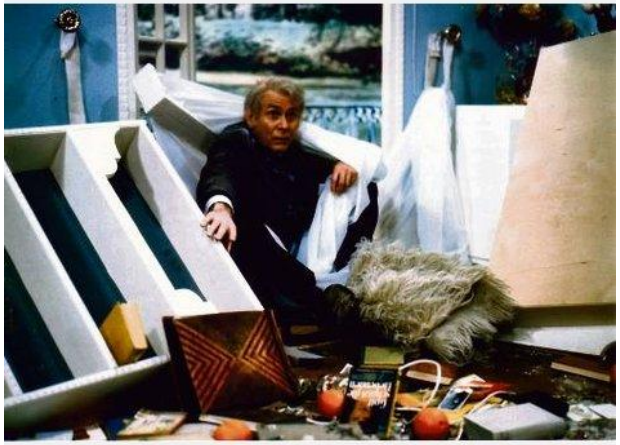
Media Engineering

Shit Happens

...aber nicht notwendigerweise



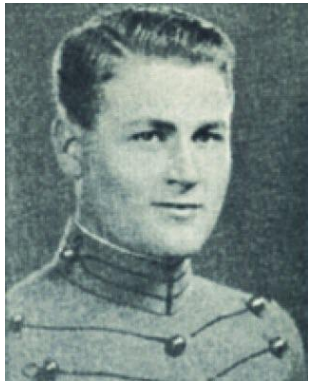
R. Weller

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Murphy's law

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



Edward A. Murphy

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: 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 wurden anstatt am Anfang *am Ende* der Kontonummern automatisch mit Nullen aufgefüllt

9	7	8	6	5	9	3	1	6	0

Beispiel: The Fappening

- 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: KSC

- 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

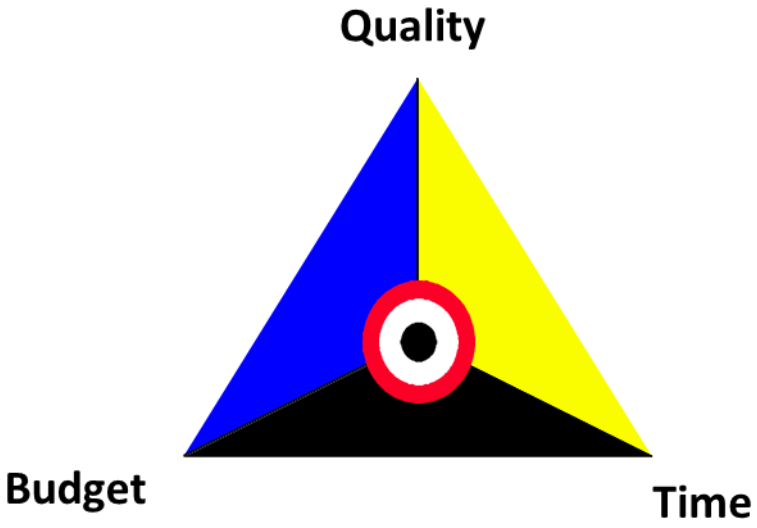
- Erstmals angekündigt: 1997
- 1998: Wechsel von Eigenentwicklung zur Quake2-Engine
- 2000: Wechsel zur Unreal-Engine
- 2009: Nach mehreren weiteren Engine-Wechseln stellt 3D Realms die Entwicklung ein und entlässt zahlreiche Mitarbeiter
- 2010: Weitergabe der Entwicklung an Gearbox
- 2011: Veröffentlichung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

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

Was bedeutet scheitern?

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



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)
    - ...

