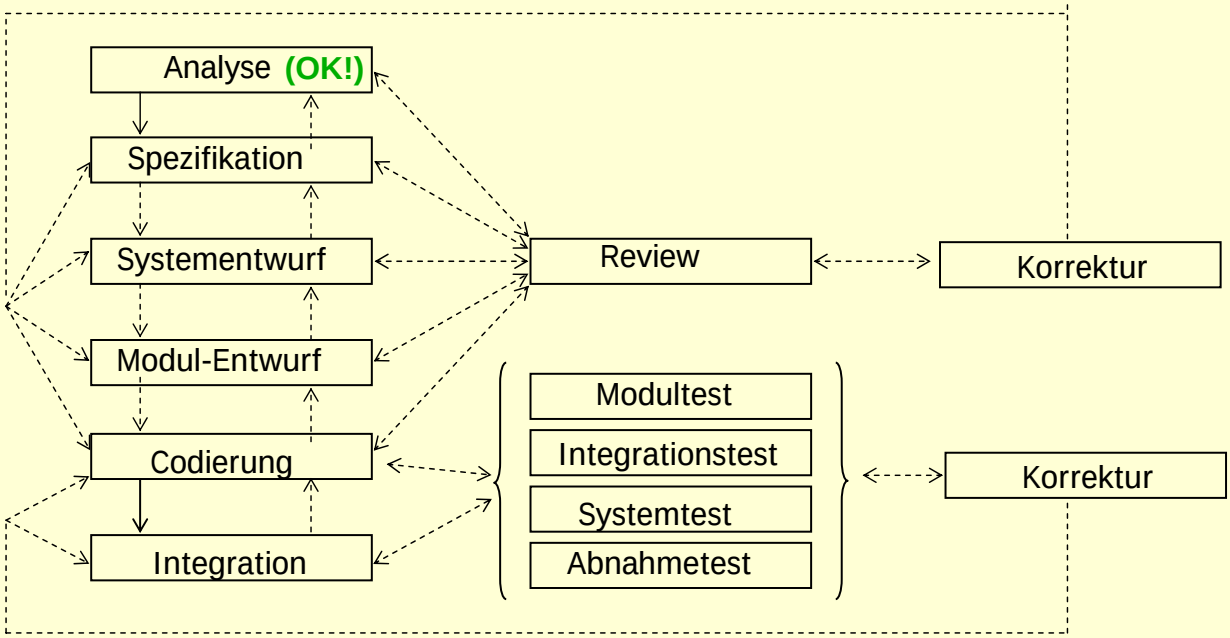


Entwicklungsprozess:

➔ PhasenModell / Wasserfall

- Analyse (Analysis) – bereits erledigt !!!!
- Spezifikation (Specification)
- System-Design (Systementwurf/Grobdesign)
- Modul-Design (Modulentwurf/Feindesign)
- Codierung (Coding)
- Modul-Test (Module test)
- Integration (Integration test)
- Systemtest (System test)
- Abnahme-Test (Acceptance Test)

Aufgabenstellung (4/4)



Lernziele

- Sie sollen gegenüber dem Projekt dieselbe Rolle einnehmen wie ein realer Projektleiter
  - selbständiges Planen und Durchführen von Software-Projekten
  - trainieren wichtiger Projektmanagement-Funktionen
- Erlernen von Entscheidungsfähigkeit unter unvollständiger Information
  - Projektleiter müssen ihre Entscheidungen in der Regel bei nur geringer Information über das Projekt treffen
  - Studenten sollen schwierige Situationen erfahren und lernen diese zu meistern
- Entscheidungsfindung in komplexen Entscheidungssituationen
  - typische Schwierigkeiten bei der Planung und Kontrolle von Projekten erleben

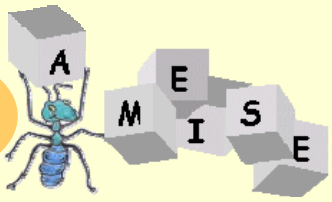
Fragen
?

