

Wirtschaftliche Betrachtung der Softwareentwicklung

Skript zum Vortrag im Kurs „Software Engineering“
bei Prof. Trautwein,
Fachhochschule Aachen
Mai 2004
Meerwais Osmani
Michael Wiedau

1. Inhaltsverzeichnis

1.	Inhaltsverzeichnis.....	2
2.	Produktivität.....	3
2.1.	Allgemeines zur Produktivität.....	3
2.2.	Definition der Produktivität.....	3
3.	Einflussfaktoren der Produktivität.....	6
3.1.	Produkteinflüsse.....	6
3.2.	Prozesseinflüsse.....	6
3.3.	Mitarbeitereinflüsse.....	10
3.4.	Managementeinflüsse.....	11
4.	Qualität und Produktivität.....	12
5.	Produktivitätssteigerung.....	13
6.	Fazit.....	14
7.	Literaturverzeichnis.....	15

2. Produktivität

2.1. Allgemeines zur Produktivität

Ein zentrales Ziel des Software-Managements besteht darin, permanent die Produktivität der Software-Erstellung zu erhöhen. Dies kann verschiedene bedeuten:

- Software-Produkte schneller, d.h. in kürzerer Kalenderzeit entwickeln. Diese Art der Produktivitätssteigerung wird oft als **Effizienzsteigerung** bezeichnet.
- Software-Produkte so zu 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.

2.2. Definition der Produktivität

Um Produktivitätssteigerungen feststellen zu können, muß die Produktivität gemessen werden können. Hierzu sind **Produktivitätsmaße** erforderlich.

Global lässt sich Produktivität folgendermaßen definieren:

1) Produktivität = Leistung / Aufwand

Es gibt aber unterschiedliche Ansätze, um Leistung und Aufwand zu definieren. Boehm definiert Leistung durch die produzierten Ergebnisse im Entwicklungsprozess.

2) Produktivität = Prod. Ergebnisse / Eingesetzter Aufwand

Sneed definiert Leistung durch die Anzahl der Software-Elemente und Aufwand durch geleistete Mitarbeitertage.

3) Produktivität = Anzahl Software-Elemente / Geleistete Mitarbeitertage

Nach diesen Definitionen kann die Produktivität verbessert werden, wenn

- die Ergebnisse vermehrt,
- der Aufwand verringert oder
- die Ergebnisse vermehrt und der Aufwand verringert werden.

Die Definitionen 2) und 3) sind problematisch. Ein Problem besteht darin, geeignete Ergebnisse bzw. Elemente zu definieren, die **quantifizierbar** sind.

Software ist ein vielfältiges Produkt. Es setzt sich möglicherweise aus mehreren **Teilprodukten** wie Programme, Benutzerhandbuch, Hilfesysteme, Testdaten etc. zusammen. Des Weiteren ist Software immaterial (kein Gewicht, keine Länge, keine Breite,...). Daher ist es eine grobe Vereinfachung, wenn Software auf ein Teilprodukt oder ein Element reduziert wird. Dennoch geschieht dies heute in aller Regel, denn eine halbe Lösung ist besser als gar keine.

Generell verwendet man im Softwarebereich die Anzahl der Quellprogrammzeilen **Line of Code (LOC)**. Ein Problem liegt darin, daß nicht alle Quellprogrammzeilen gleichwertig sind. Die Firma **HP** verwendet als Teil ihres Produktivitätsmaßes die von LOC abgeleitete Methode **NCSS (non commented source statements)**. Sie ist laut **HP** folgendermaßen definiert:

Anzahl Quellanweisungen einschließlich Compileranweisungen, Datendeklarationen und ausführbaren Anweisungen, aber ohne Leerzeilen oder Zeilen, die vollständig aus Kommentaren bestehen.

Abb. 1 zeigt eine frühe Statistik von **HP**. Die Anzahl NCSS werden durch die gesamte Projektzeit, gemessen in Ingenieurmonaten, geteilt. Jede Säule stellt die durchschnittliche Produktivität der Projekte geschrieben in der jeweiligen Programmiersprache, dar. Die Säule "Wiederverwendet" bedeutet, daß mindestens 75 Prozent des Codes bereits existierte. Die Säule "verschieden Sprachen" gibt an, daß keine Sprache mehr als 75 Prozent Codeanteil ausmacht.

