Software-Engineering

Vorlesung 9 vom 13.12.2004

Sebastian Iwanowski
FH Wedel

Software-Engineering

Vorlesungsthemen:

1. Überblick über das Thema und die Vorlesung
2. Grundlegende Prinzipien
3. Softwareplanung
4. Systemanalyse
5. Softwareentwurf
6. CASE-Tools (UML und ARIS)
7. Aufwandsabschätzung
8. Qualitätsmanagement
9. Projektmanagement

Überblick Aufwandsabschätzung

Allgemeine Prinzipien:

- **Versuche, Analogien zu vergleichbaren Projekten / Aufgaben zu finden, deren Aufwand bekannt ist**
- **Berücksichtige das Maß der Ähnlichkeit zwischen den Aufgaben**
- **Verwende möglichst eigene Erfahrungen und – wenn diese nicht vorhanden sind – allgemein verbreitete Erkenntnisse und Statistiken**

Grundlegende Fragestellungen:

1) **Welche Größen sind relevant für den in einem Projekt geleisteten Aufwand ?**
2) **Wie bestimmt man das Maß der Ähnlichkeit zu einem anderen Projekt ?**

Berechnung relevanter Größen

LOC ist keine gute einzelne relevante Größe für ein Projekt:

LOC in Abhängigkeit von der Programmiersprache:

Language	Size in FP	Lang. Level	LOC per FP	Size in LOC
Assembly	1,500	1	250	375,000
C	**1,500**	**3**	**127**	**190,500**
CHILL	1,500	3	105	157,500
PASCAL	**1,500**	**4**	**91**	**136,500**
PL/I	1,500	4	80	120,000
Ada83	1,500	5	71	106,500
C++	**1,500**	**6**	**55**	**82,500**
Ada95	1,500	7	49	73,500
Objective C	1,500	11	29	43,500
Smalltalk	**1,500**	**15**	**21**	**31,500**
Average	1,500	6	88	131,700

Basismethoden

Balzert klassifiziert folgende Basismethoden:

- **Gewichtungsmethode**
- **Parametrische Gleichungen**
- **Multiplikatormethode**

} **für die Berechnung relevanter Größen**

- **Analogiemethode**
- **Relationsmethode**
- **Kennzahlenverfahren**

} **für das Bestimmen des Ähnlichkeitsmaßes**

- **Prozentsatzmethode** — **für die Berücksichtigung des Projektfortschritts**

Basismethoden: Berechnung relevanter Größen

Gewichtungsmethode:

- Bestimme Merkmale, die für die Abschätzung relevant erscheinen:
  - subjektive Merkmale, z.B. Qualifikation des Personals
  - objektive Merkmale, z.B. verwendete Programmiersprache
- Ordne jedem Merkmal Gewichtungswerte als Faktoren zu.
- Addiere die gewichteten Merkmalswerte zu einem Gesamtaufwand.

Parametrische Gleichungen:

- wie Gewichtungsmethode, aber mit einer Formel für den Gesamtaufwand, die nicht notwendigerweise linear ist.
- Merkmale und Gewichte werden nicht nach Gefühl bestimmt, sondern durch statistische Analysen (Korrelationsanalysen)

Basismethoden: Berechnung relevanter Größen

Multiplikatormethode:

- Zerlegung des Software-Produkts in Teilsysteme, bis jedem Teilsystem ein bekannter Umfang (z.B. in LOC) zugeordnet werden kann
- Zuordnung der Teilsysteme zu einigen charakteristischen Kategorien
- Multiplikation des Umfangs jedes Teilsystems mit dem Gewichtungsfaktor der jeweiligen Kategorie (Aufwand pro Einheit)

Basismethoden: Bestimmen des Ähnlichkeitsmaßes

Analogiemethode:

- Bestimme die Ähnlichkeit zu bereits abgeschlossenen Projekten nach einem der folgenden Kriterien:
  - gleiches oder ähnliches Anwendungsgebiet
  - gleicher oder ähnlicher Produktumfang
  - gleicher oder ähnlicher Komplexitätsgrad
  - gleiche Programmiersprache

