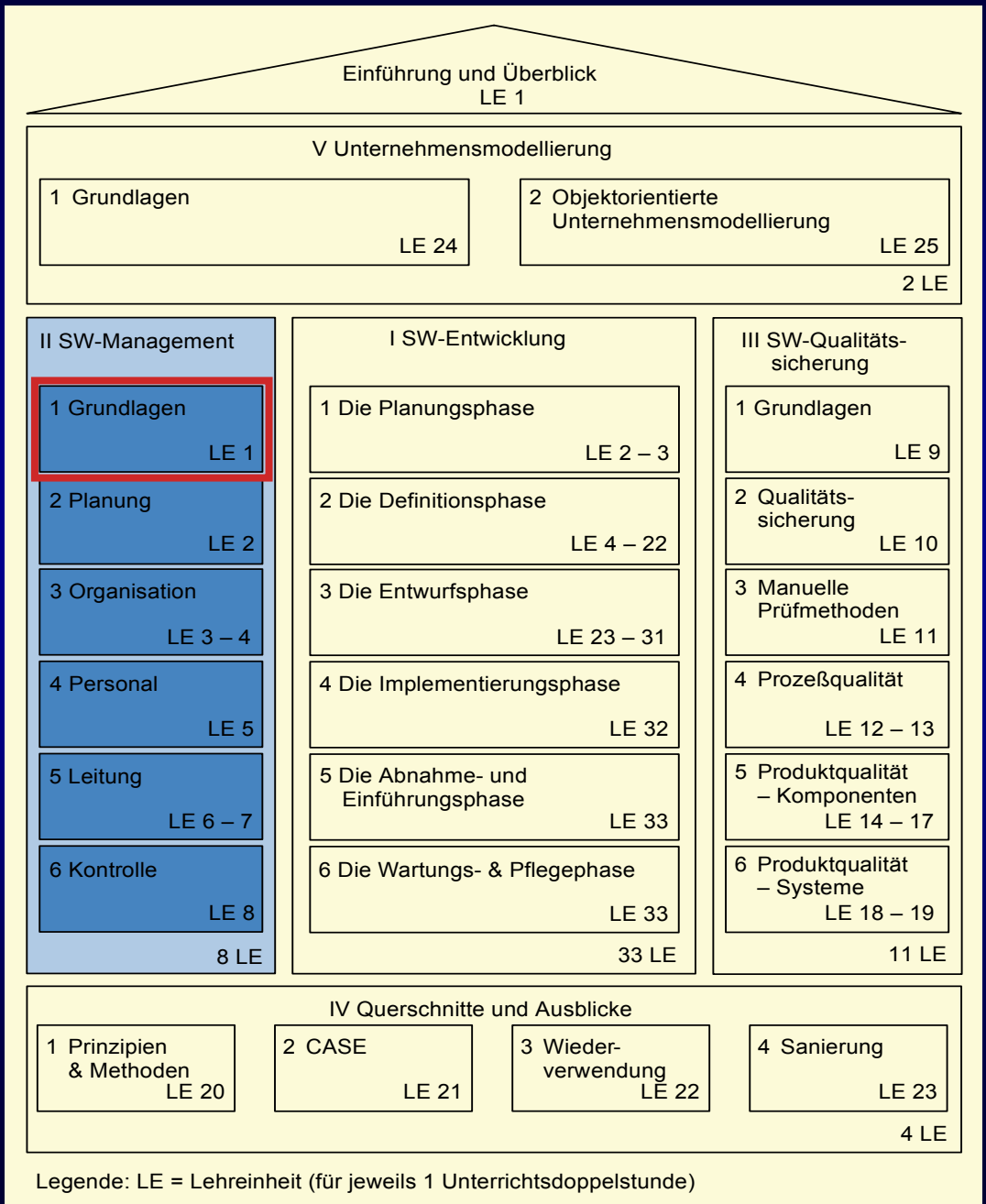
Softwaretechnik II
Software-Management

1 Grundlagen

Prof. Dr. Joachim Hertel
Fachrichtung Informatik
Universität des Saarlandes

II Software-Management - Grundlagen



Legende: LE = Lehreinheit (für jeweils 1 Unterrichtsdoppelstunde)

Lernziele

- Definitionen für die Software-Produktivität und ihre Problematik aufzeigen können
- Den Zusammenhang zwischen Produktivität und Qualität darstellen können
- Für gegebene Szenarien notwendige Managementaktivitäten identifizieren können
- Für gegebene Szenarien Produktivitätsverbesserungsmaßnahmen vorschlagen und begründen können.

Inhalt

Zur Lerneffizienz

- Wieviel behält man?
  - lesen 10%
  - hören 25%
  - sehen 25%
  - hören und sehen 50%
  - selbst etwas sagen 70%
  - selbst etwas tun 90%.

Fragen

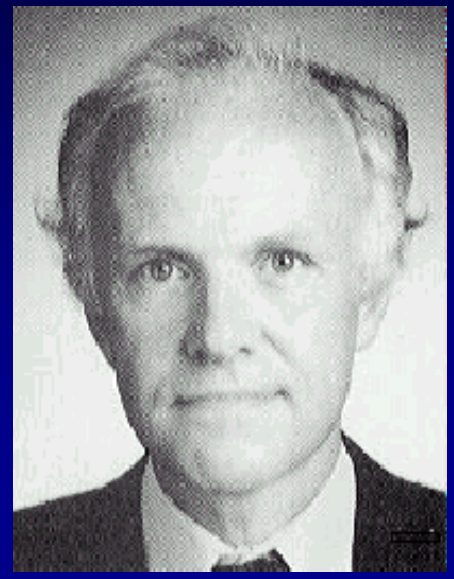
- Wer will (Software-) Manager werden?
  - Warum bzw. warum nicht?
- Wie wichtig ist Fachwissen?
- Wer hat schon einen Rhetorik-Kurs besucht?
- Welche Aufgaben hat ein Manager zu erledigen?
- Wodurch unterscheidet sich Software-Management vom Management anderer Ingenieurbereiche?

Literaturempfehlungen

- Lehrbuch der Software-Technik, Band 1
- Wien wartet auf Dich! *Peopleware* (engl. Ausgabe) von DeMarco, Lister
- Wachstum durch Verzicht – Schneller Wandel zur Weltklasse: Vorbild Elektronikindustrie von McKinsey & Company, Inc.

1 Grundlagen

- Zur Historie
  - **Tom DeMarco**
    *20.8.1940 in Hazleton, Pennsylvania, USA
    Principal, The Atlantic Systems Guild
  - Erfinder der Strukturierten Analyse (SA)
  - Buch: Structured Analysis and System Specification, 1978
  - **Bedeutende Beiträge zum Software-Management**
  - Buch: Peopleware, zusammen mit T. Lister, 1987
  - Buch: Controlling Software Projects, 1982.



1.1 Einführung

- Drei primäre Managementstrategien:
  - Maximierung der Kundenzufriedenheit
  - Minimierung des Aufwands und der Zeit der Softwareerstellung
  - Minimierung von Fehlern
- Das Endziel aller drei Strategien ist die Kundenzufriedenheit.

1.1 Einführung

- **Charakteristika von Software-Geschäftsstrategien**

Hauptcharakteristika	Max. Kundenzufriedenheit	Min. Aufwand & Zeit	Min. Fehler
Hauptgeschäftsstrategie Effizient...	Marktanteile erlangen ...beim ersten Markteinstieg	Konkurrenz erfordert neue Produkte oder Kostenkontrolle ...bei mehreren Konkurrenzprodukten oder wenn man profitablere Produkte verkauft	Halten oder vergrößern des Marktanteils ...bei konkurrenzfähigen Eigenschaften und ein adäquater Marktanteil gehalten wird
Charakteristische Eigenschaften	Kommunikation mit dem Kunden; schnelle Reaktion	Fokus auf Auslieferungsdatum und Aufwand	Analyse und Entfernen von Fehlerursachen

1.1 Einführung

- **Software-Management vs. Management anderer Ingenieurbereiche**
  - **Besonderheiten**
    - **Produkt ist immateriell**
    - **Entwicklungsfortschritt objektiv nicht zu ermitteln**
    - **Eine Software-Entwicklung verläuft nicht-deterministisch.**