Auffällig ist der Produktivitätsunterschied zwischen Projekten, bei denen ein grosser Teil der Software wiederverwendet wird gegenüber denjenigen, bei denen alles neu geschrieben wird.

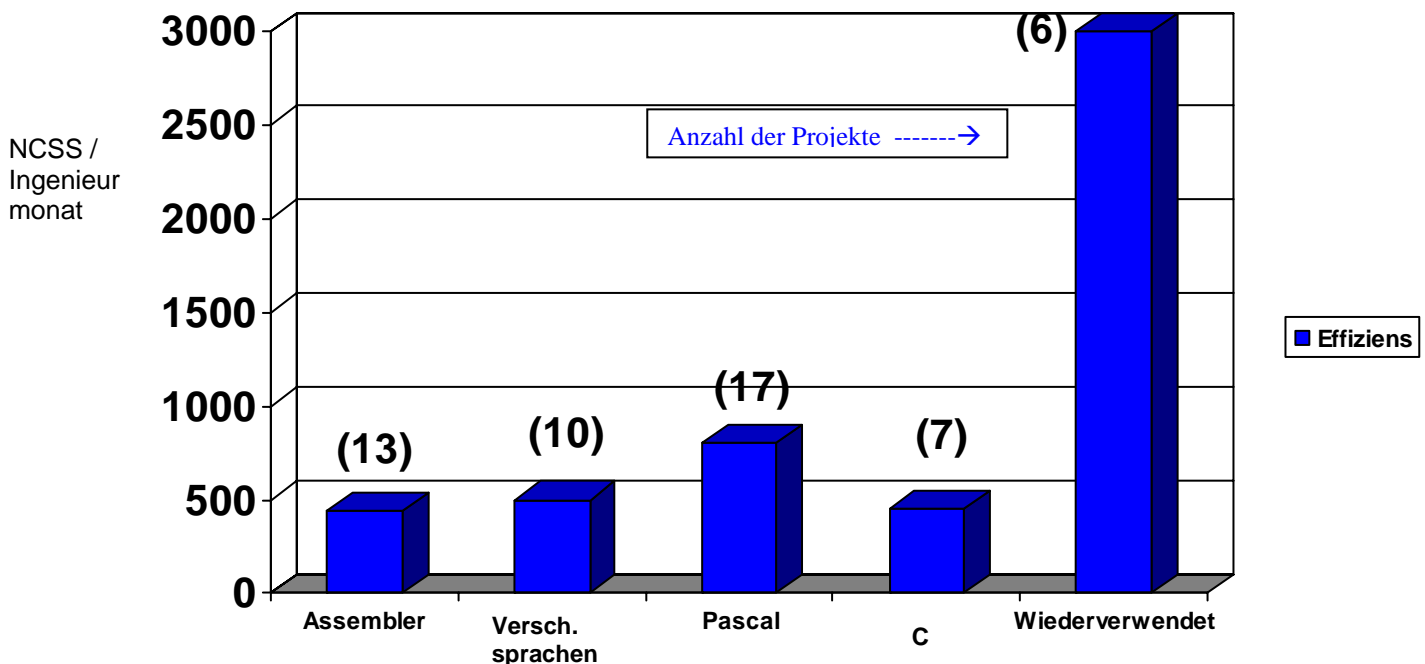


Abb. 1 Durchschnittliche NCSS/Ingenieurmonate pro Sprache

Im Gegensatz zur Definition und Messung von produzierten Ergebnissen bzw. Software-Elementen erscheint das Maß „**Eingesetzter Aufwand**“ weniger problematisch.

