



Media Engineering

Shit Happens

...aber nicht notwendigerweise



R. Weller

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Murphy's law

- “Anything that can go wrong, *will* go wrong.”
- Dies gilt auch (oder insbesondere) für Projekte mit IT-Bezug



Edward A. Murphy

Beispiel F-16

- Bei Überquerung des Äquators brachte der Autopilot das Flugzeug in Rückenlage
- Ursache: Man hatte keine „negativen“ Breitengrade berücksichtigt
- Fehler wurde erst sehr spät im Simulator entdeckt



Beispiel: Ariane 5

- Am 4. Juni 1996 startete die ESA eine Rakete von Französisch Guyana aus.
- Vierzig Sekunden nach dem Start explodierte die Rakete.
- Verlust ca. 500 Millionen \$ für Rakete und Ladung (vier Satelliten).
- Entwicklungskosten ca. 7 Milliarden \$.
- Ursache:
 - Umwandlung einer 64-Bit Gleitkommazahl in 16-Bit signed Integer
 - Die Zahl war die Horizontalgeschwindigkeit und wurde von der langsameren Ariane 4 übernommen



Beispiel: Therac-25

- Therac-25 war ein Gerät für die Strahlentherapie
- Der Computer wurde gleichzeitig für die Dateneingabe und die Steuerung des Geräts zuständig
- Bei Korrektur wurden Eingaben manchmal nicht korrekt übernommen
- Mehrere Patienten wurden schwer verletzt, zwei starben sogar

| | | | |
|---------------------------|--------------|--------------|------------------------------------|
| PATIENT NAME | : TEST | BEAM TYPE: X | ENERGY (MeV): 25 |
| TREATMENT MODE | : FIX | | |
| | | ACTUAL | PRESCRIBED |
| UNIT RATE/MINUTE | | 0 | 200 |
| MONITOR UNITS | | 50 50 | 200 |
| TIME (MIN) | | 0.27 | 1.00 |
| GANTRY ROTATION (DEG) | | 0.0 | 0 VERIFIED |
| COLLIMATOR ROTATION (DEG) | | 359.2 | 359 VERIFIED |
| COLLIMATOR X (CM) | | 14.2 | 14.3 VERIFIED |
| COLLIMATOR Y (CM) | | 27.2 | 27.3 VERIFIED |
| WEDGE NUMBER | | 1 | 1 VERIFIED |
| ACCESSORY NUMBER | | 0 | 0 VERIFIED |
| DATE | : 84-OCT-26 | SYSTEM | : BEAM READY OP. MODE : TREAT AUTO |
| TIME | : 12:55: 8 | TREAT | : TREAT PAUSE X-RAY 173777 |
| OPR ID | : T25V02-R03 | REASON | : OPERATOR COMMAND: |

Beispiel: Arbeitsagentur für Arbeit

- Kurz vor Inkrafttreten der Arbeitsmarktreform 2004 warteten zahlreiche Arbeitslose umsonst auf die Überweisung der Arbeitsagentur
- Ursache: 10-stellige Maske für Kontonummern.
- Manche Banken haben aber kürzere Kontonummern
- Leere Felder werden anstatt am Anfang *am Ende* der Kontonummern automatisch mit Nullen aufgefüllt

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 7 | 8 | 6 | 5 | 9 | 3 | 1 | 6 | 0 |
|---|---|---|---|---|---|---|---|---|---|

Beispiel: The Fappinging

- FAZ 1.9.14: Ein Hacker hat Nacktbilder von mehr als 100 Prominenten im Internet veröffentlicht.
- Betroffen: Nutzer von iOS-Systemen
- Der Hacker, ist an die Passwörter ihrer iPhones gelangt, indem er automatisiert zahllose Passwörter nacheinander ausprobierte (Brute-Force-Methode mittels eines Python-Skriptes).
- Über die Programmierschnittstelle des Services „Find my iPhone“ konnte er unbegrenzt Passwortabfragen tätigen. In der Regel tritt nach mehreren fehlgeschlagenen Versuchen eine Sperre in Kraft.
- Bei „Find my iPhone“ vergaß Apple das allerdings.



Beispiel: Schufa

- 2015 verkündete die Wirtschaftsauskunftei zukünftig soziale Netzwerke wie Facebook zur Bewertung heranzuziehen
- Realisierung in Forschungsprojekt in Zusammenarbeit mit dem Hass-Plattner-Institut
- Gestoppt nach Protesten von Verbraucherschützern



- Der Fußballverein KSC wollte ein Verfahren zur Gesichtserkennung im Stadion Testen
- Entwickelt in einem Forschungsprojekt am Karlsruher Instituts für Technologie (KIT)
- Geplanter Test bei drei Heimspielen wurde nach Fanprotesten abgesagt



Beispiel: Duke Nukem Forever

- Erstmals angekündigt: 1997
- 1998: Wechsel von Eigenentwicklung zur Quake2-Engine
- 2000: Wechsel zur Unreal-Engine
- 2009: Nach mehreren weiteren Engine-Wechseln stellt 3D Realms die Entwicklung ein und entlässt zahlreiche Mitarbeiter
- 2010: Weitergabe der Entwicklung an Gearbox
- 2011: Veröffentlichung und sehr durchwachsener Erfolg



Beispiel: Toll Collect

- Software zur Einnahme der LKW-Maut
- Entwicklungszeit: > 3 Jahre
- Kosten:
- Verzögerung der Einführung um 16 Monate
- Heutiger Streitwert ca. 7.500.000.000 €