1.1 Einführung

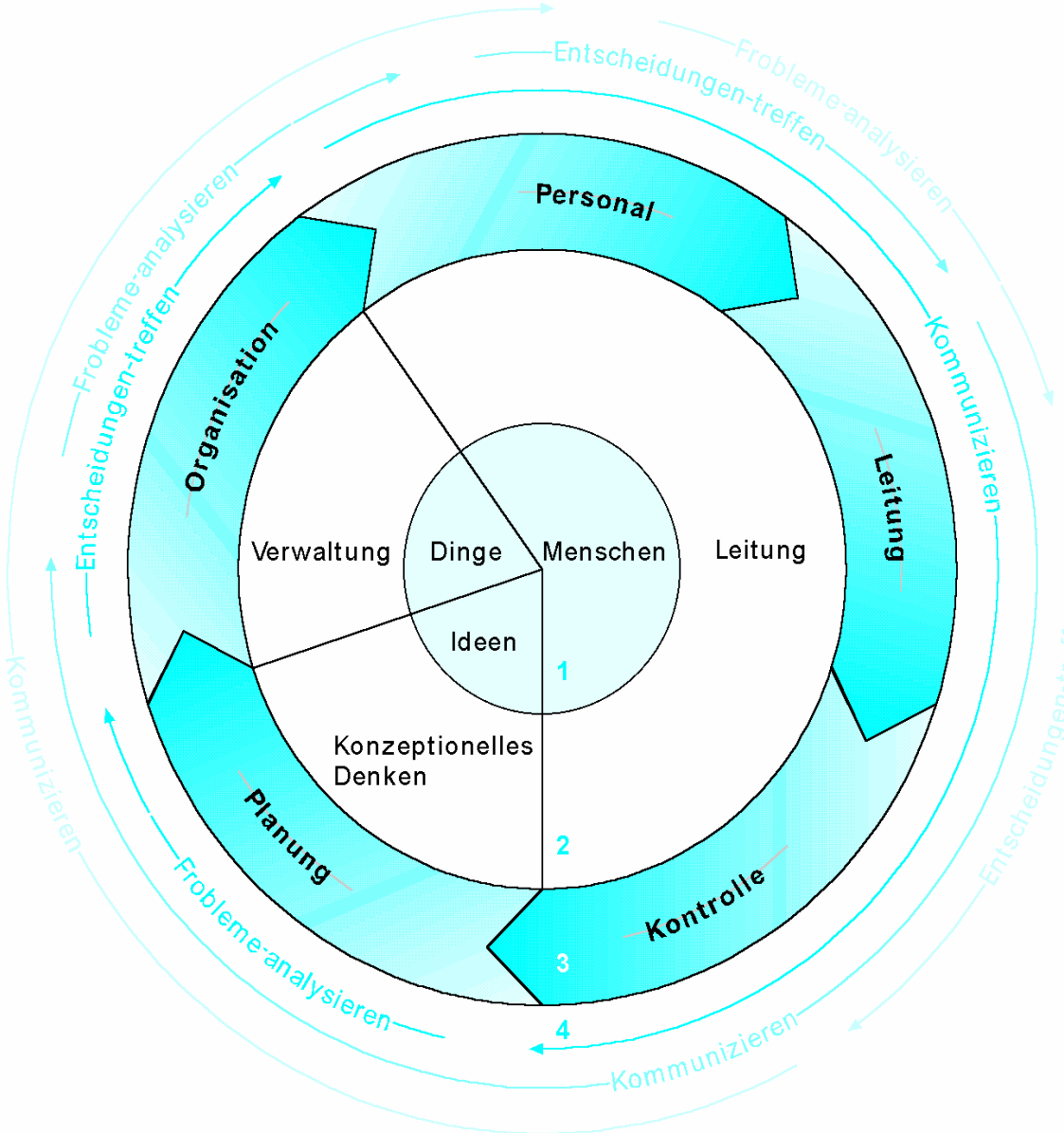
- Noch kein klares Verständnis vom Entwicklungsprozeß
- Große Software-Systeme sind einmalige Entwicklungen
- Unteilbarkeit der Arbeit
- Die Software-Technik ist keine Naturwissenschaft
- Hoher Grad an Abstraktion, bei gleichzeitig niedrigem Grad an Normierung.

1.1 Einführung

- Konsequenzen
  - Für einen Außenstehenden Unterschiede nur schwer zu erkennen
    - Viele Manager aus anderen Branchen scheitern, wenn sie Software-Entwicklungen managen
  - Viele Software-Manager sind von Managementideen geprägt, die aus typischen Produktionsprozessen stammen.

1.2 Aufgaben

- Der Management-Prozeß



Legende: 1 Elemente
2 Aufgaben
3 Sequentielle Funktionen
4 Kontinuierliche Funktionen

1.2 Aufgaben

- **Definition**
  - **Management umfaßt alle Aktivitäten und Aufgaben, die von einem oder mehreren Managern durchgeführt werden, um die Aktivitäten von Mitarbeitern zu planen und zu kontrollieren, damit ein Ziel oder der Abschluß einer Aktivität erreicht wird, die durch die Mitarbeiter alleine nicht erreicht werden können**
- **Kurzform:**
  - **Management sorgt dafür, daß Ziele durch Mitarbeiter erreicht werden.**

1.2 Aufgaben

- **Planungsaktivitäten**
  - Ziele setzen
  - Strategien und Taktiken entwickeln
  - Termine festlegen
  - Entscheidungen treffen
  - Vorgehensweisen und Regeln festlegen
  - Zukünftige Situationen vorhersehen
  - Budgets vorbereiten.

1.2 Aufgaben

- **Organisationsaktivitäten**
  - **Identifizieren und Gruppieren der zu erledigenden Aufgaben**
  - **Auswahl und Etablierung organisatorischer Strukturen**
  - **Festlegen von Verantwortungsbereichen und disziplinarischen Vollmachten**
  - **Festlegen von Qualifikationsprofilen für Positionen.**

1.2 Aufgaben

- **Personalaktivitäten**
  - Positionen besetzen
  - Neues Personal einstellen und integrieren
  - Aus- und Weiterbildung von Mitarbeitern
  - Personalentwicklung planen
  - Mitarbeiter beurteilen
  - Mitarbeiter bezahlen
  - Mitarbeiter versetzen oder entlassen.

1.2 Aufgaben

- Leitungsaktivitäten
  - Mitarbeiter führen und beaufsichtigen
  - Kompetenzen delegieren
  - Mitarbeiter motivieren
  - Aktivitäten koordinieren
  - Kommunikation unterstützen
  - Konflikte lösen
  - Innovationen einführen.

1.2 Aufgaben

- **Kontrollaktivitäten**
  - **Prozeß- und Produktstandards entwickeln und festlegen**
  - **Berichts- und Kontrollwesen etablieren**
  - **Prozesse und Produkte vermessen**
  - **Korrekturaktivitäten initiieren**
  - **Loben und Tadeln.**

1.3 Produktivität

- **Produktivitätssteigerung kann verschiedenes bedeuten:**
  - **Software-Produkte schneller, d.h. in kürzerer Kalenderzeit entwickeln**
    - **Oft als Effizienzsteigerung bezeichnet**
  - **Software-Produkte so entwickeln, daß sie einen höheren *»Return on Investment«* liefern**
    - **Es soll weniger Geld ausgegeben werden, um Produkte mit gleichen Anforderungen zu entwickeln und zu pflegen**
  - **Software-Produkte mit besserer Qualität entwickeln.**

1.3 Produktivität

- **Produktivitätsmaße**

$$\text{Produktivität} = \frac{\text{Leistung}}{\text{Aufwand}}.$$

1.3 Produktivität

- **Ansätze, um Leistung und Aufwand von Software zu definieren:**
  - **/Boehm 87/:**

$$\text{Produktivität} = \frac{\text{Produzierte Ergebnisse}}{\text{Eingesetzter Aufwand}}$$

  - **/Sneed 87/:**

$$\text{Produktivität} = \frac{\text{Anzahl Software - Elemente}}{\text{Geleistete Mitarbeitertage}} .$$

1.3 Produktivität

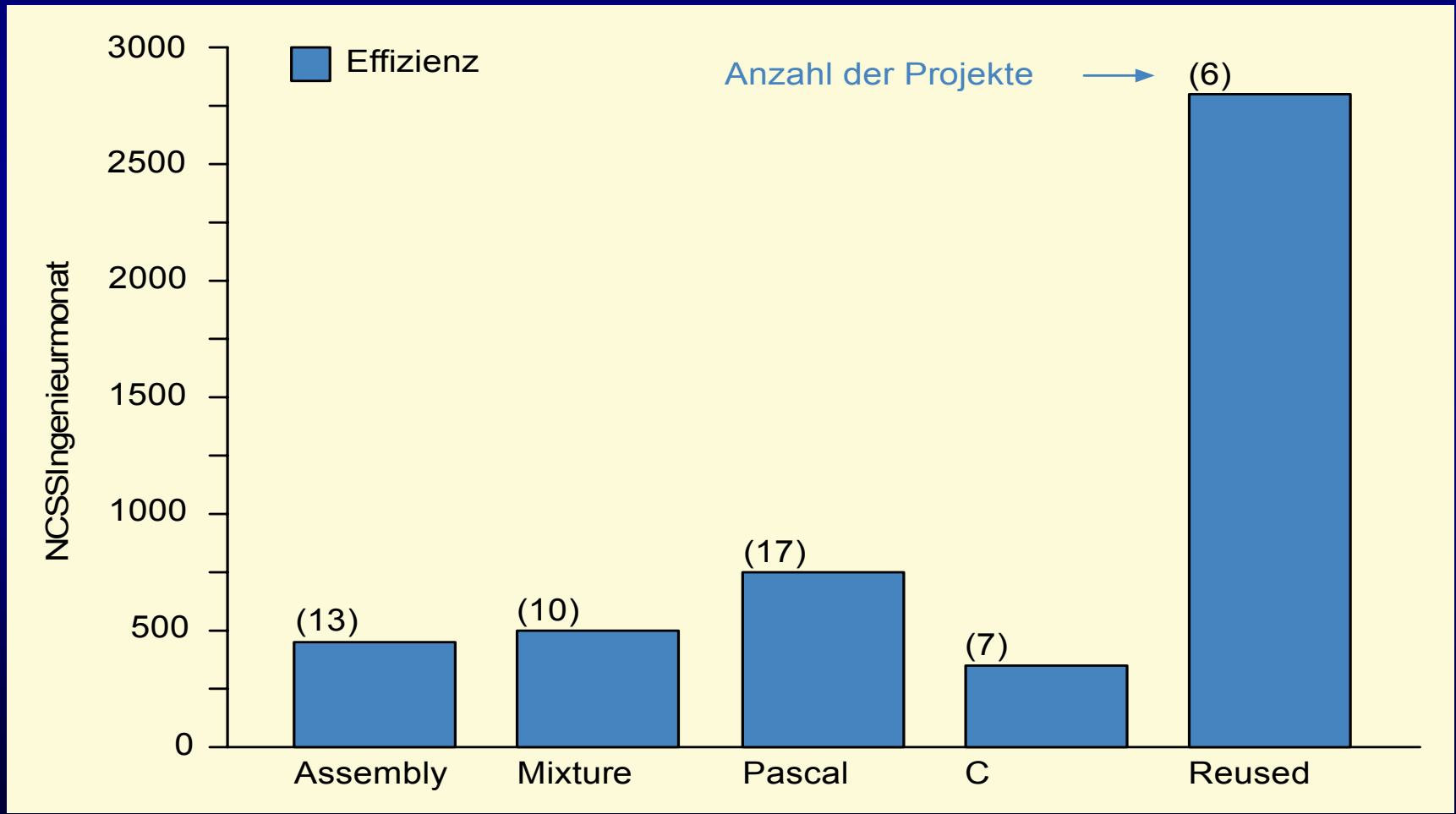
- **Produktivität kann verbessert werden, wenn...**
  - **die Ergebnisse vermehrt**
  - **der Aufwand verringert oder**
  - **die Ergebnisse vermehrt und der Aufwand verringert werden.**