Relationsmethode:

- wie Analogiemethode, aber mit einem formalisierten Verfahren:
  - Bilden von Faktorentabellen für Ähnlichkeitskriterien
  - Bilden von Verrechnungsformeln
- Berücksichtigung des Mehr- / Minderaufwands mit Punktzuschlägen / -abschlägen

Basismethoden: Bestimmen des Ähnlichkeitsmaßes

Kennzahlenverfahren:

- Auswertung eigener abgeschlossener Projekte durch Bestimmung von Kennzahlen für:
  - Anforderungen aus dem Lastenheft
  - Messgröße der entstanden Software (z.B. LOC)
  - benutzte Programmiersprache
  - ...
- Verwendung dieser Kennzahlen zum Bestimmen des Ähnlichkeitsmaßes

Basismethode: Berücksichtigung des Projektfortschritts

Prozentsatzmethode:

- Ermittlung von typischen prozentualen Verteilungen des Entwicklungsaufwandes auf die verschiedenen Entwicklungsphasen
- Alternative Vorgehensweisen:
  - Eine Phase abschließen, aus dem Aufwand mittels prozentualem Anteil Gesamtaufwand errechnen
  - Eine Phase detailliert schätzen, daraus den Gesamtaufwand abschätzen

Beispiele für Verteilungen:

Bertelsmann (nach Bender 83)	
Definition	30%
Entwurf	30%
Codierung	15-20%
Test	20-25%

Hewlett-Packard (nach Grady 92)	
Definition	18%
Entwurf	19%
Codierung	34%
Test	29%

Basismethode: Berücksichtigung des Projektfortschritts

Auch die relative Aufteilung auf die SWE-Phasen hängt von der Wahl der Programmiersprache ab:

Language	Req. Analysis (Months)	Design (Months)	Code (Months)	Test (Months)	Doc. (Months)	Management (Months)	TOTAL (Months)
Assembly	13.64	60.00	300.00	277.78	40.54	89.95	781.91
C	13.64	**60.00**	**152.40**	**141.11**	**40.54**	**53.00**	**460.69**
CHILL	13.64	60.00	116.67	116.67	40.54	45.18	392.69
PASCAL	13.64	**60.00**	**101.11**	**101.11**	**40.54**	**41.13**	**357.53**
PL/I	13.64	60.00	88.89	88.89	40.54	37.95	329.91
Ada83	13.64	60.00	76.07	78.89	40.54	34.99	304.13
C++	13.64	**68.18**	**66.00**	**71.74**	**40.54**	**33.81**	**293.91**
Ada95	13.64	68.18	52.50	63.91	40.54	31.04	269.81
Objective C	13.64	68.18	31.07	37.83	40.54	24.86	216.12
Smalltalk	13.64	**68.18**	**22.50**	**27.39**	**40.54**	**22.39**	**194.64**
Average	13.64	63.27	100.72	100.53	40.54	41.43	360.13

Allumfassende Schätzverfahren

1) Function Point Methode (FP):

- entwickelt von Allen J. Albrecht (1979)
- wird in user group IFPUG (http://www.ifpug.org) kontinuierlich weiterentwickelt
- mehrere kommerzielle Anbieter für Abschätzsoftware nach dieser Methode

2) Constructive Cost Model (COCOMO):

- entwickelt von Barry Boehm (1981)
- wurde in den 90er Jahren an der University of Southern California (USC) weiterentwickelt (COCOMO II)
- Abschätzsoftware als Freeware erhältlich (http://sunset.usc.edu/research/COCOMOII/)