- Die BBC wollte 2008 alle Radio- und TV-Sender auf eine gemeinsame digitale Produktionsplattform umstellen
- Archivierung, Verarbeitung, Übertragung und Verteilung über IT-Infrastruktur
- Digitalisierung des kompletten Archivmaterials
- Zunächst beauftragung von Siemens IT Solutions, ab 2010 Eigenentwicklung
- Einstellung 2013
- Kosten: 100 Millionen Pfund

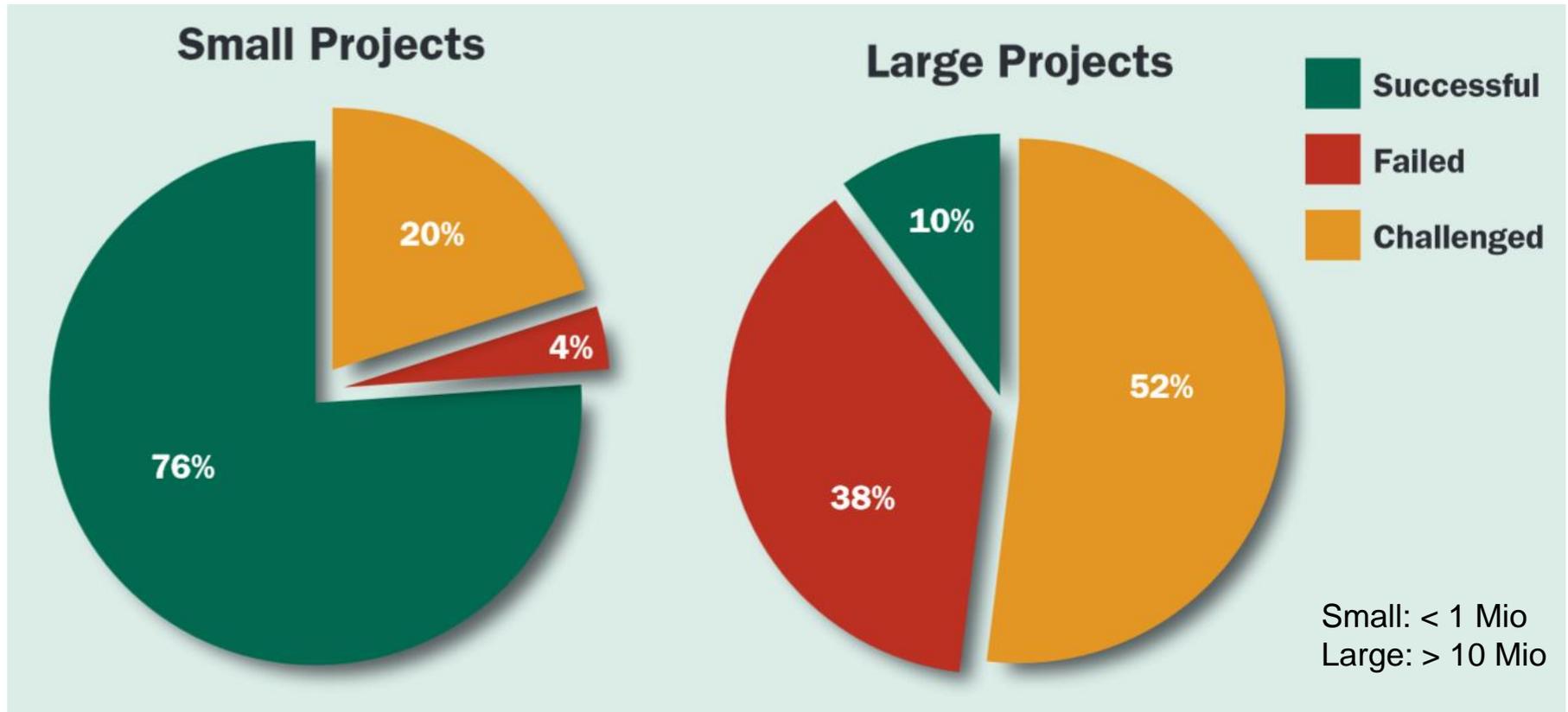


Hall of Shame (sehr unvollständige Auswahl)

| Projektname/Beteiligte | Dauer | Ende | Kosten |
|---|-----------|------|-------------------------------|
| Passion for Performane Otto und SAP | 3 Jahre | 2012 | Zweistelliger Millionenbetrag |
| Elektronische Gesundheitskarte | >14 Jahre | ? | Bislang > 700 Millionen |
| Modesta Strafjustiz in Berlin | 4 Jahre | 2010 | 8,5 Millionen |
| MyCalPays SAP und Kalifornien | 7 Jahre | 2012 | \$254 Millionen |
| Elena Verschiedene Bundesministerien | 8 Jahre | 2010 | >188 Millionen |
| DaZu Bundesamt für Umwelt, CH | 3 Jahre | 2012 | 6 Millionen Franken |
| Insieme, Eidgenössische Steuerverwaltung, CH | 7 Jahre | 2012 | 124 Millionen Franken |
| NHS Connecting for Health UK Department of Health | 7 Jahre | 2013 | 12 Milliarden Pfund |

Scheitern alle IT-Projekte?