1.3 Produktivität

- Anzahl Software-Elemente
  - Firma HP:
    - NCSS
      - Anzahl Quellenanweisungen einschl. Compiler-Anweisungen, Datendeklarationen und ausführbaren Anweisungen, aber ohne Leerzeilen oder Zeilen, die vollständig aus Kommentaren bestehen
  - Anderes Maß:
    - *Function Points.*

1.3 Produktivität

- **Durchschnittliche NCSS/Ingenieurmonate pro Sprache**



1.3 Produktivität

- Eingesetzter Aufwand
  - Personalkosten
  - Kosten für Computerressourcen
  - Hilfsmittel.

  - Ende 1. Vorelesung.

1.3 Produktivität

- **Geistige vs. körperlicher Anwesenheit**

$$\text{Umweltfaktor} = \frac{\text{ungestörte Stunden}}{\text{Stunden körperlicher Anwesenheit}}$$

  - **Umweltfaktor hoch:**
    - **Arbeitsumgebung erlaubt den Mitarbeitern eine hohe Produktivität**
  - **40% sind ein erreichbarer Wert**
  - **Innerhalb einer Firma:**
    - **Unterschiede zwischen 10% und 38%**
- **Aufwand bei der Firma HP:**
  - **Quantifiziert durch benötigte Ingenieurmonate:**
    - **40 - 50 Stunden/Monat ohne Urlaub & Krankheit.**

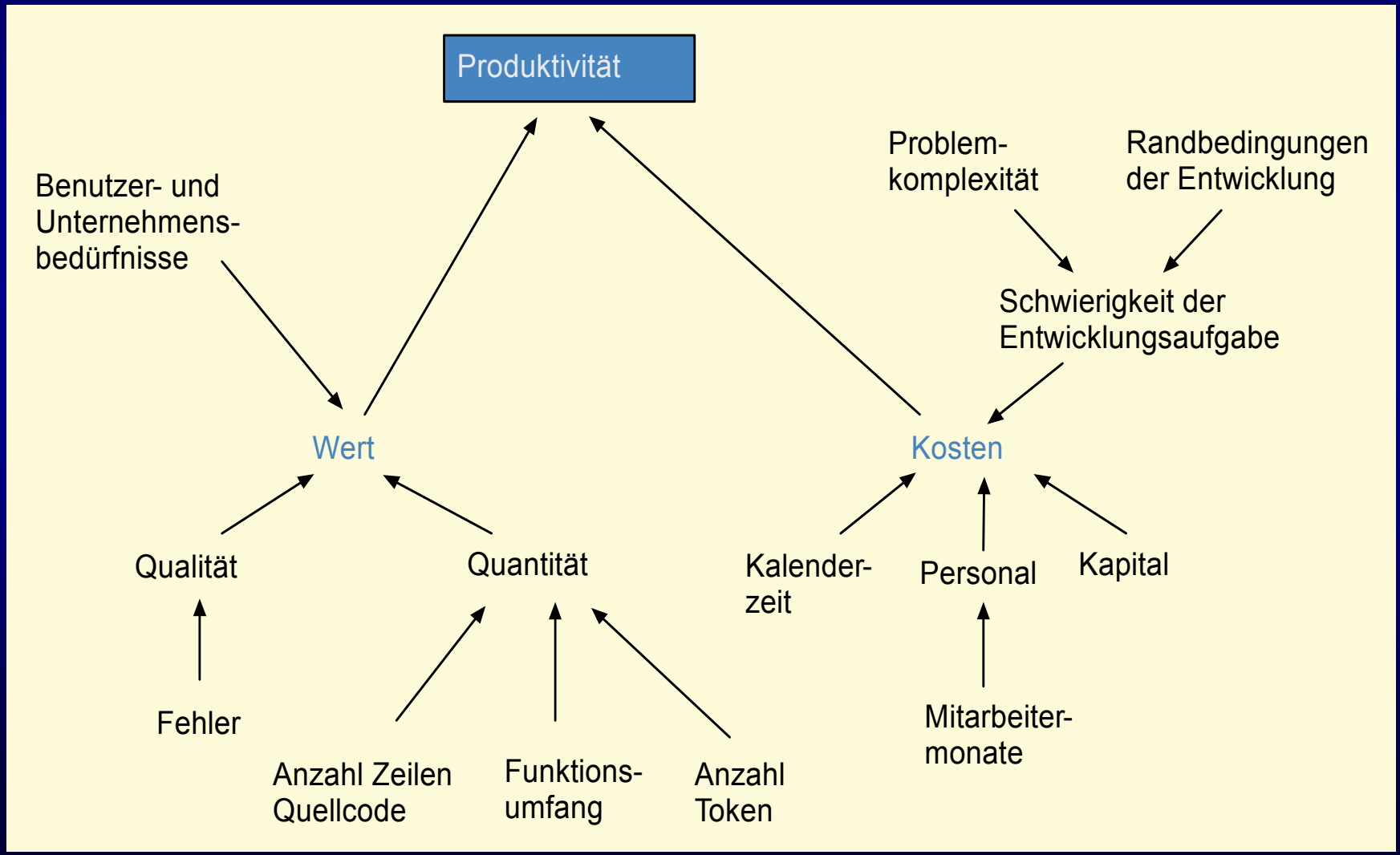
1.3 Produktivität

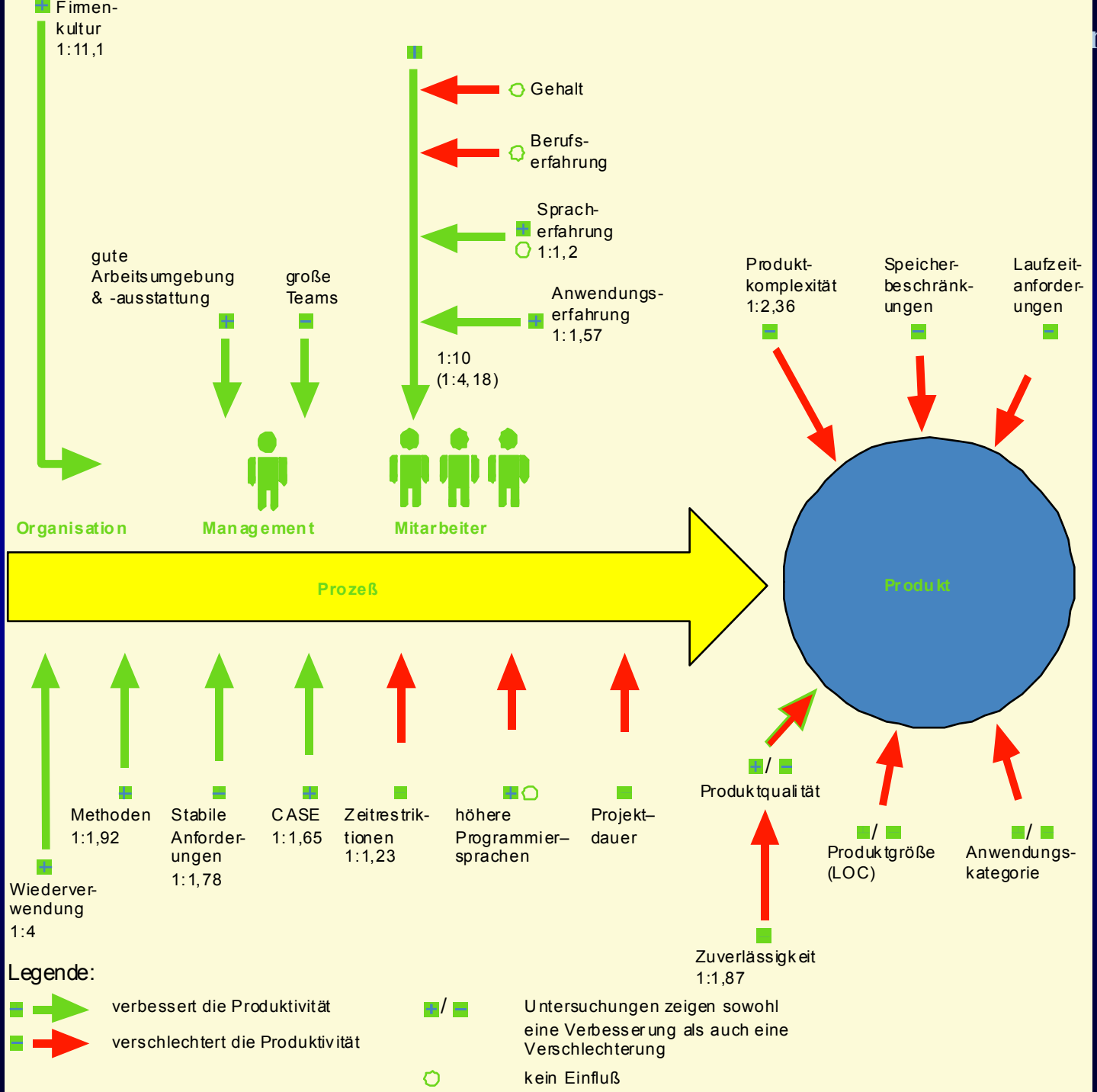
- **Basili:**
  - **Ersetzt produzierte Ergebnisse durch Produktwert**
  - **Software-Produktion ist ein Prozeß der Wertschöpfung**
  - **Das entstehende Produkt stellt einen Wert dar:**