Vielfalt an Software

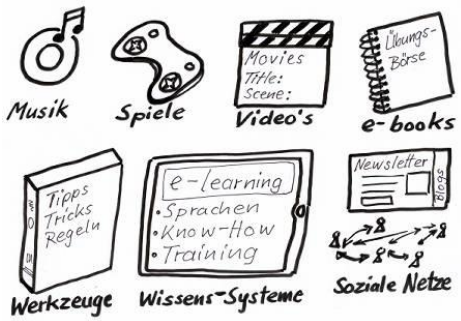
- Eigenständige Programme (Office-Anwendungen, Bild- oder Videobearbeitungssoftware, Computerspiele...)
- Interaktive, transaktionsbasierte Systeme (z.B. im Internet)
- Eingebettete Systeme (Mobiltelefon, Auto, Mikrowelle, aber auch Robotersteuerung in der Industrie,...)
- Automatische Batch-Processing 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

Kostencharakteristik digitaler Güter

- Die Industriebetriebe für digitale Güter unterliegen einer revolutionären Entwicklung (Verlage, Musiklabels, Filmindustrie etc.)
  - Lieferkosten (durch das Internet) extrem niedrig
  - Die erste Kopie erzeugt praktisch die gesamten Kosten: digitale Kopien (Reproduktion) kosten praktisch nichts
  - Kosten für Logistik wurden drastisch reduziert (Server anstatt Lagerräume)
  - Preisgestaltung kann damit sehr variable gestaltet werden (Wie definiert man überhaupt den Wert von Informationen?)



Laudon et al. (2010)

Effekte digitaler Märkte

Tabelle 10.2

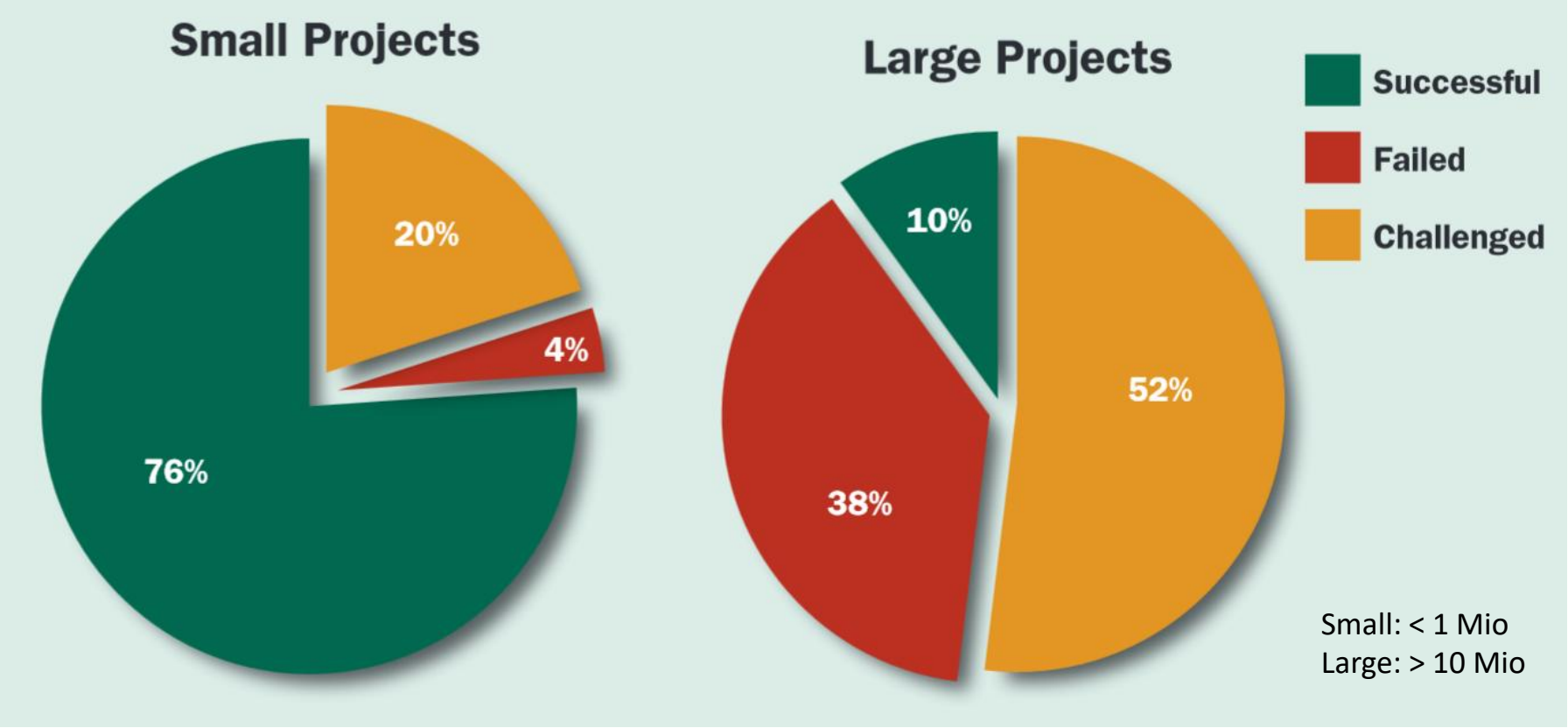
Digitale Märkte verglichen mit traditionellen Märkten