Es gibt diverse **Aufwandschätzmethoden**, die sich i.a. in 2 Kategorien einteilen lassen:

- **Analytisch orientierte Methoden** : Kennzeichnend für diese Methode sind theoretische Annahmen über Prozess und Produkt, von denen das Schätzergebnis im wesentlichen abhängt. Ein Beispiel hierfür ist die „**Schätzformel der Software Science**“.
- **Phänomenologisch orientierte Methoden** : Werden aufgrund empirischer Beobachtungen entwickelt. Ein prominentes Beispiel ist das von **Boehm** erfundene **COCOMO**.

Im Allgemeinen kann man sagen, dass sich der Aufwand für die Software-Entwicklung zusammensetzt aus:

- **Personalkosten**
- **Kosten für Computerressourcen**
- **Kosten für Hilfsmittel**

Den dominanten Kostenblock bilden dabei die **Personalkosten**, so dass in der Praxis Personalkosten den Aufwand repräsentieren. Hieraus folgend wird der Aufwand auf die geleistete Mitarbeitertagen fokussiert.

So hat sich aus diesen Überlegungen nach einer neuen europäischen Untersuchung folgende **beste Produktivitätsmetrik** im Softwarebereich etabliert und bewährt :

**Produktivität = Anzahl Zeilen Quellcode (LOC) / Aufwand in
Mitarbeitermonaten**

Trotz aller hier aufgezeigten Schwächen ist jeder Quantifizierungsversuch der Software-Produktivität besser als gar keiner. Denn nur wenn die Produktivität quantifizierbar und messbar ist, ist es auch möglich, das Erreichen eines Produktivitätsziels festzustellen.

3. Einflussfaktoren der Produktivität

Damit ein Software-Manager die Produktivität verbessern kann, muss er wissen, welche **Faktoren** die Produktivität am meisten beeinflussen.

3.1. Produkteinflüsse

- **Produktkomplexität:** beeinflusst wesentlich die Produktivität. Boehm gibt ein Verhältnis 1 : 2.36. Das heisst ein einfaches Produkt ist im Vergleich zu einem komplexen bis zu 2.36-fach produktiver.
- **Produktgröße:** beeinflusst ebenfalls wesentlich die Produktivität. In der Regel geht man davon aus, daß der Aufwand überproportional mit der Produktgröße (gemessen in LOC) zunimmt (die Produktivität demzufolge abnimmt).
- **Qualität:** Ein zuverlässiges Programm kann laut Boehm bis zu 1.87 Mal produktiver sein als ein unzuverlässiges.
- **Laufzeitanforderungen / Speicherbedarf:** Zunehmende Laufzeitanforderungen und zunehmende Speicherrestriktionen führen zu sinkender Produktivität

3.2. Prozesseinflüsse

Eine Software-Entwicklung wird wesentlich durch die verwendeten Methoden und die sie unterstützenden Werkzeugen beeinflusst

- **CASE-Tools:** Durch optimalen Einsatz von **CASE-Tools** kann ein Steigerungsfaktor von 1,65 erreicht werden, wobei in der Einführungsphase zunächst ein Produktivitätsverlust eintritt, später aber große Steigerungen möglich sind.

Abb. 2 zeigt eine mögliche Produktivitätssteigerung durch den Einsatz von CASE-Werkzeugen.