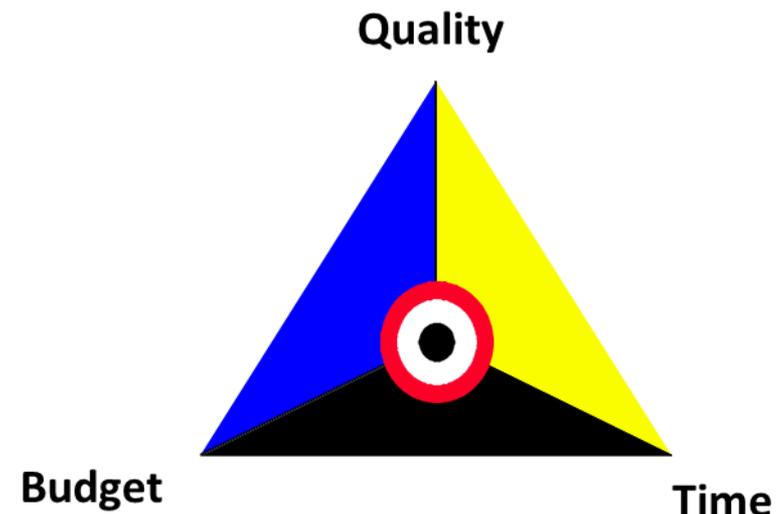
- Fast. Tendentiell: Je größer das Projekt, desto höher die Wahrscheinlichkeit des Scheiterns



Chaos Manifesto 2013, Standish Group

Was bedeutet scheitern?

- Ein Projekt welches Ressourcen verbraucht bezeichnen wir als gescheitert falls
 - Es nicht wie versprochen ausgeliefert wird
 - Mehr kostet als veranschlagt
 - Länger bis zur Fertigstellung braucht als geplant
 - Nicht den Spezifikationen entspricht

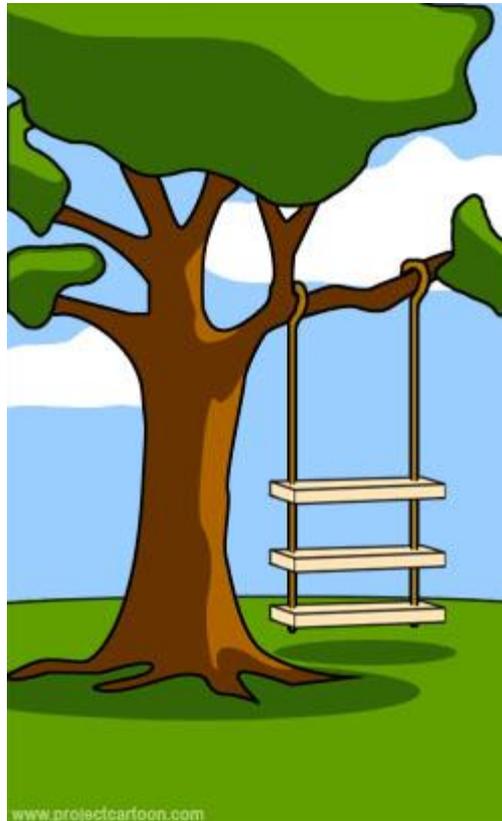


Gründe für das Scheitern von IT-Projekten

- Unrealistische oder undeutlich formulierte Projektziele
- Falsche Einschätzung der notwendigen Ressourcen
- Schlecht definierte Anforderungen an das System
- Kaum Risikomanagement
- Fehlende Kommunikation der Entwickler mit dem Kunden
- Fehlende Dokumentation
- Verwendung unfertiger Technologien
- Unterschätzung der Komplexität
- Schlampige Programmierung
- Entscheidungen der Eigentümer
- Kommerzieller Druck
- Änderung der Gesetzeslage
- Unterschätzen der Öffentliche Meinung
- **Schlechtes Projektmanagement**

Herausforderung: Kommunikation

- An einem Projekt können sehr viele Menschen mit unterschiedlichem Hintergrund beteiligt sein:
 - Kunde
 - Aussendienst
 - Projektleiter
 - Designer
 - Entwickler
 - Marketing
 - Controlling
 - Support
 - ...
- Diese verstehen oft unter den gleichen Begriffen etwas völlig unterschiedliches