$$\text{Produktivität} = \frac{\text{Produktwert}}{\text{Kosten}} .$$

1.3 Produktivität

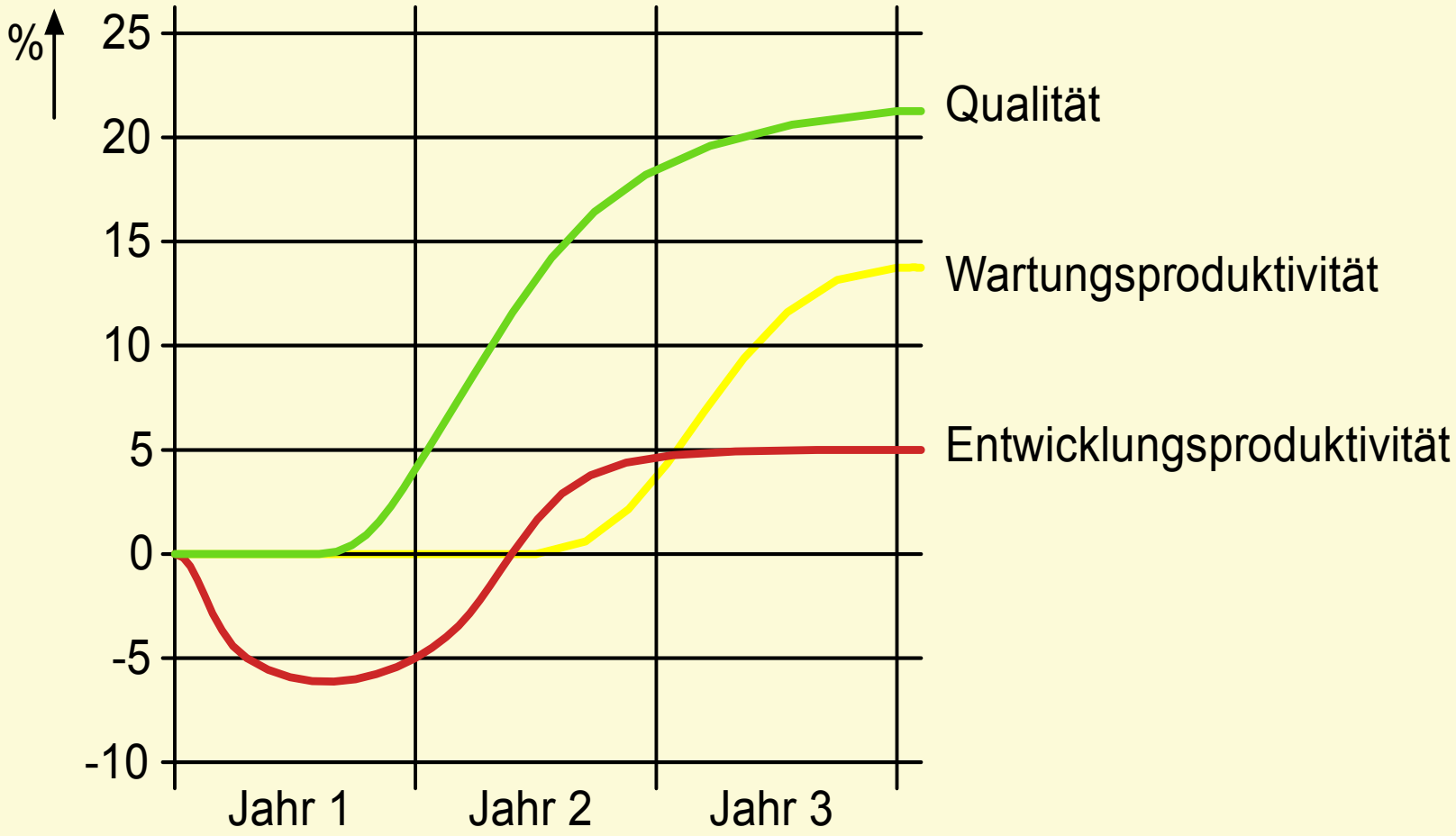
- **Produktivität und ihre Einflußfaktoren (Basili)**




Firmen-
kultur
1:11,1
Gehalt
Berufs-
erfahrung
Sprach-
erfahrung
1:1,2
Anwendungs-
erfahrung
1:1,57
1:10
(1:4,18)
gute
Arbeitsumgebung
& -ausstattung
große
Teams
Produkt-
komplexität
1:2,36
Speicher-
beschränk-
ungen
Laufzeit-
anforder-
ungen
Organisation
Management
Mitarbeiter
Prozeß
Produkt
Methoden
1:1,92
Stabile
Anforder-
ungen
1:1,78
CASE
1:1,65
Zeitrestrik-
tionen
1:1,23
höhere
Programmier–
sprachen
Projekt–
dauer
Wiederver-
wendung
1:4
Produktqualität
Produktgröße
(LOC)
Anwendungs-
kategorie
Zuverlässigkeit
1:1,87
Legende:
verbessert die Produktivität
verschlechtert die Produktivität
Untersuchungen zeigen sowohl
eine Verbesserung als auch eine
Verschlechterung
kein Einfluß