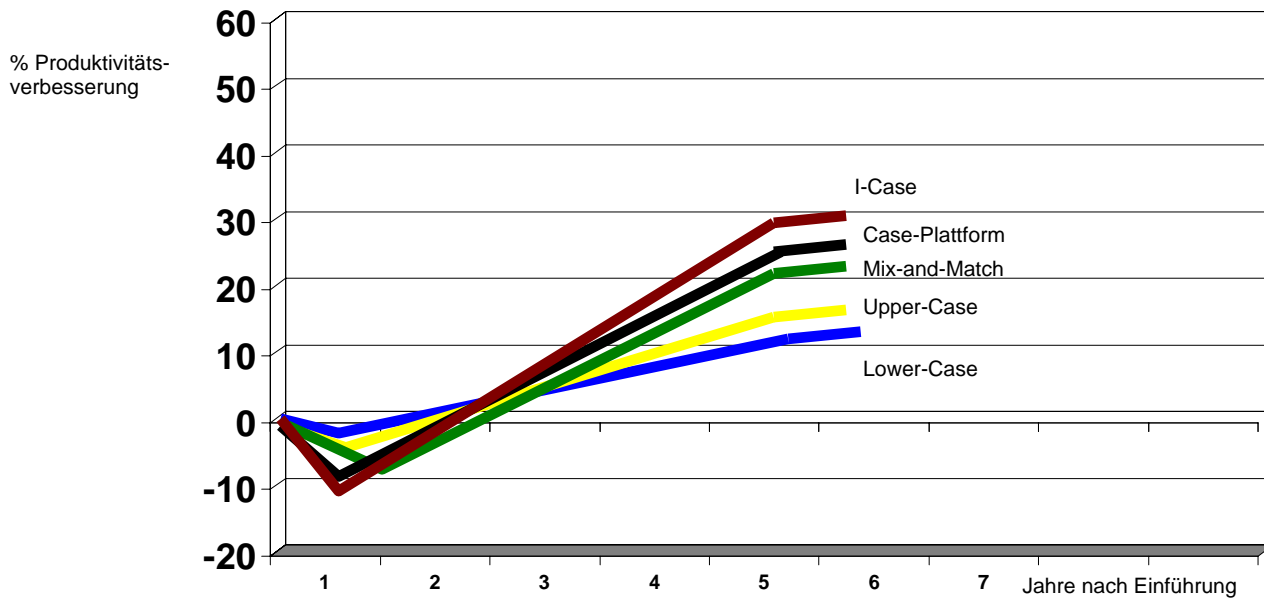


Abb. 2 Produktivitätsverlauf in Abhängigkeit vom CASE-Umfang

- I-CASE

Werkzeugketten, die die gesamte Entwicklung unterstützen.

- CASE-Plattform

Rahmensystem, das Basisleistungen abdeckt und das Einbetten von Werkzeugen erlaubt.

- Mix-and-Match

Einsatz von verschiedenen Werkzeugen in den verschiedenen Phasen.

- Upper-CASE

Werkzeuge für Planung, Definition und Entwurf.

- Lower-CASE

Werkzeuge für Implementierung, Wartung und Pflege.

Auffällig hierbei ist, dass je intensiver und länger CASE-Werkzeuge in ein Projekt involviert sind, desto höhere Produktivität möglich ist.

- **Stabile Anforderungen** ermöglichen laut Böhm eine Produktivitätssteigerung vom Faktor 1.78.
- **Wiederverwendung:** Eine sehr wichtige Rolle bei der Produktivitätssteigerung spielt auch die Wiederverwendung von Software. Bei **HP** gibt es folgende Definition dafür:

„Software, die in ein Produkt eingebracht wird und vorher in einem anderen Produkt oder einem anderen Teil desselben Produkts einwandfrei arbeitete.“

Abb. 3 zeigt die HP-Produktivität in Abhängigkeit von Anwendungskategorien. Auffällig ist, dass die mit Abstand größte Steigerung mit wieder verwendeter Software möglich ist. Die Säule „Wiederverwendet“ bedeutet wiederum, daß mindestens 75 Prozent des Codes bereits existierte.