AMEISE Workshop Emden 2007

Simulationshinweise

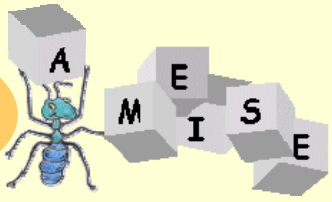
QS200 Modell (200 AFP)
QS60 Modell (60 AFP)

Hinweise (1/11)



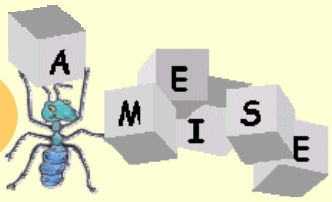
- Personaleinsatz
  - Vorsicht: Mitarbeiter kosten Geld …
    - auch wenn sie nicht beschäftigt sind!
    - Man kann sie in den Pool der Firma entlassen,
      - und sie sind deswegen nicht böse, aber
      - finden vielleicht eine andere Beschäftigung (1-60 Tage)
  - Voraussetzungen
    - Anweisungen können natürlich nur ans eigene Personal gegeben werden.
    - Bevor man jemanden entlässt, muss die Arbeit abgeschlossen oder mindestens beendet werden.

Hinweise (2/11)



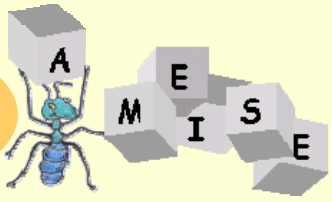
- Dokumente
  - Qualifizierte Mitarbeiter leisten bessere Arbeit.
  - Reviews benötigen Zeit (1/2 Tag). Man kann sie neben der “Hauptaufgabe” durchführen, aber auch die Koordination mehrer Aufgaben frisst Zeit.
  - Mehrere Personen, die an einer Aufgabe arbeiten, müssen sich abstimmen. Die Kommunikation kostet natürlich Zeit.

Hinweise (3/11)



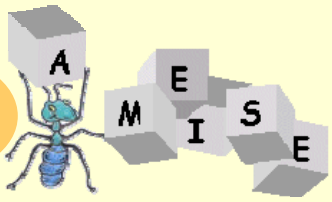
- Reviews
  - Ein Autor eines Dokuments findet nicht nur selbst keine Fehler mehr. Er/sie verhindert auch, dass andere Fehler finden.
    * Sorry, negative Grundannahme in SESAM.
  - Allerdings ist er/sie bei der Fehlerkorrektur effizienter, als jemand der sich erst einarbeiten muss.
    * Vertrautheit mit dem Dokument

Hinweise (4/11)



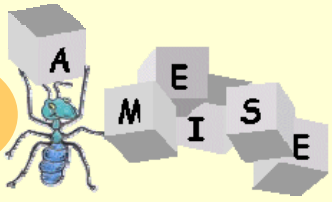
- Parallel-Arbeit
  - Das QA Model fußt auf dem Wasserfall-Modell.
  - Eine Folgephase kann vor Abschluss des aktuellen Meilensteins begonnen werden, aber …
    * mindestens 50 % des Referenzdokuments müssen bereits vorhanden sein,
    * die Unvollständigkeit wird zu Fehlern führen, die zur gegebenen Zeit korrigiert werden sollten ("correct all documents")

Hinweise (5/11)



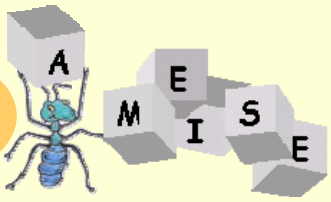
- Partizipative Entwicklung
  (insbesondere Spezifikation, Dokumentation)
  - Ein Kunde entdeckt Fehler, die andere nicht finden können.
  - Früh erkannte fehlende Funktionalität kann relativ günstig eingebaut werden.

Hinweise (6/11)



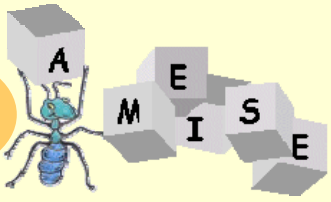
- Eigenheiten des aktuellen Modells
  - Datum
    * Der Tag wird angezeigt, Stunden, Minuten sind irrelevant.
    * Datumsformat: Jahr/Monat/Tag/Stunde:Minute
    * Das Modell berücksichtigt Wochenenden, aber keine Feiertage (z.B. Weihnachten, Neujahr, …)
  - Review
    * Wenn ein Autor Reviewer ist, werden keine Fehler gefunden.
  - Fehler
    * korrigieren sich nicht von selbst!