Wie der Kunde es erklärt hat



Wie der Aussendienst es versprochen hat



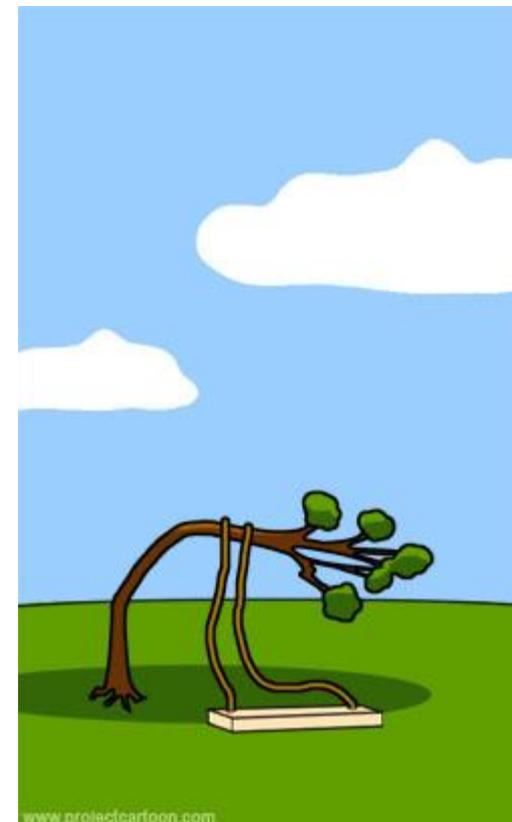
Wie der Designer es entworfen hat



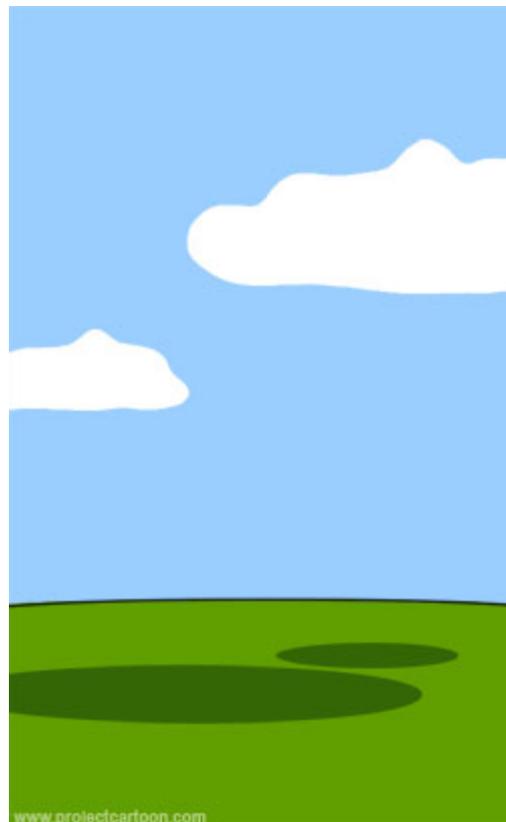
Wie der Projektleiter
es verstanden hat



Wie der Programmierer
es programmiert hat



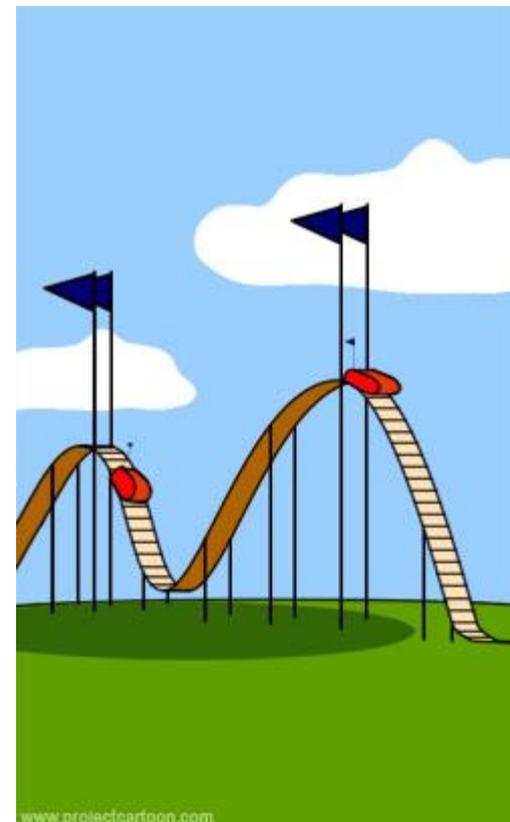
Der Notfallplan



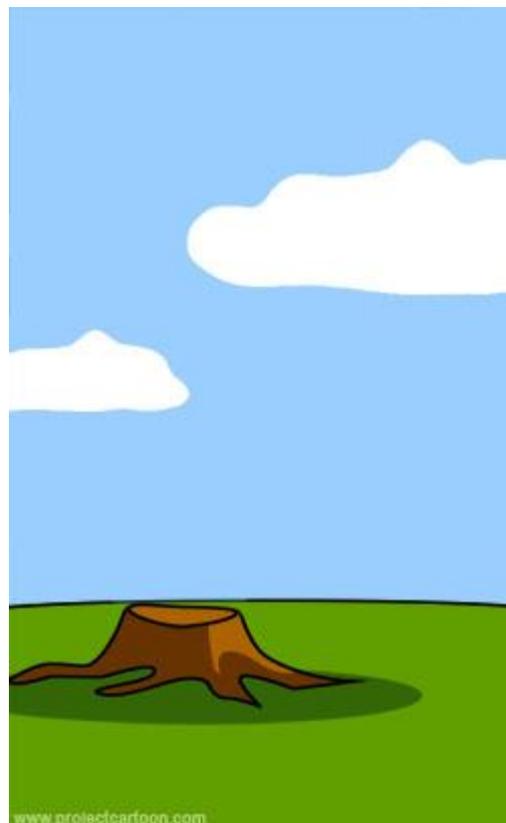
Wie das Projekt dokumentiert wurde



Was beim Kunden installiert wurde



Wie die Rechnung für den Kunden aussieht



Wie der Support aussah



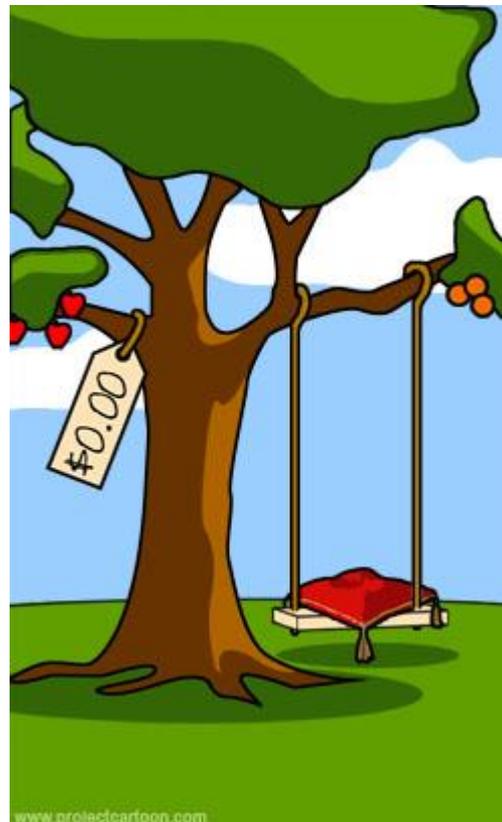
Wie Patches ausgeliefert wurden



Der Zeitpunkt der Auslieferung



Was der Kunde eigentlich braucht



Wie der Kunde es sich vorgestellt hat



Wie die Marketing-
abteilung es bewarb

Scheitern nur IT-Projekte?