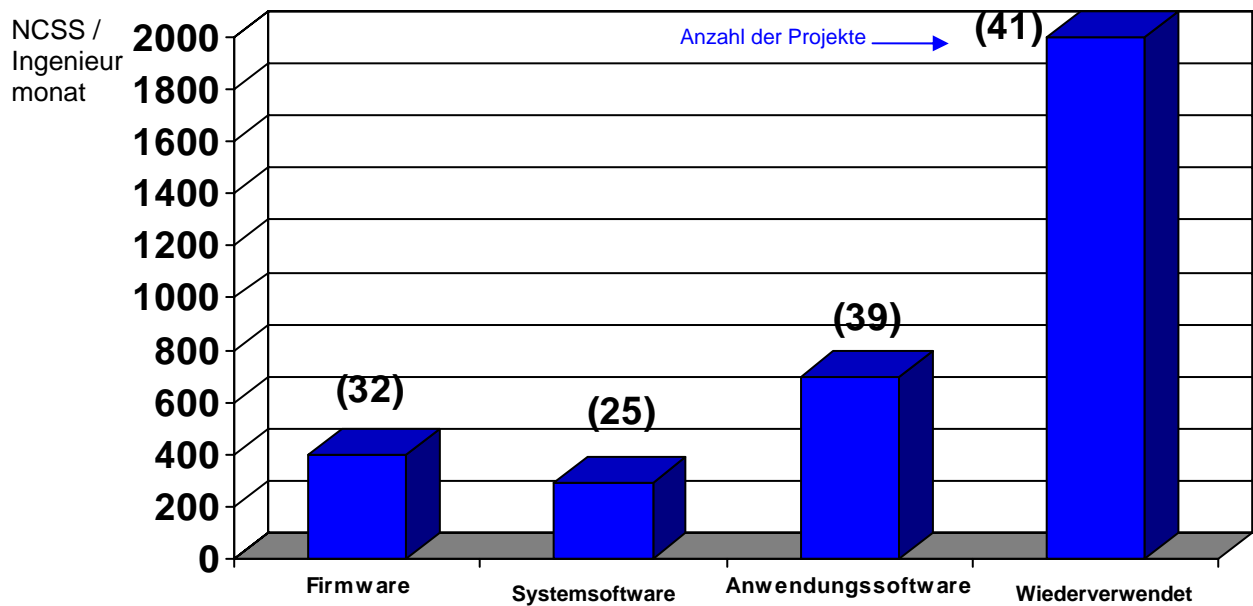


Abb. 3 HP-Produktivität in Abhängigkeit von Anwendungskategorien

Abb. 4 zeigt die HP-Fehlerquote in Abhängigkeit von Anwendungskategorien. Auch hier fällt auf, dass die Fehlerquote bei wieder verwendeter Software am geringsten ist.

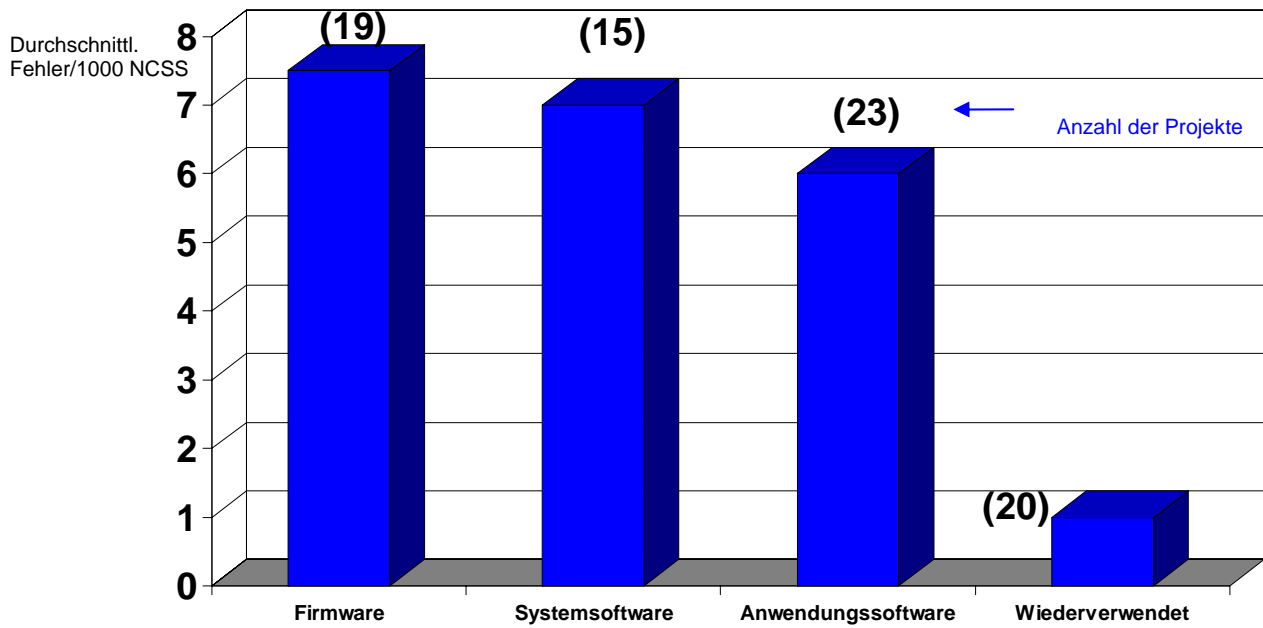
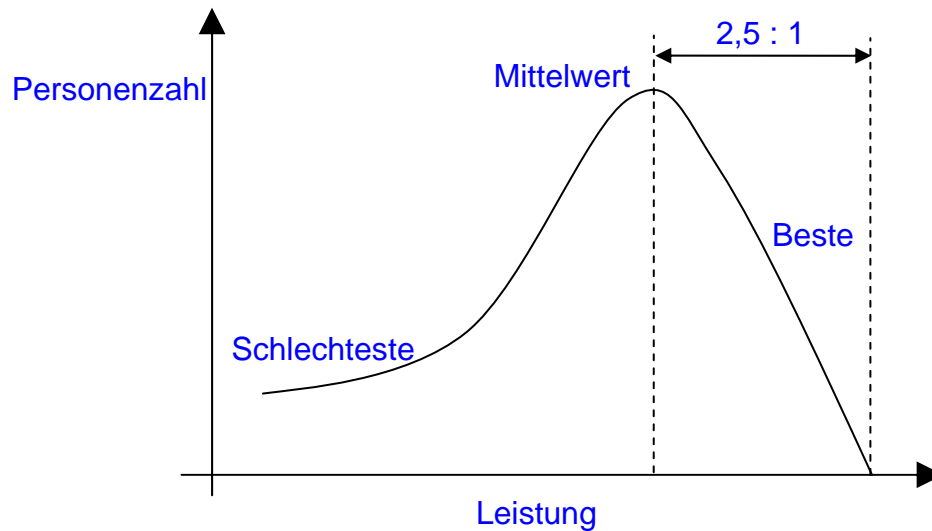