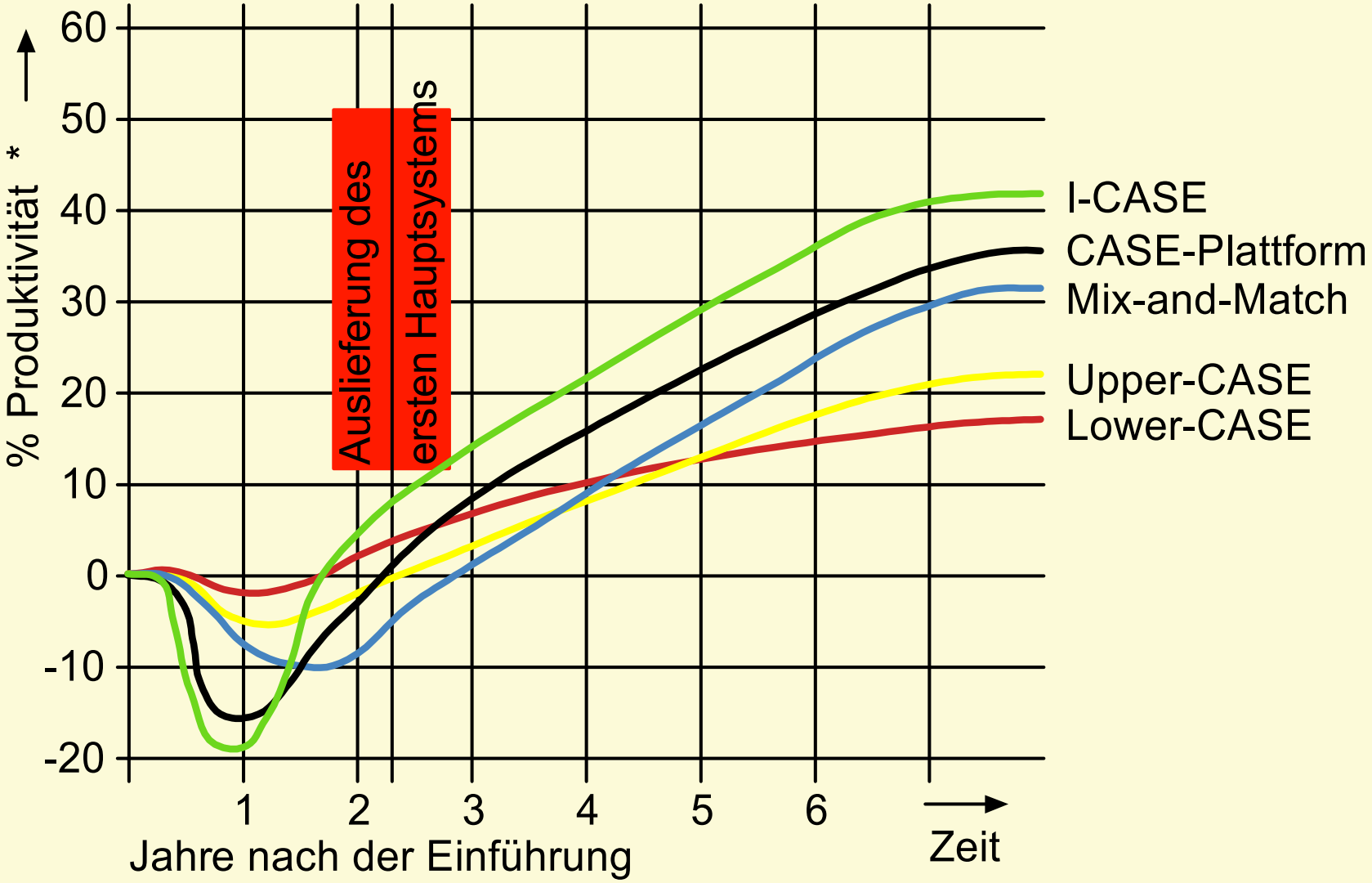
1.4 Einflußfaktoren

- **Produktivitäts- und Qualitätsverlauf durch den Einsatz von CASE**



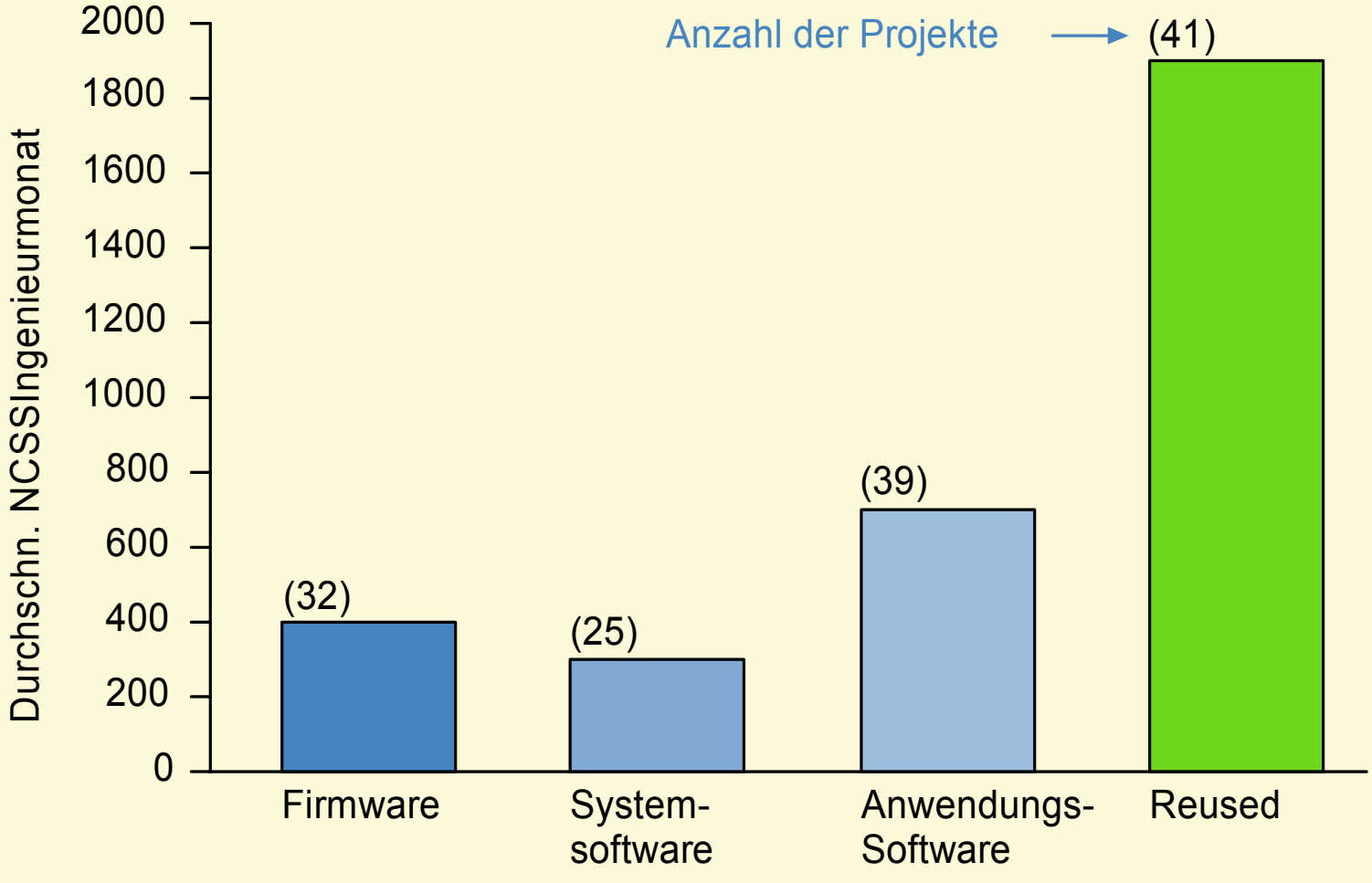
1.4 Einflußfaktoren

- **Produktivität abhängig vom CASE-Umfang**



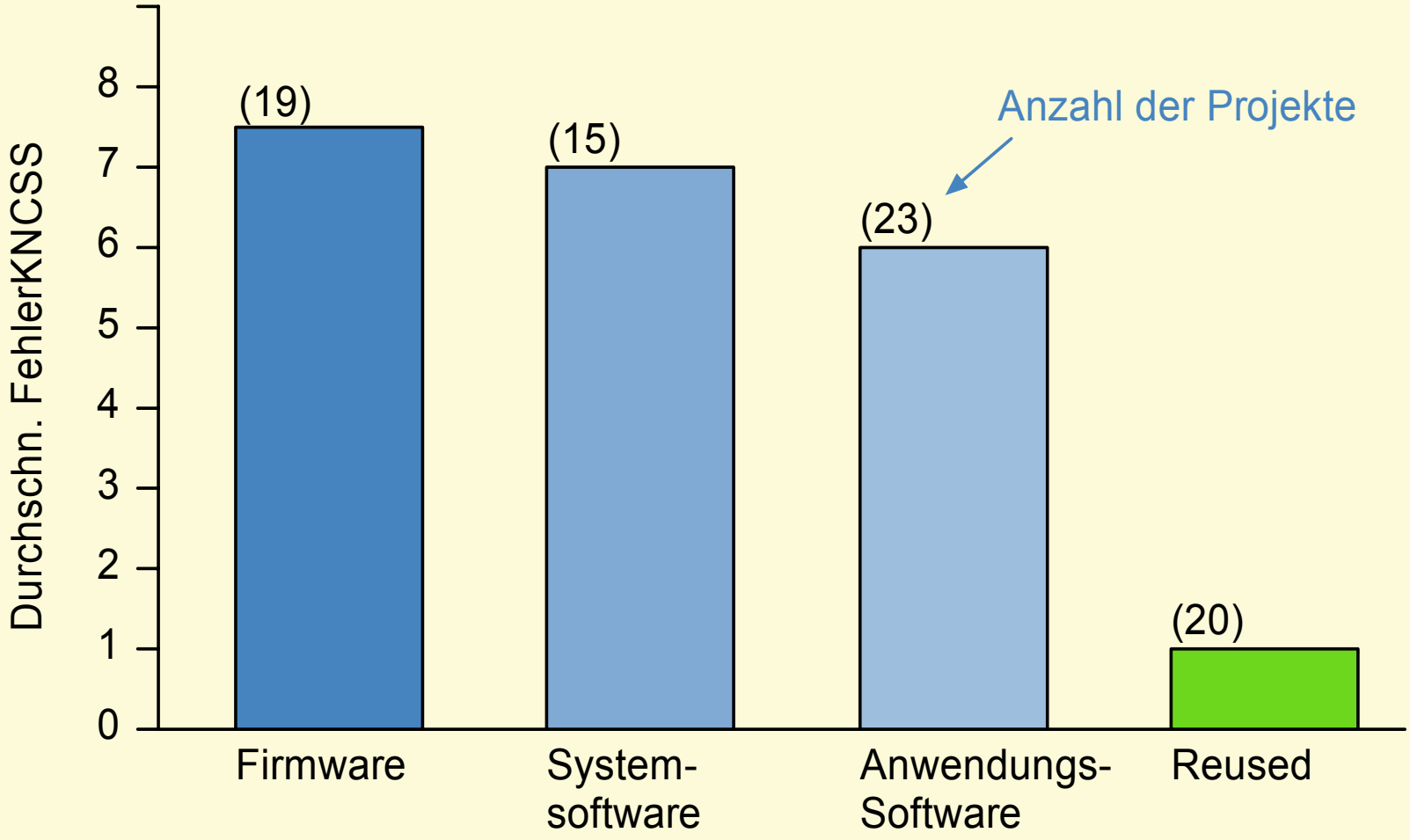
1.4 Einflußfaktoren

- **HP-Produktivität in Abhängigkeit von Anwendungsklassen**



1.4 Einflußfaktoren

- **HP-Fehlerquote vor Produktfreigabe**



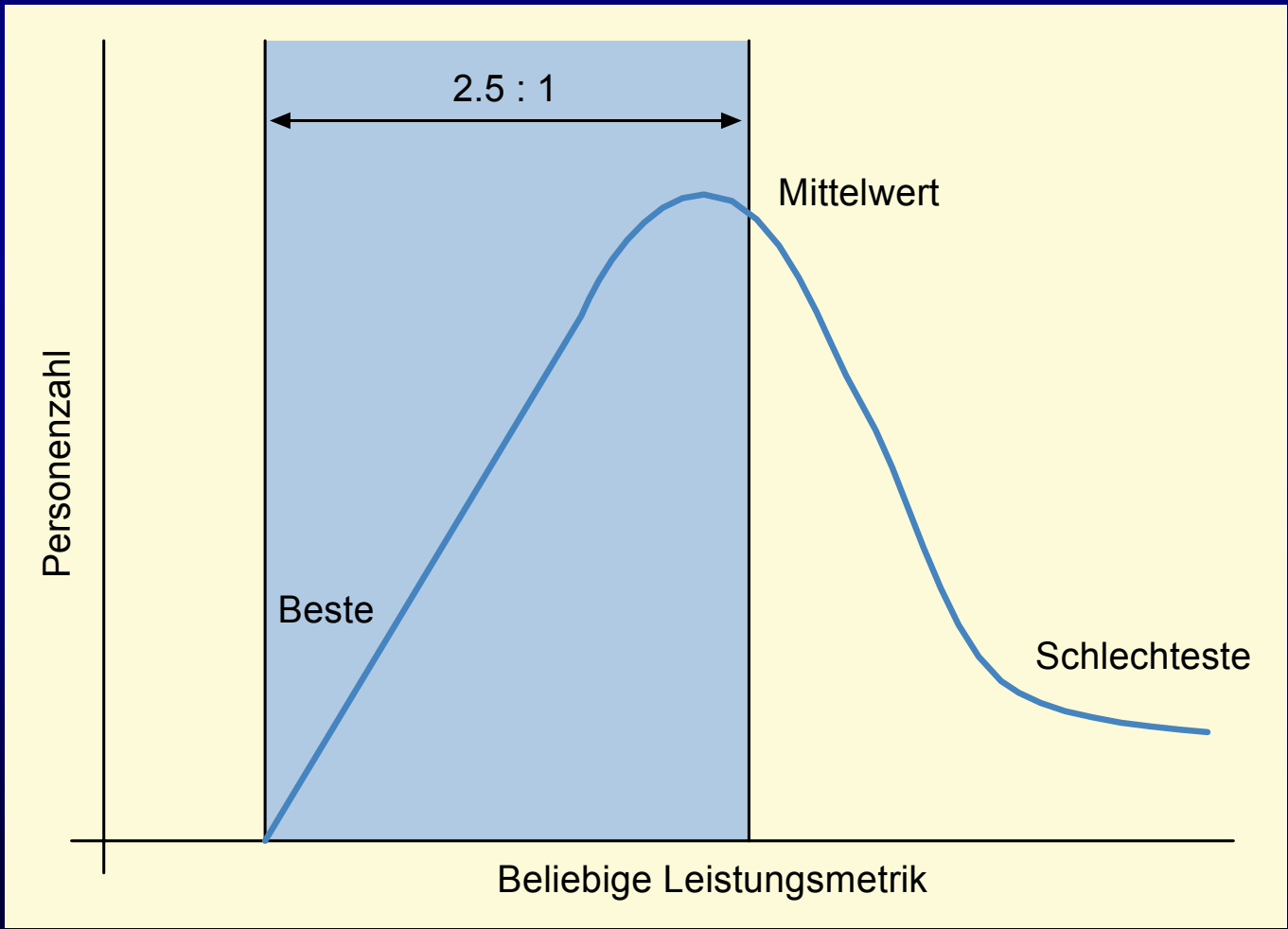
KNCSS = 1000 NCSS

1.4 Einflußfaktoren

- **Faustregel:**
  - **Projekte, die hauptsächlich aus wiederverwendeter Software bestehen, benötigen ungefähr 1/4 der Zeit und der Ressourcen, die eine Neuentwicklung benötigt.**

1.4 Einflußfaktoren

- **Mitarbeitereinflüsse**
  - **Bandbreite der individuellen Leistungen**

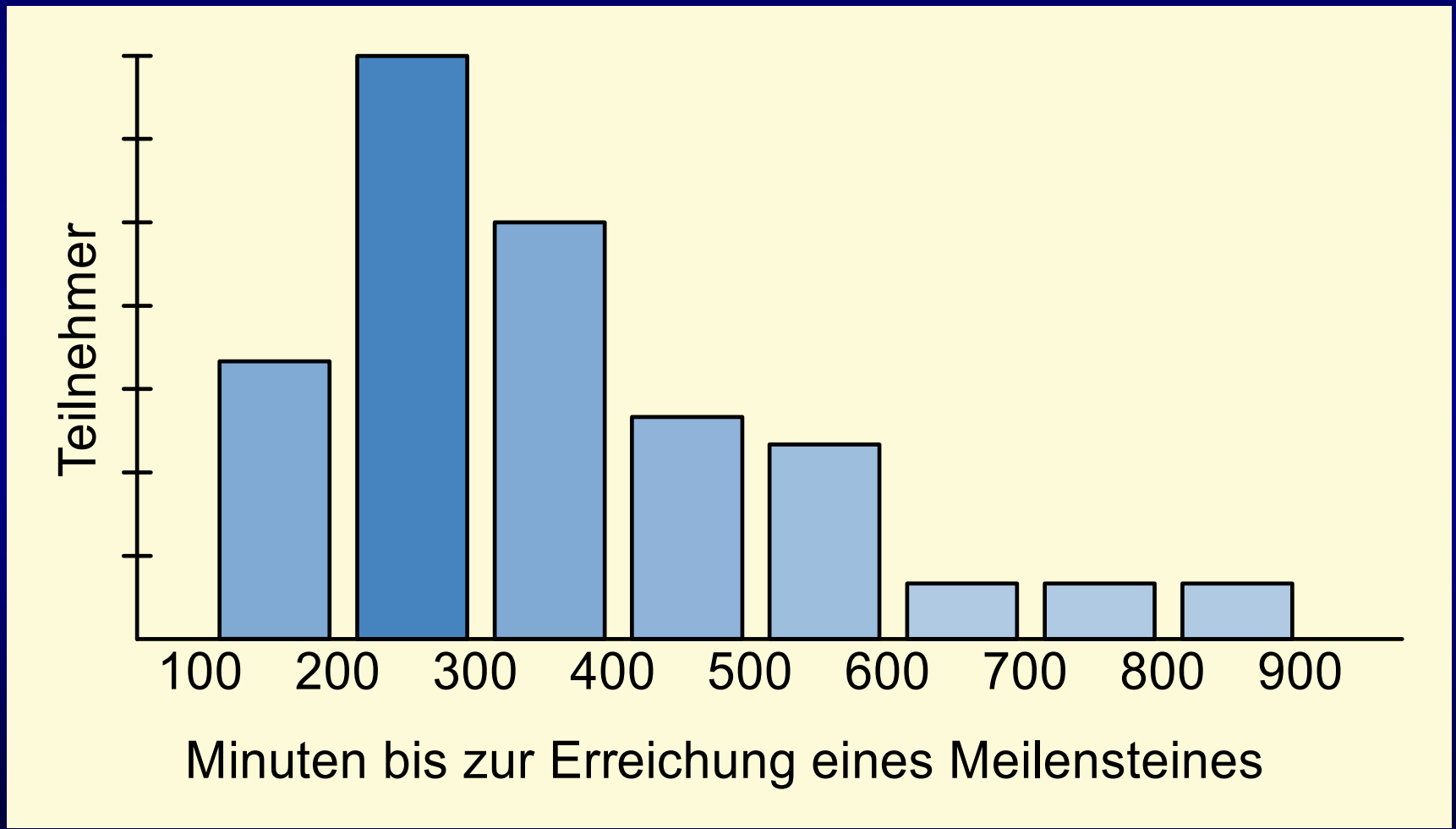


1.4 Einflußfaktoren

- Regeln
  - Die besten Mitarbeiter sind um einen Faktor 10 besser als die schlechtesten
  - Die besten Mitarbeiter sind 2,5 mal besser als der Durchschnitt
  - Die überdurchschnittlichen Mitarbeiter übertreffen die unterdurchschnittlichen Mitarbeiter im Verhältnis 2:1