	Digitale Märkte	Traditionelle Märkte
Informationsasymmetrie	Reduziert	Hoch
Suchkosten	Niedrig	Hoch
Transaktionskosten	Niedrig (manchmal gegen null)	Hoch (Zeit, Reisen)
Dynamische Preisgestaltung	Niedrige Kosten, sofort	Hohe Kosten, verzögert
Preisdifferenzierung	Niedrige Kosten, sofort	Hohe Kosten, verzögert
Marktsegmentierung	Niedrige Kosten, moderate Präzision	Hohe Kosten, geringere Präzision
Wechselkosten	Höher/niedriger (abhängig von Produkteigenschaften)	Hoch
Netzwerkeffekte	Häufig, stark wirkend	Seltener, schwächer wirkend
Disintermediation	Eher möglich/wahrscheinlich	Weniger möglich/wahrscheinlich

Laudon et al. (2010)

Scheitern alle IT-Projekte?

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



Chaos Manifesto 2013, Standish Group

Gründe für das Scheitern von IT-Projekten

- Unrealistiche oder undeutlich formulierte Projektziele
- Falsche Einschätzung der notwendigen Resourcen
- 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

Und wie verhindert man jetzt das Scheitern?

- Die Lösung (?): Software Enineering

> 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"

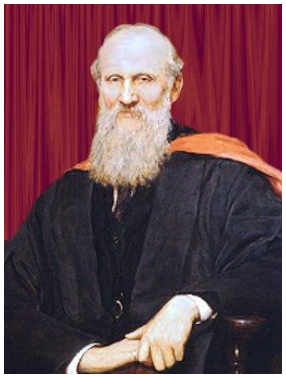
(Barry W. Boehm, 1979)

"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"

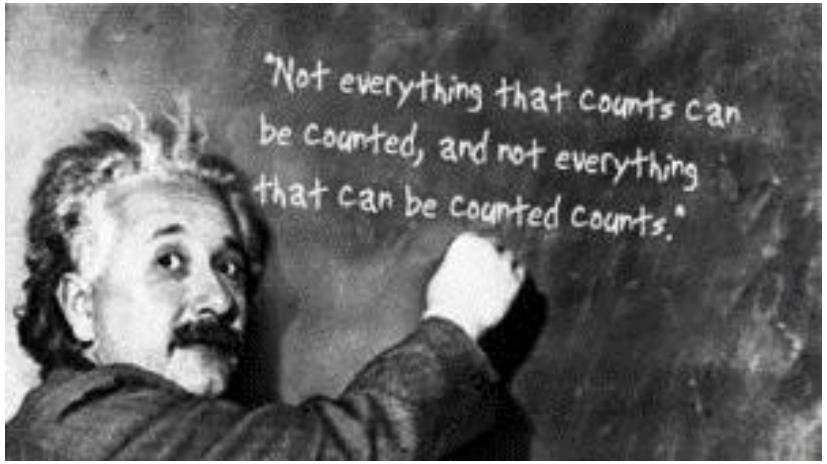
(Roger S. Pressman, 2001)