Abb. 4 HP-Fehlerquote in Abhängigkeit von Anwendungskategorien

3.3. Mitarbeitereinflüsse

Um festzustellen wie produktiv Mitarbeiter sind hat man empirische Untersuchungen zusammengetragen. Diese empirischen Untersuchungen sind in der unteren Grafik verdeutlicht.

Auf der Unteren Achse wurde die Leistung der Mitarbeiter eingetragen, auf der linken Achse die Anzahl der Mitarbeiter die diese Leistung erbringen.



Es ist dabei sehr interessant, dass die Besten 2,5-mal so gut sind, wie der Durchschnitt. Sie erbringen 10-mal soviel Leistung wie die Schlechtesten.

Der Anstieg der Leistung ist sehr stark. Das bedeutet, dass es nur wenige sehr gute Mitarbeiter gibt und die meisten Mitarbeiter im Mittelfeld liegt.

Die bessere Hälfte der Mitarbeiter erledigt die gleiche Arbeit in der Hälfte der Zeit die die schlechtere Hälfte braucht.

Dabei gibt es verschiedene Faktoren die die Mitarbeiter unterscheidet:

- **Programmiersprachen**
Je nach dem wie viele Programmiersprachen und vor allem welche prinzipiellen Grundkenntnisse über Programmiersprachen der Mitarbeiter hat, desto besser kann er als Programmierer eingesetzt werden.
- **Berufserfahrung**
Die Berufserfahrung spielt eine wichtige Rolle um die Leistung des Mitarbeiters einschätzen zu können. Heute ist vor allem Projekterfahrung und Teamkompetenz sehr wichtig.
- **Fehleranzahl**
Die Anzahl der Fehler die ein Programmierer macht spielt laut empirischer Untersuchungen eine nicht so eine grosse Rolle wie viele Firmen glauben. Allerdings kann auch dieser Faktor die Zeitspanne die ein Projekt braucht beeinflussen.
- **Gehalt**
Höhere Gehälter tragen nur für kurze Zeit zur Leistungssteigerung der Mitarbeiter bei.