Allumfassende Schätzverfahren: FP

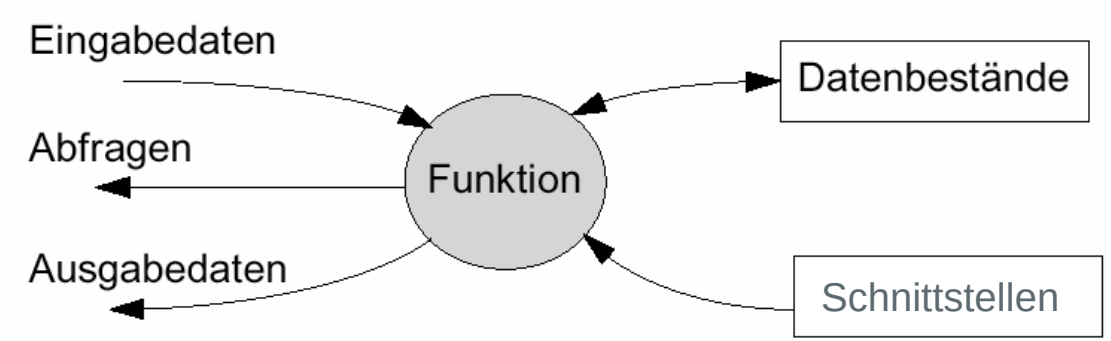
1) Function Point Methode (FP):

Ziele:

- Abschätzung des Aufwands in Abhängigkeit vom Umfang und vom Schwierigkeitsgrad des neuen Produkts
- Ableitung des Umfangs direkt aus den Anforderungen

Vorgehen:

- Zuordnung jeder Produktanforderung in eine von fünf Kategorien, die sich um die Funktionalität des Produkts gruppieren
- Einordnung jeder einzelnen Anforderung in eine der Klassen "Einfach", "Mittel", "Komplex" mit Hilfe von Tabellen und Richtlinien
- Eintragen der einzelnen Anforderungen in ein Formular zur Berechnung der Function Points



Allumfassende Schätzverfahren: FP

1) Function Point Methode (FP):

Gewichtungstabelle:

		einfach	mittel	komplex	Werte
Eingabefunktionen	**EF**	3	4	5	
Ausgabefunktionen	**AF**	4	5	7	
Abfragefunktionen	**AF**	3	4	6	
Dateneinheiten	**DE**	7	10	15	
Schnittstellen	**SES**	5	7	10	
				Summe	

Summe(Eingabefunktionen (EF) * Gewichte)

Summe(Zahl der Berechnungsfunktionen (BF) * Gewichte)

Summe(Zahl der Abfragefunktionen (AF) * Gewichte)

Summe(Zahl der gespeicherten Dateneinheiten (DE) * Gewichte)

Summe(Zahl der ext. Schnittstellen (SES) * Gewichte)

} Gesamtsumme ergibt Unadjusted Function Points (**UFP**)

Allumfassende Schätzverfahren: FP

1) Function Point Methode (FP):

Katalog mit Zuschlägen und Abschlägen

➔ System Complexity Faktor (**SCF**)

FP = UFP • SCF

Allumfassende Schätzverfahren: FP

1) Function Point Methode (FP):

Was benötigt man zur Bestimmung der Eingabedaten in das FP-Schätzverfahren ?

Nützliche Spezifikationsdaten:

- Bildschirmmasken
- Datenbankentwürfe
- Funktionsmodelle
- Schnittstellen
- Druckoutput-Entwürfe

Allumfassende Schätzverfahren: FP

1) Function Point Methode (FP):

Wie kann man die FP effizient nach Fertigstellung der SW schätzen ?

- Miss die Lines of Code !
- Dividiere durch Faktor der jeweiligen Programmiersprache !
  (siehe Tabelle)

Warum will man das machen ?

Allumfassende Schätzverfahren: FP

1) Function Point Methode (FP):

Vorteile:

- SW-Komplexität ist abschätzbar unabhängig von der Programmiersprache
- FP-Methode kann schnell erlernt werden
- Mit FP wird auch Produktivität ex post beurteilbar (FP / Personenmonat)
- Aus FPs kann man viele weitere Erfahrungsgrößen vorhersagen - zum Beispiel die Anzahl der erwarteten Fehler

Allumfassende Schätzverfahren: FP

1) Function Point Methode (FP):

Nachteile:

- Die Spezifikation muss immer noch in einer Ausführlichkeit vorliegen, die zum Zeitpunkt der Angebotserstellung noch nicht vorhanden ist.
- Die Ersparnis durch Wiederverwendung ist problematisch miteinzubeziehen (wieviel FPs Rabatt sollte man bei Wiederverwendung eines Moduls mit 2.500 FP geben)
- FP ist ausgelegt auf funktionale Technologie

➢ Allerdings gibt es für OO, Web, .. Erweiterungen

Allumfassende Schätzverfahren: COCOMO

2) COCOMO:

Prinzip der Originalversion von Boehm:

Unterscheide 3 Projektkategorien zur Anpassung der Kennzahlen:

- Einfache Projekte
  - vertraute Systemumgebung
  - große Erfahrung
  - wenige Schnittstellen
- Mittelschwere Projekte
- Komplexe Projekte
  - enge Verzahnung mit Systemumgebung
  - geringe Erfahrung
  - zahlreiche Schnittstellen

Allumfassende Schätzverfahren: COCOMO

2) COCOMO:

Prinzip der Originalversion von Boehm:

$$A = C * KLOC^{B}$$

mit
A: Entwicklungsaufwand in MM
B,C: Konstanten gemäß Tabelle

$$T = D * A^{E}$$

mit
T: Entwicklungszeit
D,E : Konstanten gemäß Tabelle

Projekt	C	B
Einfach	2,4	1,05
Mittel	3,0	1,12
Komplex	3,6	1,20

Projekt	D	E
Einfach	2,5	0,38
Mittel	2,5	0,35
Komplex	2,5	0,32

Allumfassende Schätzverfahren: COCOMO

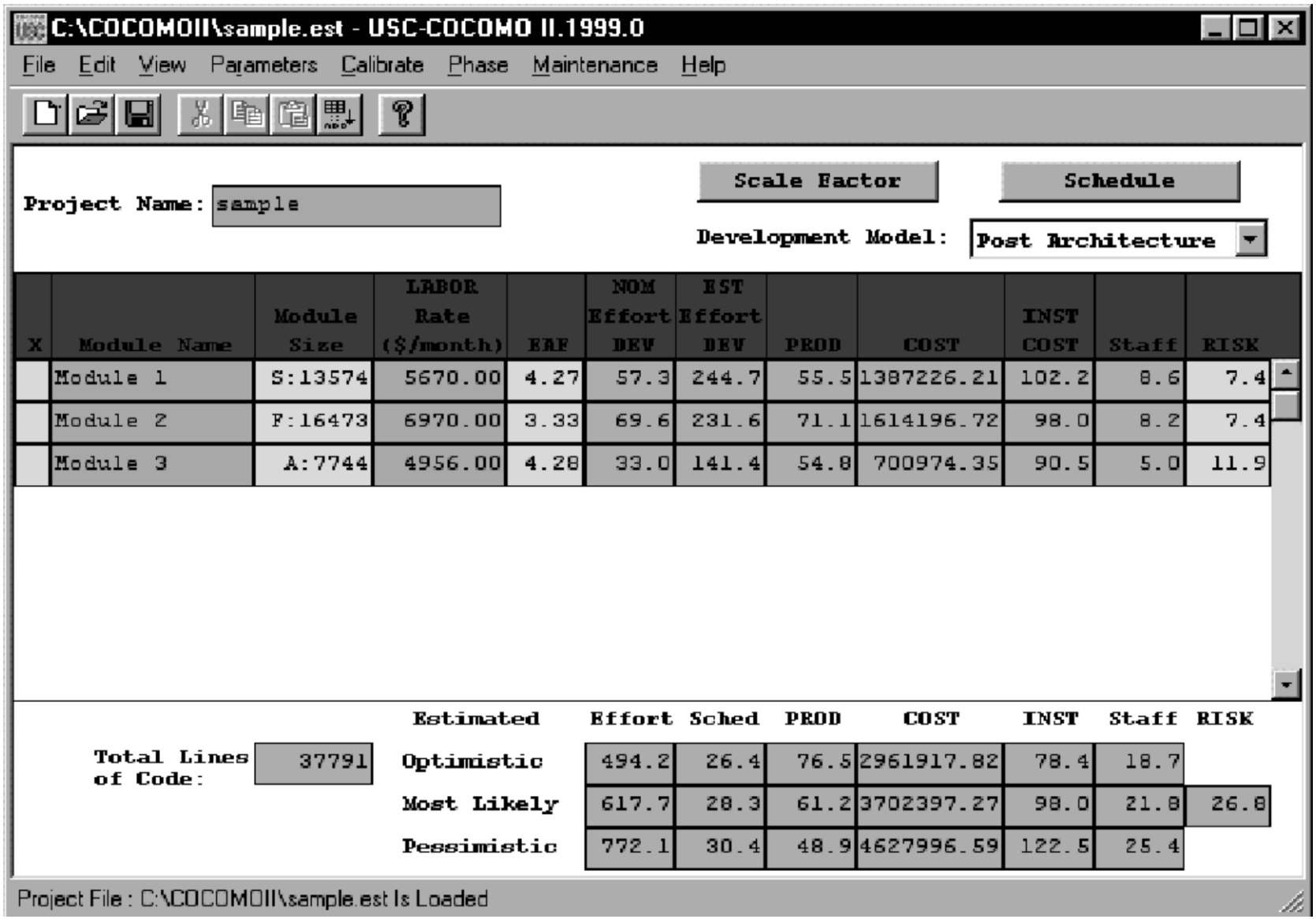
2) COCOMO:

Weiterentwicklung zu COCOMO II an der USC:

- Berücksichtigung von insgesamt 14 Kosteneinflussfaktoren, darunter: Produkt-, Rechner-, Personal- und Projektmerkmale
- Einordnung in jedes Kosteneinflussfaktors in die Kategorien 'sehr gering', 'gering', 'normal', 'hoch', 'sehr hoch' und 'extrem'
- Zuordnung von Faktorenwerten F zu den Kategorien in einer Tabelle
- Multiplikative Verknüpfung aller Faktoren mit der ursprünglichen Formel

Die Software der USC kann heruntergeladen werden von:

http://sunset.usc.edu/research/COCOMOII/



Software-Engineering

Vorlesungsthemen:

1. Überblick über das Thema und die Vorlesung
2. Grundlegende Prinzipien
3. Softwareplanung
4. Systemanalyse
5. Softwareentwurf
6. CASE-Tools (UML und ARIS)
7. Aufwandsabschätzung
8. Qualitätsmanagement
9. Projektmanagement

Qualitätsmanagement

Qualitätskriterium: Korrektheit

Korrektheit bedeutet:

- Übereinstimmung zwischen funktioneller Spezifikation und Programmfunktionalität

Nachweis der Korrektheit:

- durch Programmverifikation oder durch systematische Tests

- Korrektheitsbeweise mit Programmverifikation nur für kleine Teilalgorithmen möglich
- Vollständige Tests aller Programmzustände zu aufwändig

- Mathematische Korrektheit kann nicht bewiesen werden
- In der Praxis akzeptierte Korrektheit immer noch schwierig

Qualitätsmanagement

Was ist eigentlich Qualität ?

Software-Qualität bedeutet:

Nach DIN 55 350:

- "Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Produkts oder einer Tätigkeit, die sich auf die Eignung zur Erfüllung gegebener Erfordernisse beziehen"

Das bedeutet für die Praxis:

- SW-Qualität ist mehr als Korrektheit
- SW-Qualität ist kein exakt definierter Begriff
- ist nicht exakt messbar
- SW-Qualität wird anhand von verschiedenen Kriterien charakterisiert
- SW-Qualität ist relativ