Imho

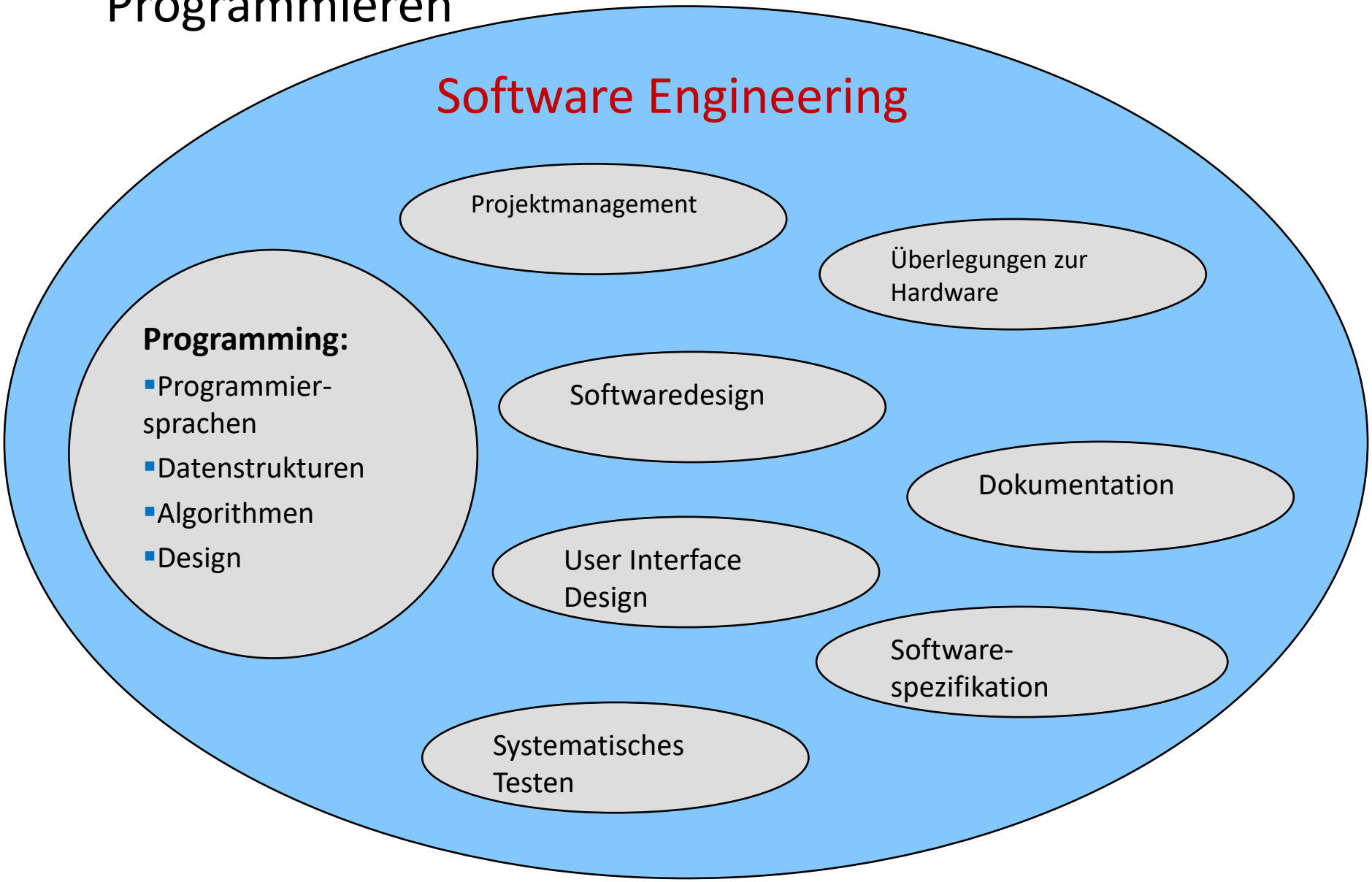
If you cannot measure it, then it is not science.