3.4. Managementeinflüsse

Grossen Einfluss auf die Produktivität der Mitarbeiter hat der Arbeitsplatz. Ein optimaler Arbeitsplatz sollte folgende grundlegende Kriterien erfüllen:

1. **Ruhe am Arbeitsplatz** (kein Maschinenlärm, etc.)
2. **wenig Störungen** (Telefon, etc.)
3. **gute Privatsphäre**
4. **Größe des Arbeitsplatzes**

Untersuchungen haben gezeigt, dass ein Mitarbeiter ca. **15 Minuten** braucht um „in Fahrt“ zu kommen. Innerhalb dieser Zeit sollten **keine plötzlichen Störungen** durch Mitarbeiter eintreten. Das Telefon des Mitarbeiters sollte die Möglichkeit bieten stumm geschaltet werden zu können. Die Mitarbeiter sollten daher **Einzelzimmer** haben.

Diese Maßnahmen haben bei IBM zu einer Leistungssteigerung von ca. **11%** geführt.

In manchen Firmen mit Großraumbüros lässt man Musik laufen um den Geräuschpegel zu überspielen. Musik wird in der rechten Gehirnhälfte wahrgenommen.

Sequentielle Standard-Vorgänge die beim Programmieren vielfach vorkommen werden in der linken Gehirnhälfte vorgenommen. So genannte Aha-Erlebnisse und kreative Erkenntnisse kommen aus der rechten Gehirnhälfte. Wenn die rechte Gehirnhälfte nun damit beschäftigt ist Musik wahrzunehmen, so bleiben diese Erkenntnisse aus.

Gute Programmierer versammeln sich meistens in bestimmten Firmen. Die schlechten Programmierer in anderen. Wenn gute Mitarbeiter dennoch in einer schlechten Firma arbeiten, dann sinkt deren Produktivität.

Die **Teamgröße** hat einen wichtigen Einfluss auf die Produktivität der Mitarbeiter.

Wenn das Team zu groß wird, dann sinkt die Produktivität.

Frei nach dem Motto „zu viele Köche verderben den Brei“.

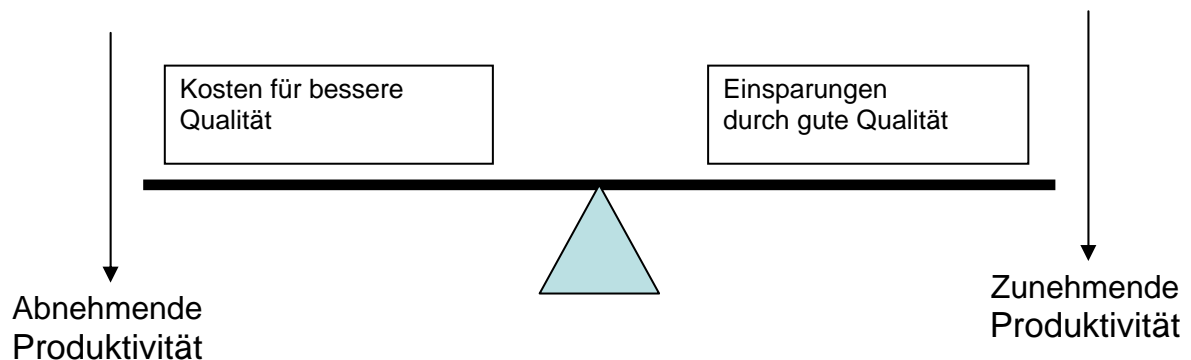
4. Qualität und Produktivität

Es gibt zwei unterschiedliche Meinungen zum Zusammenhang zwischen Qualität und Produktivität.