- Die bessere Mitarbeiterhälfte erledigt eine Arbeit in der Hälfte der Zeit, verglichen mit der schlechteren Mitarbeiterhälfte
- Die fehleranfälligere Hälfte macht mehr als 2/3 der Fehler usw.

1.4 Einflußfaktoren

- **Ergebnisse eines Programmierwettbewerbs**



1.4 Einflußfaktoren

- Die besten Leistungen lagen um den Faktor 2,1 über dem Durchschnitt
- Die Hälfte über dem Mittelwert schlug die andere im Verhältnis 1,9 : 1
- Boehm: Produktivitätsspannbreite zwischen Mitarbeitern: Verhältnis 1 : 4,18.

1.4 Einflußfaktoren

- Welche Faktoren verursachen die Produktivitätsunterschiede?
  - DeMarco, Lister: Faktoren mit wenig oder gar keiner Korrelation mit der Leistung:
    - Programmiersprachen
    - Berufserfahrung
    - Anzahl der Fehler
    - Gehalt.

1.4 Einflußfaktoren

- Boehm: Spracherfahrung und Anwendungserfahrung wirken auf die Produktivität
  - Mit Erfahrung in der verwendeten Programmiersprache:
    - Im Verhältnis 1:1,2 produktiver
  - Mit Erfahrung auf dem Anwendungsgebiet:
    - Im Verhältnis 1:1,57 produktiver.