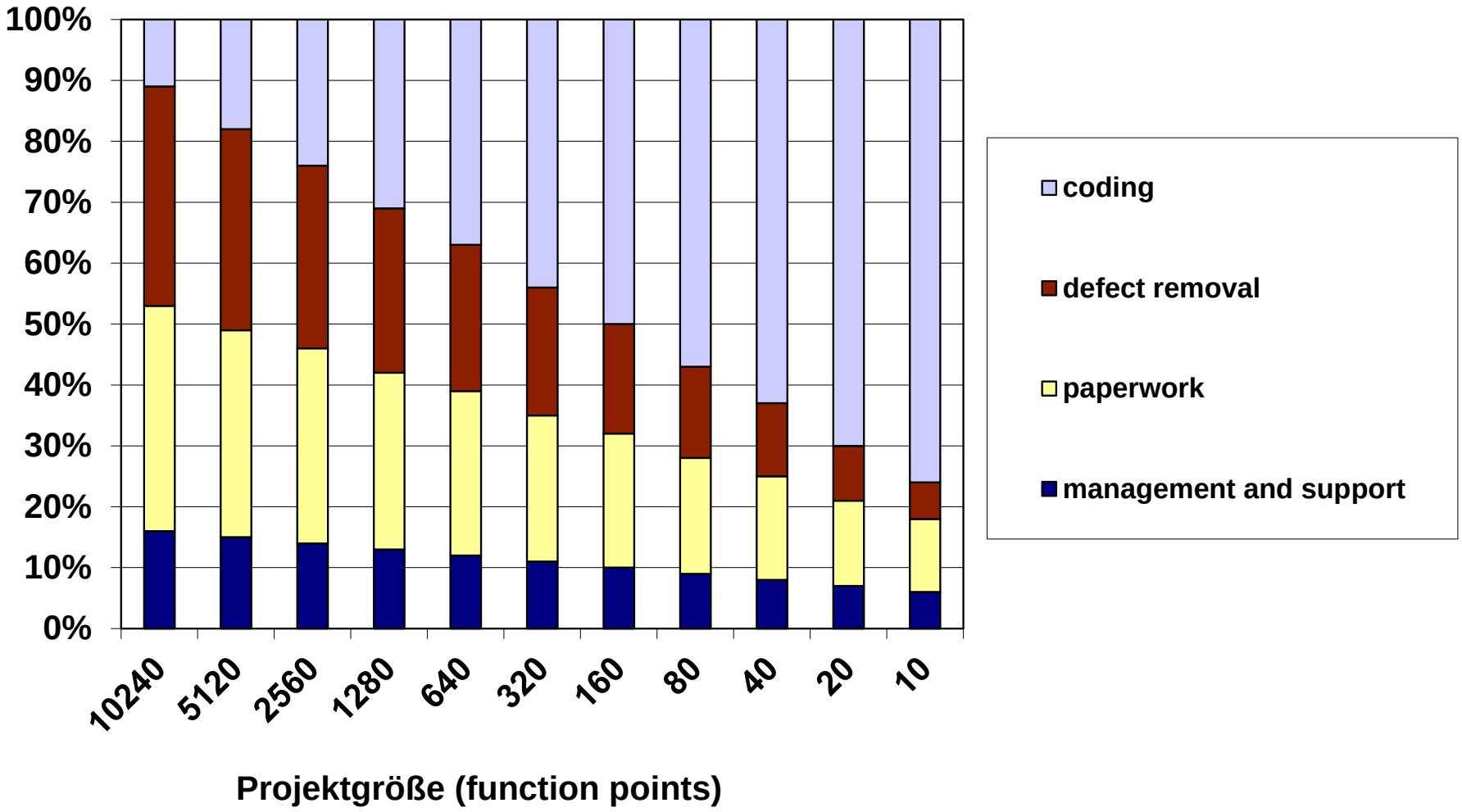
William Thomson Kelvin



Hinweis: Software-Engineering ist mehr