- Scheitern ist kein spezifisches Problem von IT-Projekten



- Aber Software hat einige Eigenschaften, die das Scheitern besonders unterstützen



Was ist Software überhaupt?

- Software besteht aus:
 - Einem ausführbaren **Programm** (aber nicht nur)
 - Der dazugehörigen **Dokumentation**
 - Dazu gehört sowohl die (interne) Dokumentation des Systems, die die Struktur des Systems im Detail erklärt
 - Aber auch das Benutzerhandbuch, welches erklärt, wie das System verwendet wird
 - Und **Daten**
 - Z.B. Konfigurationsdateien
 - Datenbanken
 - 3D Modellen und Texturen (bei Spielen)
 - ...

- Eigenständige Programme (Office-Anwendungen, Bild- oder Videobearbeitungssoftware,...)
- Interaktive, transaktionsbasierte Systeme (z.B. im Internet)
- Eingebettete Systeme (Mobiltelefon, Auto, Mikrowelle, aber auch Robotersteuerung in der Industrie,...)
- Automatische Batch-Processing Systeme (z.B. Telefonabrechnungen, Lohnabrechnungen)
- Unterhaltungssysteme (z.B. Spiele, besonders wichtig ist die Benutzerinteraktion)
- Datenerfassungssysteme (Sensorsysteme in Raumfahrzeugen)
- ...

Was macht Software so speziell?

- Immateriell
- Unterliegt keinen physikalischen Gesetzen
- Schwierig zu vermessen
- Leicht zu verändern
- Keine Abnutzung
- Software altert aber trotzdem
- Man kann schlecht einzelne Ersatzteile liefern ohne das ganze System zu verstehen
- Kompletter Austausch relativ einfach
- Kann Bugs enthalten die sehr lange unentdeckt bleiben

Und wie verhindert man jetzt Fehler?

- Die Lösung: Software Engineering

The application of a **systematic, disciplined, quantifiable** approach to **development, operation, and maintenance** of software; that is, the **application of engineering to software**.

(IEEE Standard Computer Dictionary, 1990)

Was ist Engineering?

The **creative application** of **scientific principles** to **design** or **develop** structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to **forecast** their **behavior** under specific **operating conditions**; all as **respects** an intended function, **economics of operation and safety to life** and property.

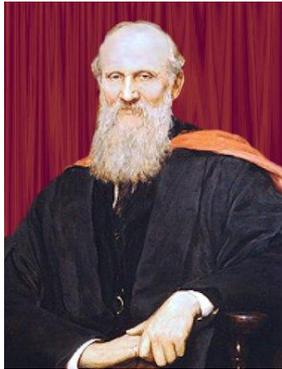
American Engineers' Council for Professional Development

Weitere Definitionsversuche

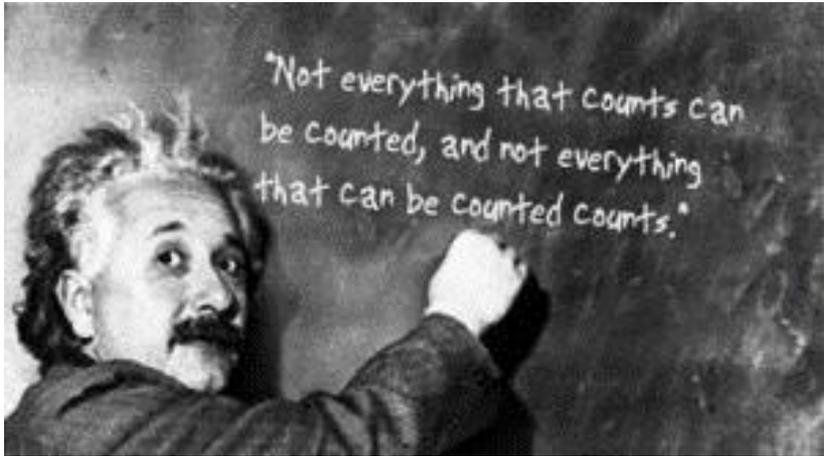
“The practical application of **scientific knowledge** in the design and construction of computer programs and the **associated documentation** required to develop, operate, and maintain them”
(Barry W. Boehm, 1979)

“Software Engineering is a discipline for the systematic production and maintenance of software which is developed **by a team within time limits and cost estimates**”
(Roger S. Pressman, 2001)