1.4 Einflußfaktoren

- **Managementeinflüsse**
  - **Physische Arbeitsumgebung und Arbeitsplatzausstattung:**
    - **Ein Mitarbeiter ist umso produktiver...**
      - **je ruhiger sein Arbeitsplatz ist**
      - **je weniger er gestört wird**
      - **je besser die Privatsphäre gewahrt ist**
      - **je größer der Arbeitsplatz ist.**

1.4 Einflußfaktoren

- Vergleich der Arbeitsplatzfaktoren des besten Viertels eines Programmierwettbewerbs und des schlechtesten Viertels

Arbeitsplatzfaktoren	bestes Viertel	schlechtestes Viertel
1. Wieviel Arbeitsplatz steht Ihnen zur Verfügung?	7m²	4.1m²
2. Ist es annehmbar ruhig?	57% Ja	29% Ja
3. Ist Ihre Privatsphäre gewahrt?	62% Ja	19% Ja
4. Können Sie Ihr Telefon abstellen?	52% Ja	10% Ja
5. Können Sie Ihr Telefon umleiten?	76% Ja	19% Ja
6. Werden Sie oft von anderen Personen grundlos gestört?	38% Ja	76% Ja.

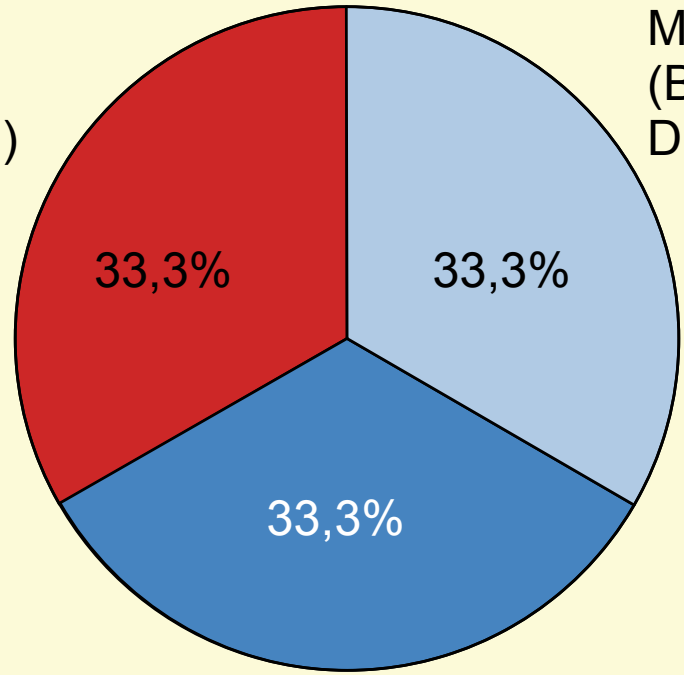
1.4 Einflußfaktoren

- **Konsequenzen:**
  - **Das Telefon eines Mitarbeiters muß für bestimmte Zeiträume abstellbar sein**
  - **Den Mitarbeitern Einzelzimmer geben**
    - **IBM:**
      - **Produktivitätssteigerung von ungefähr 11%**
    - **TRW:**
      - **Produktivitätssteigerung von ungefähr 8 %.**

1.4 Einflußfaktoren

Zeitverteilung

Projektfremde Aktivitäten (Training, Präsentationen, Reisen)



Mündliche Aktivitäten (Besprechungen, Diskussionen, Reviews)

Ruhige, konzentrierte Aktivitäten

1.4 Einflußfaktoren

- Firmenkultur
  - Das Management beeinflußt dadurch indirekt die Produktivität
    - Bei Programmierwettbewerben zwischen Firmen wurde festgestellt:
      - Die beste Organisation, d.h. diejenige, die den besten Durchschnitt aller Teilnehmer zeigte, arbeitete 11,1 mal schneller als die schlechteste
      - Zusätzlich zu dem Arbeitstempo bestanden alle Teilnehmer der besten Organisationen auch den Abnahmetest mit ihren Produkten.

1.4 Einflußfaktoren

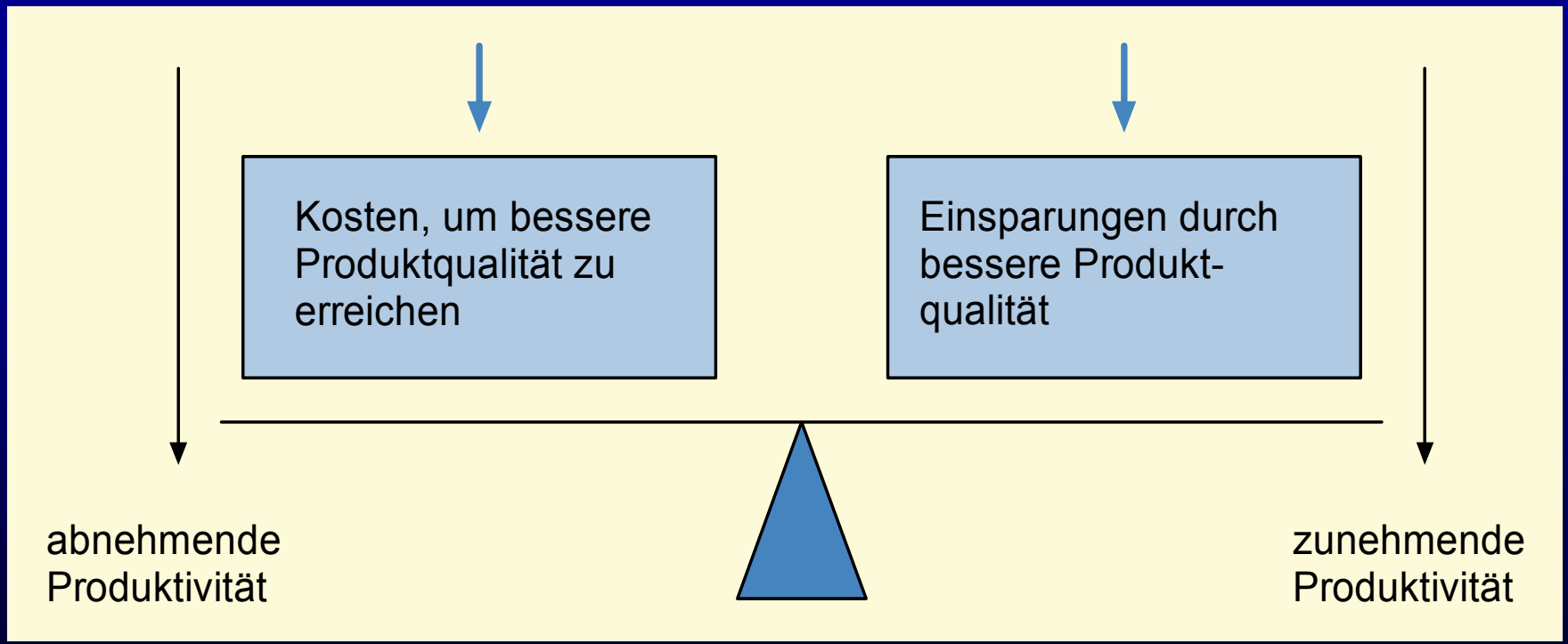
- Bei den Programmierwettbewerben bildeten je 2 Programmierer aus einer Firma ein Team
  - Sie arbeiten aber nicht zusammen, sondern konkurrieren eher miteinander, aber auch gegen alle anderen Teams
  - Die Leistungen der Teampartner lagen durchschnittlich nur um 21% auseinander
  - Dieses Ergebnis spricht dafür, daß das gleiche Umfeld und die gemeinsame Firmenkultur als Hintergrund ausschlaggebend sind.

1.4 Einflußfaktoren

- **Firmenkultur**
  - Gute Software-Ingenieure »scharen« sich in bestimmten Firmen zusammen, die schlechten in anderen
  - Bei den schlechten Firmen stimmt offensichtlich etwas in deren Arbeitsumfeld oder in deren Firmenkultur nicht
  - Gute Software-Ingenieure werden von solchen Firmen nicht angezogen oder können auf Dauer nicht gehalten werden
  - Den wenigen guten Mitarbeitern, die dort noch arbeiten, macht es das Umfeld immer schwerer, gute Ergebnisse abzuliefern
  - Firmenkultur beeinflußt Innovationseinführungen.

1.5 Produktivität und Qualität

- Hohe Qualitätsanforderungen verringern die Produktivität?
- Hohe Qualitätsanforderungen verbessern die Produktivität?