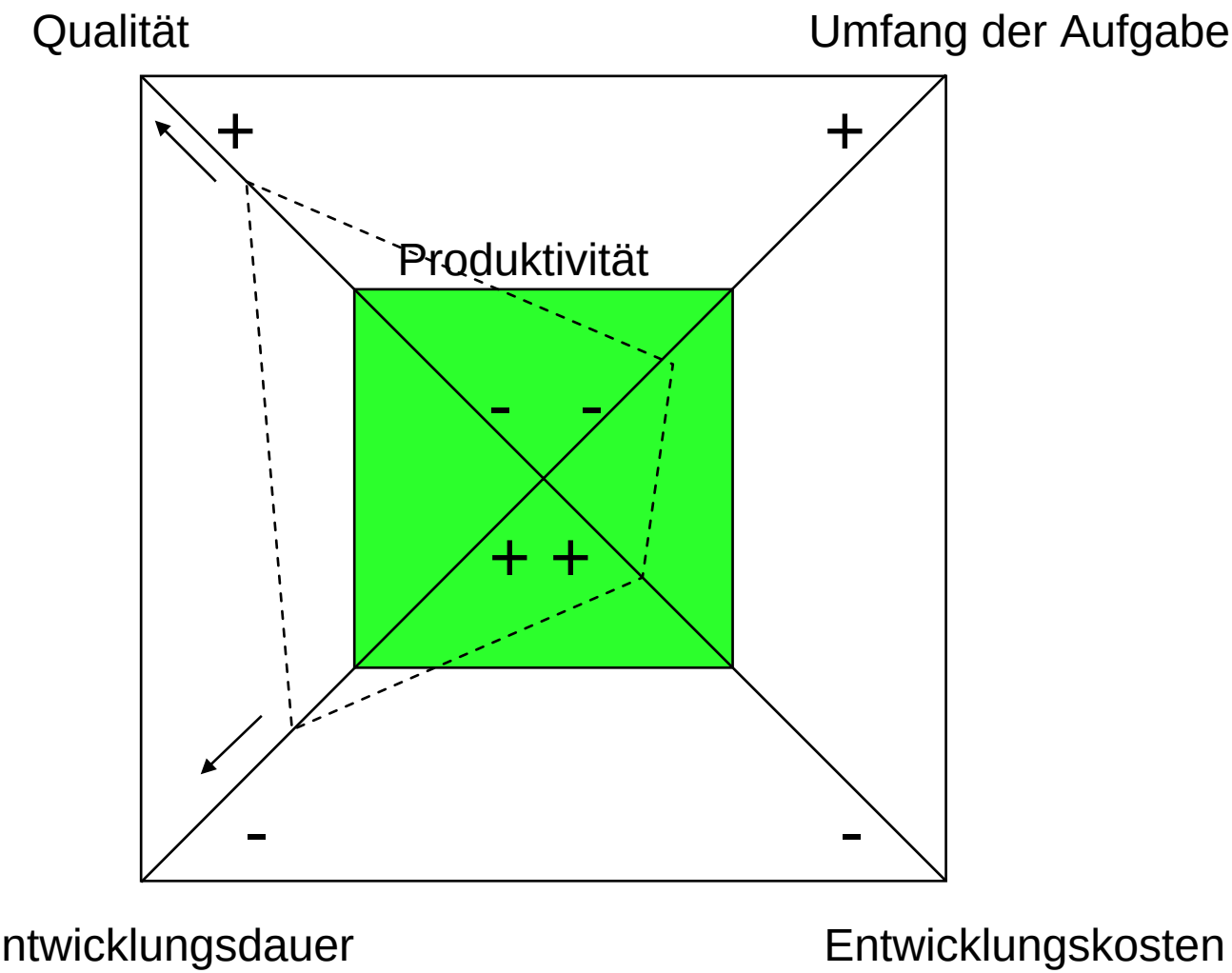
Qualitätsmanagement

Verschiedene Qualitätskriterien:

- Korrektheit
- Effizienz
- Robustheit
- Verfügbarkeit
- Zuverlässigkeit
- Benutzungsfreundlichkeit
- Sicherheit
- Verständlichkeit
- Änderbarkeit
- Prüfbarkeit
- Wiederverwendbarkeit (Portabilität)

Qualitätsmanagement

Qualität ist nicht das Maß aller Dinge !



Beim nächsten Mal:
Qualitätsmanagement (im Detail)