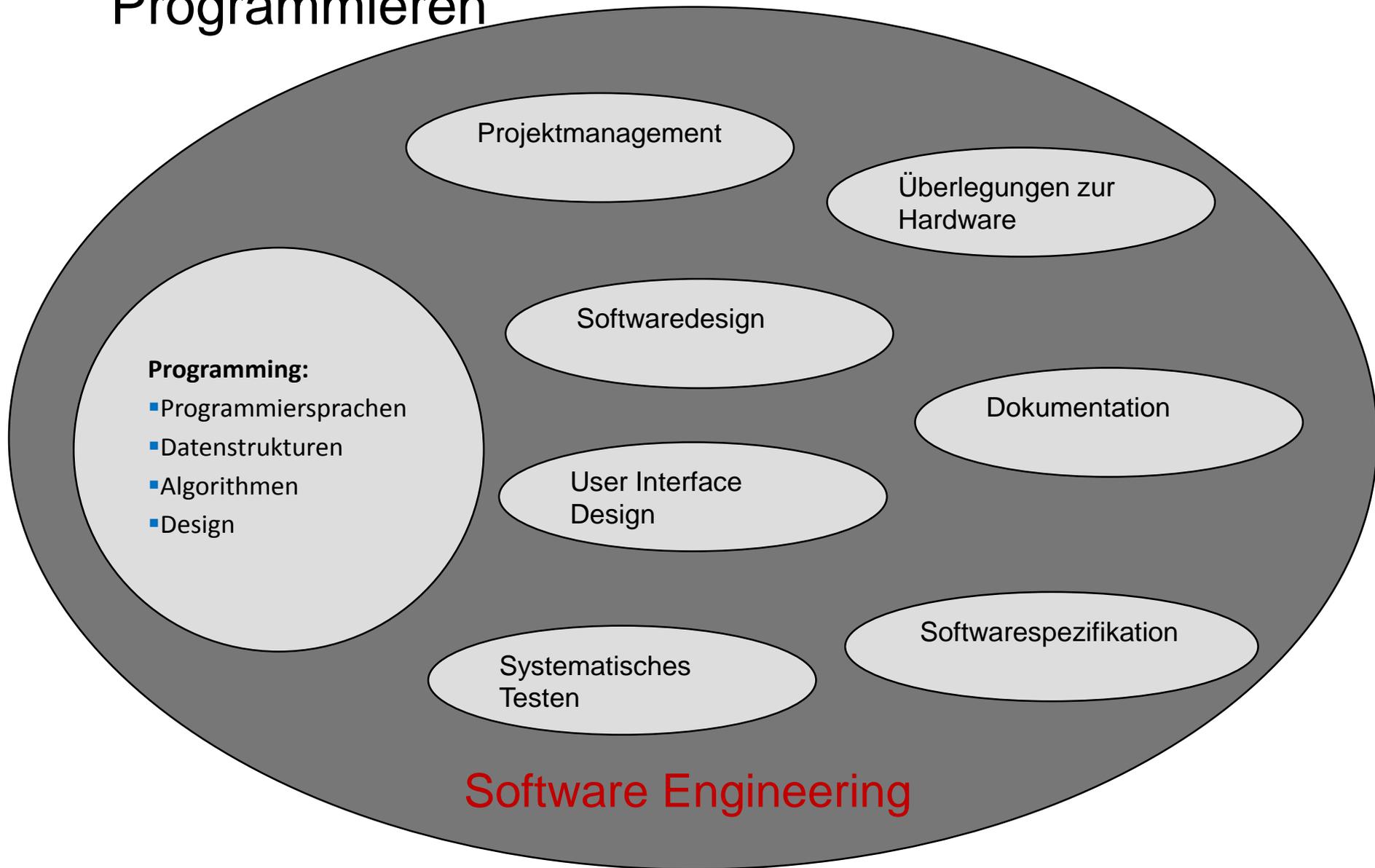
If you cannot measure it, then it is not science.