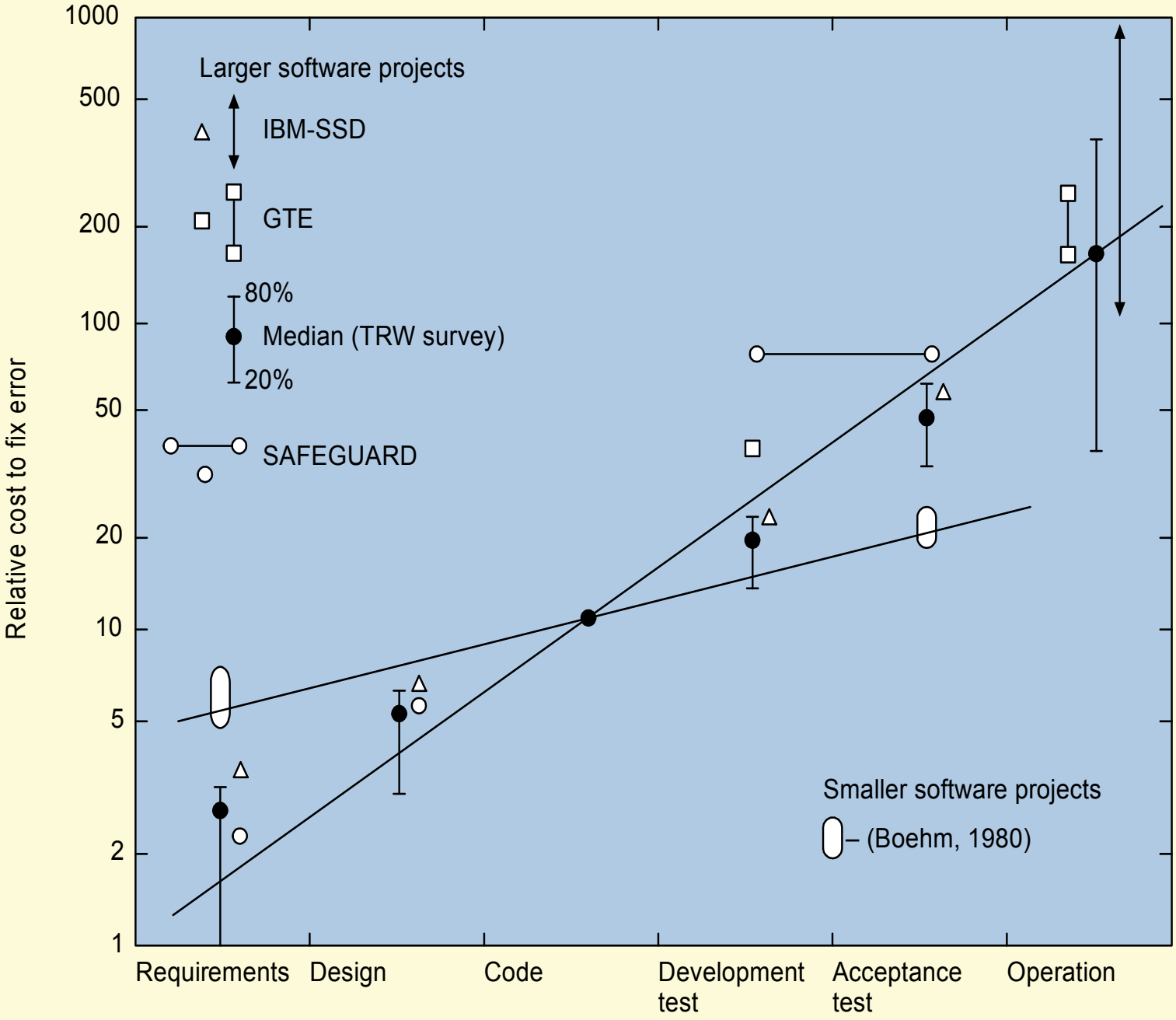


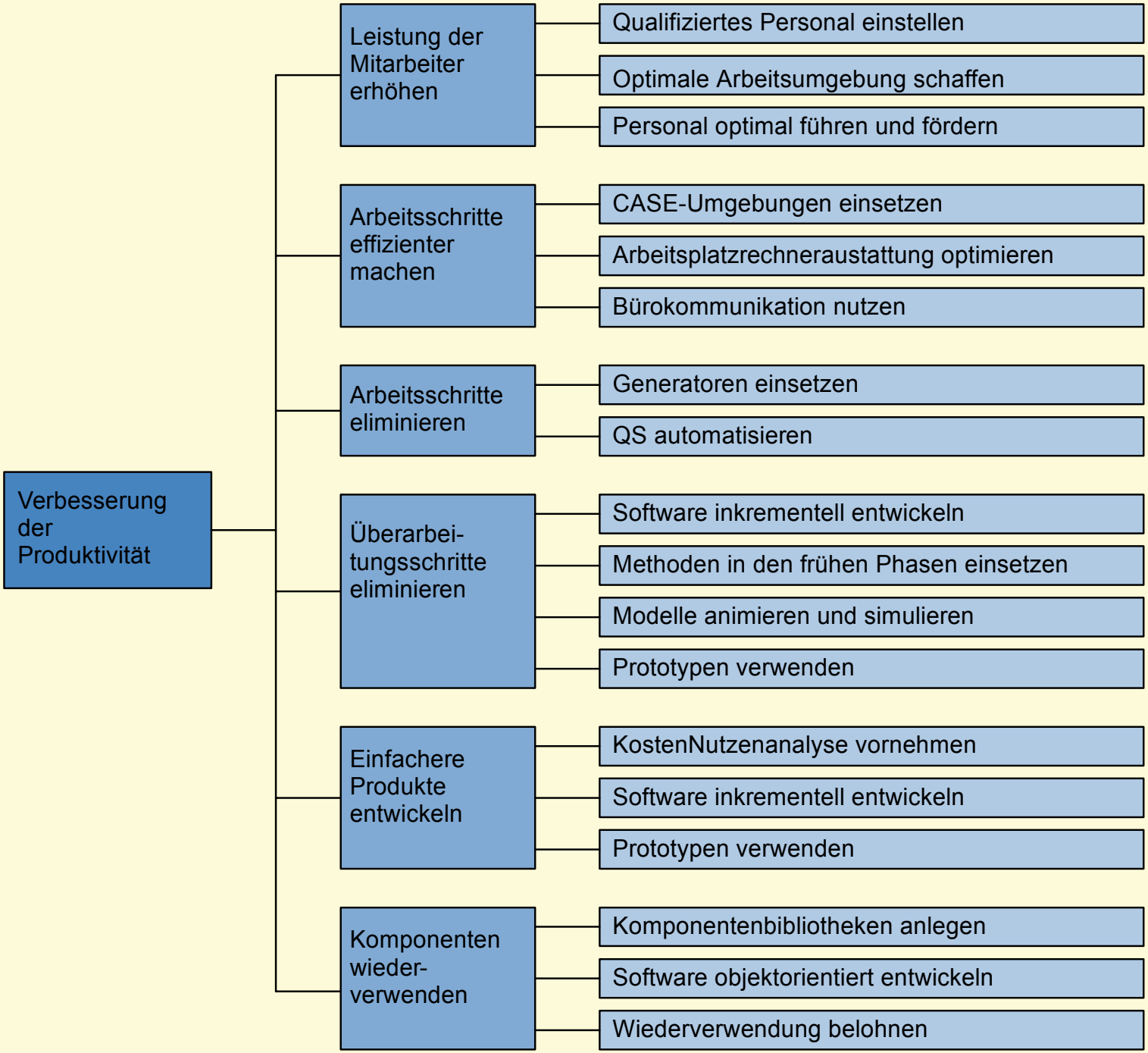
1.5 Produktivität und Qualität

- Damit die Produktivität zunimmt...
  - müssen die durch höhere Qualität bedingten Einsparungen größer sein als die zusätzlichen Kosten
- Einsparungspotential
  - Auch heute entfallen oft 2/3 aller Lebenszyklus-Kosten auf die Wartung und Pflege eines Software-Produkts
- Probleme bei den Einsparungen:
  - Wie quantifiziert man Software-Qualität?
  - Wer bezahlt die Wartung?

1.5 Produktivität und Qualität

- Probleme
  - Ist der Kunde bereit, hohe Wartungskosten zu bezahlen,dann hat der Software-Produzent kein Interesse daran, die Qualität zu verbessern
    - Oft verdient er mehr an der Wartung als am Verkauf des eigentlichen Produkts
  - Keine Trennung der Budgets für Entwicklung und Wartung
    - Der Entwicklungsleiter ist **nicht** motiviert, gute Qualität zu produzieren, da dies Kosten verursacht, er aber an den Einsparungen nicht partizipiert.


1000
500
200
100
50
20
10
5
2
1
Relative cost to fix error
Larger software projects
IBM-SSD
GTE
80%
Median (TRW survey)
20%
SAFEGUARD
Smaller software projects
– (Boehm, 1980)
Requirements
Design
Code
Development test
Acceptance test
Operation
Phase in which error was detected and corrected


Verbesserung der Produktivität
Leistung der Mitarbeiter erhöhen
Qualifiziertes Personal einstellen
Optimale Arbeitsumgebung schaffen
Personal optimal führen und fördern
Arbeitsschritte effizienter machen
CASE-Umgebungen einsetzen
Arbeitsplatzrechneraustattung optimieren
Bürokommunikation nutzen
Arbeitsschritte eliminieren
Generatoren einsetzen
QS automatisieren
Überarbei-tungsschritte eliminieren
Software inkrementell entwickeln
Methoden in den frühen Phasen einsetzen
Modelle animieren und simulieren
Prototypen verwenden
Einfachere Produkte entwickeln
KostenNutzenanalyse vornehmen
Software inkrementell entwickeln
Prototypen verwenden
Komponenten wieder-verwenden
Komponentenbibliotheken anlegen
Software objektorientiert entwickeln
Wiederverwendung belohnen

- Danke!
- Aufgaben

- Diese Präsentation bzw. Teile dieser Präsentation enthalten Inhalte und Grafiken des **Lehrbuchs der Software-Technik** (Band 2) von Helmut Balzert, Spektrum Akademischer Verlag, Heidelberg 1998