als Programmieren



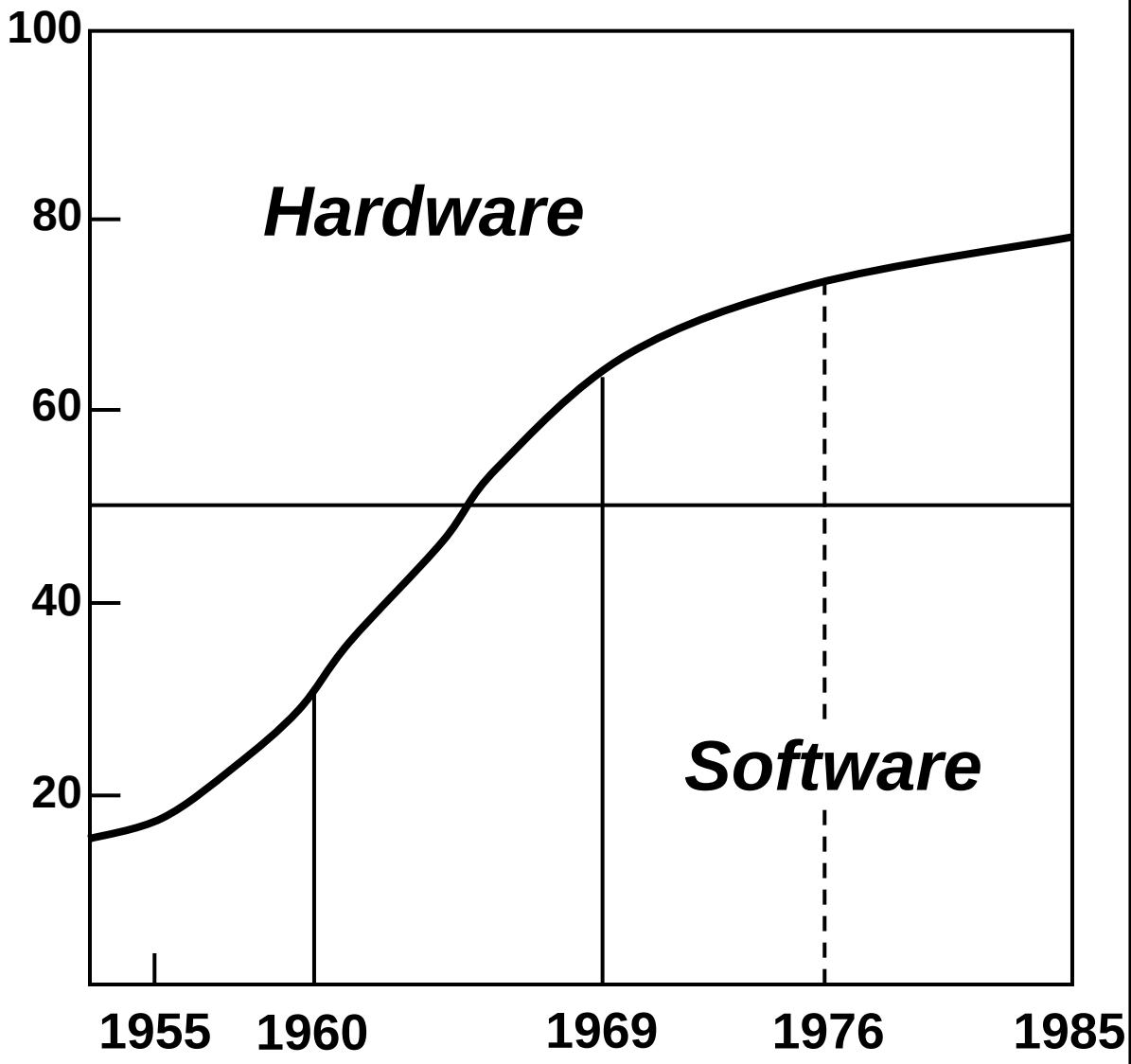
Arbeitsanteil abhängig von der Projektgröße