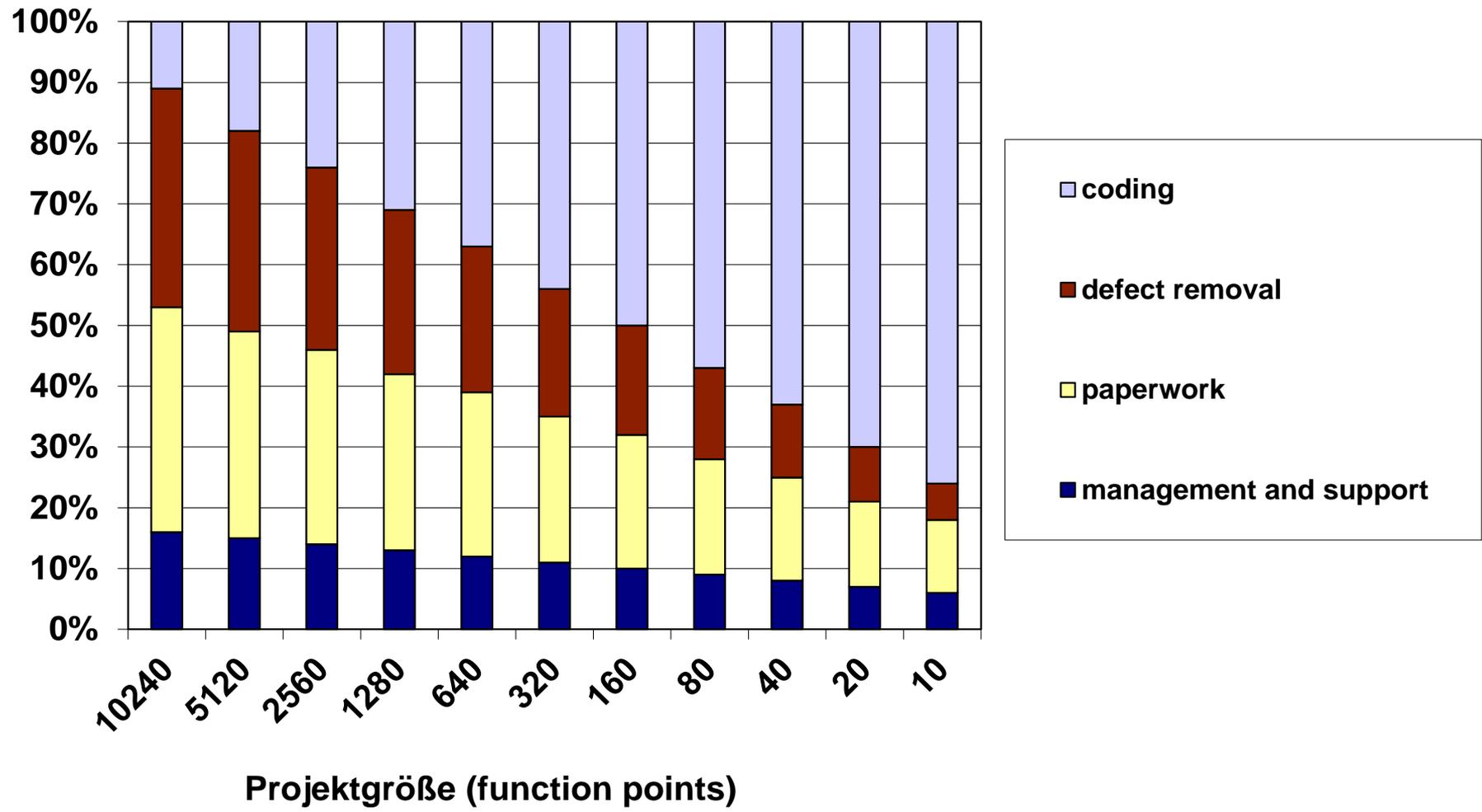
William Thomson Kelvin



Hinweis: Software-Engineering ist mehr als Programmieren



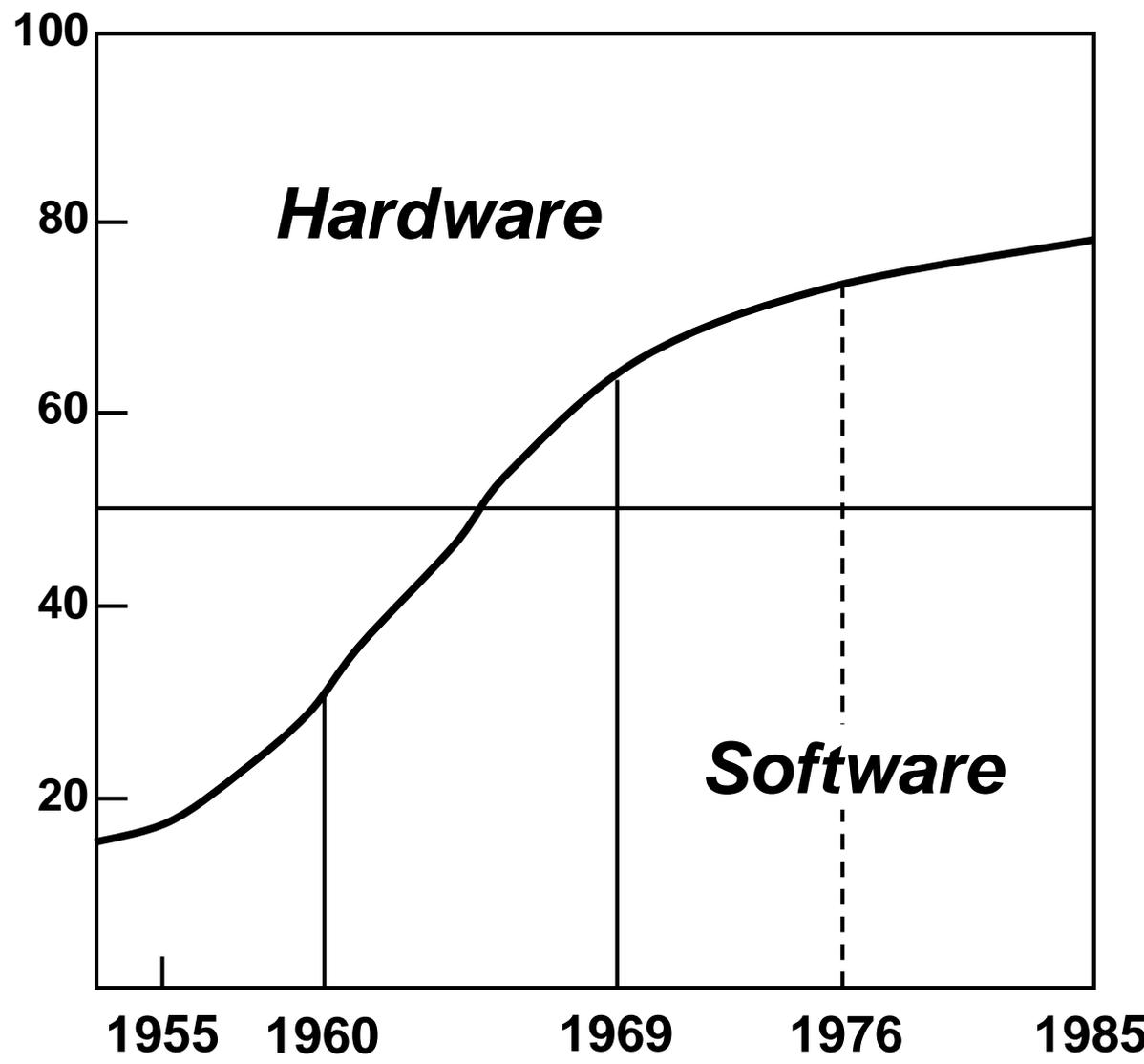
Arbeitsanteil abhängig von der Projektgröße