Hinweise (7/11)



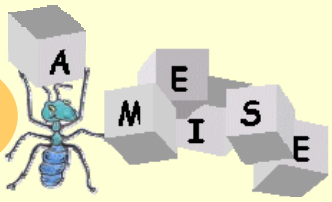
- Zuletzt angezeigtes Datum ist das Datum des aktuellen Tages
- Alle eingegebenen Aktionen finden an diesem Tag statt
  - „Proceed“ -> Simulation schreitet um einen Tag voran
  - „Proceed n“ -> Simulation schreitet um n Tage voran
- Wochenenden werden automatisch übersprungen
- Feiertage/Urlaubstage/Krankentage werden nicht berücksichtigt
- Uhrzeit ändert sich 2mal im Jahr (Sommerzeit/Winterzeit – Ende März/Oktober)
- Zeit hat im Modell (quasi) keine Bedeutung

Hinweise (8/11)



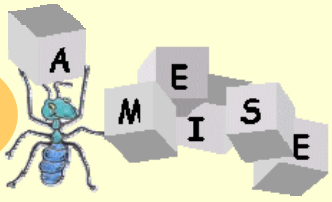
- Review:
  - Anforderungen, die einer der Gutachter als Autor erstellt hat, werden von keinem der Gutachter geprüft
- Tätigkeiten, die einem Entwickler zugewiesen werden, der nicht eingestellt ist, werden nicht bearbeitet (→ Meldung)
- alle erfolgreichen Kommandos werden sofort oder nach spätestens einem „Proceed“ mit einer Nachricht bestätigt
  - sonst gilt Kommando als nicht ausgeführt

Hinweise (9/11)



- Modell lässt Aktion nicht zu? → Situtationsabhängig
  - „finish activity“ (-> keine Rückmeldung, wenn entsprechender Entwickler keine Aufgabe hat)
  - „show available developers“ (-> keine Rückmeldung, wenn es keine freien Entwickler gibt)
  - „show hired employees“ (-> keine Rückmeldung, wenn keine Entwickler eingestellt sind)
  - nach „finish project“ und „deliver system“ müssen mehrere Proceeds ausgeführt werden. Auswertung des Spielverlaufs erfolgt erst nach drei Tagen -> Aufräumarbeiten
  - „integrate“ (verlangt mindestens 50 % der Anforderungen im Code umgesetzt wurden – sonst wird Kommando nicht ausgeführt)

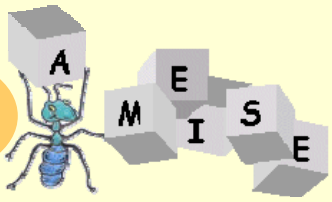
Hinweise (10/11)



- Spieleraktion bleibt ohne Wirkung?

  Folgende Fälle können auftreten:
  - führt Entwickler bestimmte Tätigkeit aus, kann ihm diese nicht nochmals zugewiesen werden (-> NOT SUCCESSFUL)
  - „release“: Entwickler kann nur entlassen werden, wenn er zuvor eingestellt wurde und gerade keine Tätigkeit ausübt bzw. an keinem Review teilnimmt
  - „release“: mit Entlassung wird Entwickler für eine Zeitspanne zw. 1 und 60 Tagen anderweitig verplant (-> er kann in dieser Zeit nicht wieder eingestellt werden)

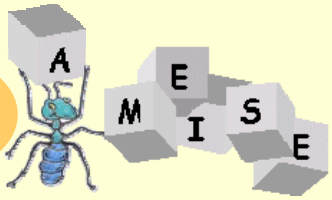
Hinweise (11/11)



- Zuteilung von Aufgaben:
  - Entwickler kann zu bestimmten Zeitpunkt nur eine Aufgabe ausführen (Ausnahme: **Review**!), sonst:
  - Entwickler ignoriert den neuen Auftrag zunächst, führ ihn aber aus, nachdem er alte Tätigkeit beendet hat (kann mehrere Zeitschritte dauern)
  - Entwickler ignoriert den neuen Auftrag (keine Rückmeldung) Entwickler beendet alte Tätigkeit und wartet auf neuen

Tipp: Entwickler erst alte Tätigkeit beenden lassen, bevor man ihm eine neue zuweist.

Start und Ende



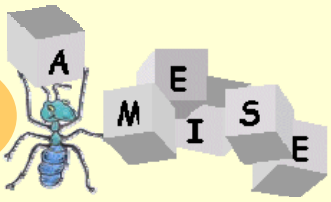
- Mit Kommando “deliver system” wird die Software an den Kunden übergeben
- Kundenreaktionen werden angezeigt
- Achtung!

  Kommando “cancel project” führt zu einem Projektabbruch!!! → Spiel ist nicht auswertbar!

- WICHTIG!!!

  Vergessen Sie nicht den Code zu integrieren!

  -> Auslieferung an den Kunden ist nur möglich, wenn mindestens 50 % der Anforderungen integriert wurden!!!



Besuchen Sie die Demo bzw. das Tutorial auf

http://ameise.uni-klu.ac.at

Alles Gute für Ihre erste Simulation!