Hohe Qualitätsanforderungen

1. verringern die Produktivität
2. verbessern die Produktivität

Konstruktive und analytische Qualitätssicherungsmaßnahmen verursachen Kosten. Folgendes Bild soll den Zusammenhang zwischen Kosten, Qualität und Produktivität verdeutlichen:



Ein optimales Ergebnis kann nur dann erreicht werden, wenn die Kosten und die Einsparungen sich die Waage halten. Wenn zu hohe Einsparungen getan werden, so leidet darunter die Qualität. Wenn die Einsparungen größer sind als die Kosten so hat dieses allerdings eine Zunahme der Produktivität zur Folge.

Das Einsparungspotenzial durch gute Qualität ist sehr hoch.

Ca. 2/3 aller Kosten sind Wartungskosten. Nur 1/3 entfällt auf die eigentlichen Entwicklungskosten. Eine Umfrage bei HP hat sogar einen Kostenanteil der Wartung von 75% ergeben.

Softwarequalität ist sehr schlecht quantitativ messbar. Dabei sind die Wartungskosten ein möglicher Ansatz um die Qualität der Software zu bestimmen. Ist ein Kunde bereit sehr hohe Wartungskosten zu bezahlen, so hat der Produzent keinen Anlass die Qualität seines Produktes zu verbessern.

In manchen Firmen werden die Kostentöpfe für Entwicklung und Wartung getrennt. Dabei kann es allerdings passieren, dass der Entwicklungsleiter schlechtere Qualität liefert um Kosten einzusparen. Damit entstehen bei der Wartungsabteilung sehr hohe Kosten und die Qualität der Software kann nur sehr langsam verbessert werden.

5. Produktivitätssteigerung

Um die Produktivität einer Firma zu steigern gibt es sechs Kategorien:

- 1. Leistung der Mitarbeiter erhöhen**
 - Qualifiziertes Personal einstellen
 - Optimale Arbeitsumgebung schaffen
 - Personal optimal führen und fördern

- 2. Arbeitsschritte effizienter machen**
 - CASE-Umgebungen einsetzen
 - Arbeitsplatzrechnerausstattung optimieren
 - Bürokommunikation nutzen

- 3. Arbeitsschritte eliminieren**
 - Generatoren einsetzen
 - QS automatisieren

- 4. Überarbeitungsschritte eliminieren**
 - Software aufeinander aufbauend entwickeln
 - Sehr früh schon Methoden der SWE einsetzen
 - Fertige Modelle simulieren
 - Prototypen verwenden

- 5. Einfachere Produkte entwickeln**
 - Kosten- /Nutzenanalyse vornehmen
 - Software aufeinander aufbauend entwickeln
 - Prototypen verwenden

- 6. Komponenten wieder verwenden**
 - Komponentenbibliotheken anlegen
 - Software Objektorientiert entwickeln
 - Wiederverwendung belohnen

6. Fazit

Lines of Code (LOC) ist die bewährteste Methode um die Leistung von Mitarbeitern zu messen. Der Aufwand wird für gewöhnlich in Mitarbeitermonaten angegeben.

Es gilt folgende grundlegende Formel:

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

Die wichtigsten Einflüsse auf die Wirtschaftlichkeit / Produktivität sind:

- **Produkteinflüsse**
- **Prozesseinflüsse**
- **Mitarbeitereinflüsse**
- **Managementeinflüsse**

Steigerung der Wirtschaftlichkeit durch folgende Kategorien:

1. **Leistung der Mitarbeiter erhöhen**
2. **Arbeitsschritte effizienter machen**
3. **Arbeitsschritte eliminieren**
4. **Überarbeitungsschritte eliminieren**
5. **Einfachere Produkte entwickeln**
6. **Komponenten wieder verwenden**

7. Literaturverzeichnis

1. Helmut Balzert, *Lehrbuch der Softwaretechnik*, Spektrum akademischer Verlag, 1998
2. Dr. Dieter Koreimann. *Grundlagen der Softwareentwicklung*. 2. Auflage, Oldenbourg Verlag, 1995
3. Tagungsband, *Diskussionsforum Fachhochschule '92*, Wirtschaftsinformatik Morgen: Prinzipien strategischer Softwareentwicklung. Darmstadt, 12./13.11.1992, Arbeitskreis Wirtschaftsinformatik an Fachhochschulen