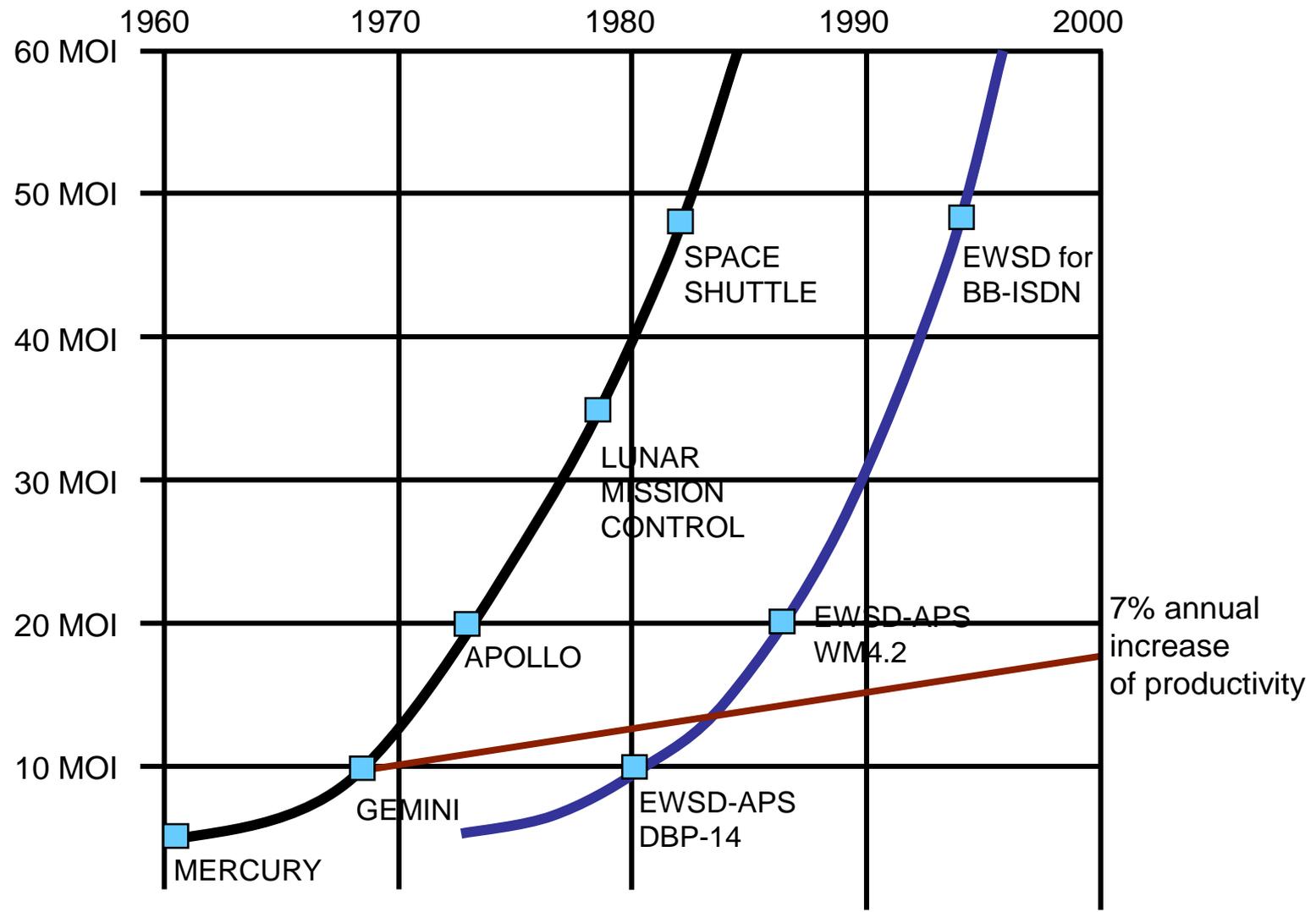
Yourdon, 1993, p. 151

Historisches: Softwarekrise

- Mitte der 60er Jahre überstiegen die Softwarekosten die Hardwarekosten



Grund: Zunehmende Komplexität der Software



MDI: millions of object-code instructions
EWSD: electronic dial system

Einige historische Meilensteine

- Erstmalige Erwähnung des Wortes Software Engineering 1968 auf einer NATO-Konferenz in Garmisch-Partenkirchen
 - Kritik der GOTO-Anweisungen von Dijkstra (1968) stattdessen strukturierte Programmierung, Nassi-Shneiderman Diagramme (Top-Down-Entwurf, graphische Veranschaulichung), ca. 1970
 - 70er: Entwicklung wichtiger Begriffe wie: Abstrakter Datentyp, Datenkapselung
 - 80er: Entwicklungsumgebungen (Editor, Compiler, Linker, Debugger)
 - 90er: Objektorientierte Programmierung wird populäre, insbesondere durch JAVA
- UML wird standartisiert (1997)

Was ist Media Engineering?

Die Anwendung der Prinzipien des Software Engineering auf die Entwicklung digitaler Medien.

Unterschiede zwischen Software und digitalen Medien

| Software | Digitale Medien |
|--|--|
| Oft für den Arbeitsplatz | Eher zur Unterhaltung |
| Nur Software | Oft Verbund von Hardware und Software (Wearables, Roboter, Cyber-Physical Systems) |
| Oft wohldefinierte und kleine Benutzergruppe | Unvorhersehbar – Ein Hit kann oft über Nacht aus 100 Benutzern 100 000 machen |
| Wohldefinierte Anforderungen | Oft konstant ändernde Anforderungen |
| Uni-modal Benutzerinterface | Multi-modal Interfaces: Web, mobile (WAP), Spache, Touchscreen ... |
| Design follows function | Function follows design |

Der Software Development-Lifecycle



Der Software Development-Lifecycle