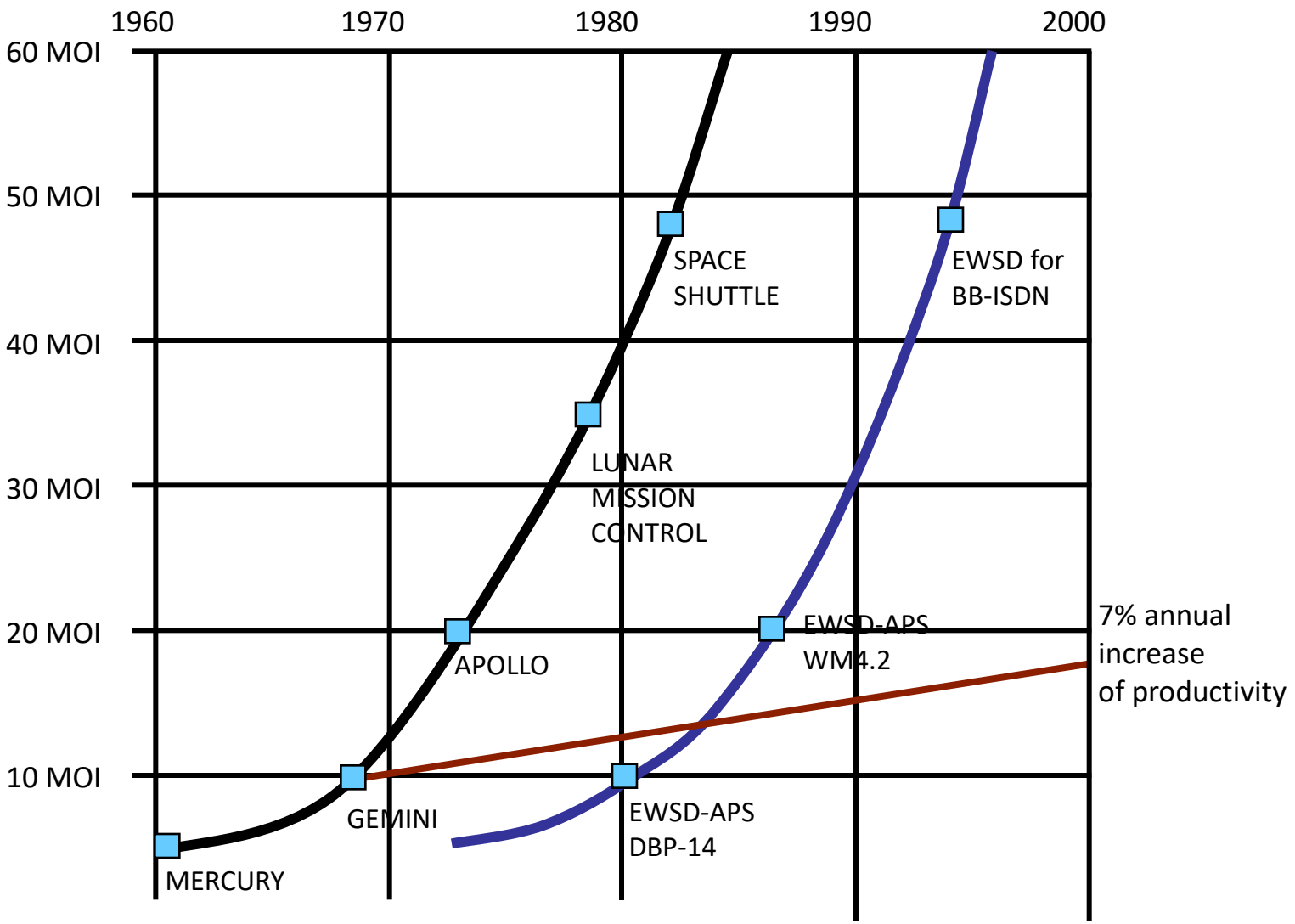
Yourdon, 1993, p. 151

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 Terminus „Software Engineering“ 1968 auf einer NATO-Konferenz in Garmisch-Partenkirchen
- Kritik der GOTO-Anweisungen von Dijskstra (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 populärer, 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 anderen 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 verändernliche Anforderungen
Uni-modal Benutzerinterface	Multi-modale Interfaces: Web, mobile (WAP), Spache, Touchscreen …
Design follows function	Function follows design

Der Software Development-Lifecycle