Die meiste Zeit des Lebens strebt der Mensch vergebens

Prof. Dr. Thorsten Koch

kobv

Kooperativer Bibliotheksverbund
Berlin-Brandenburg

ZIB
ZUSE INSTITUTE BERLIN

Scheitern (Misserfolg)

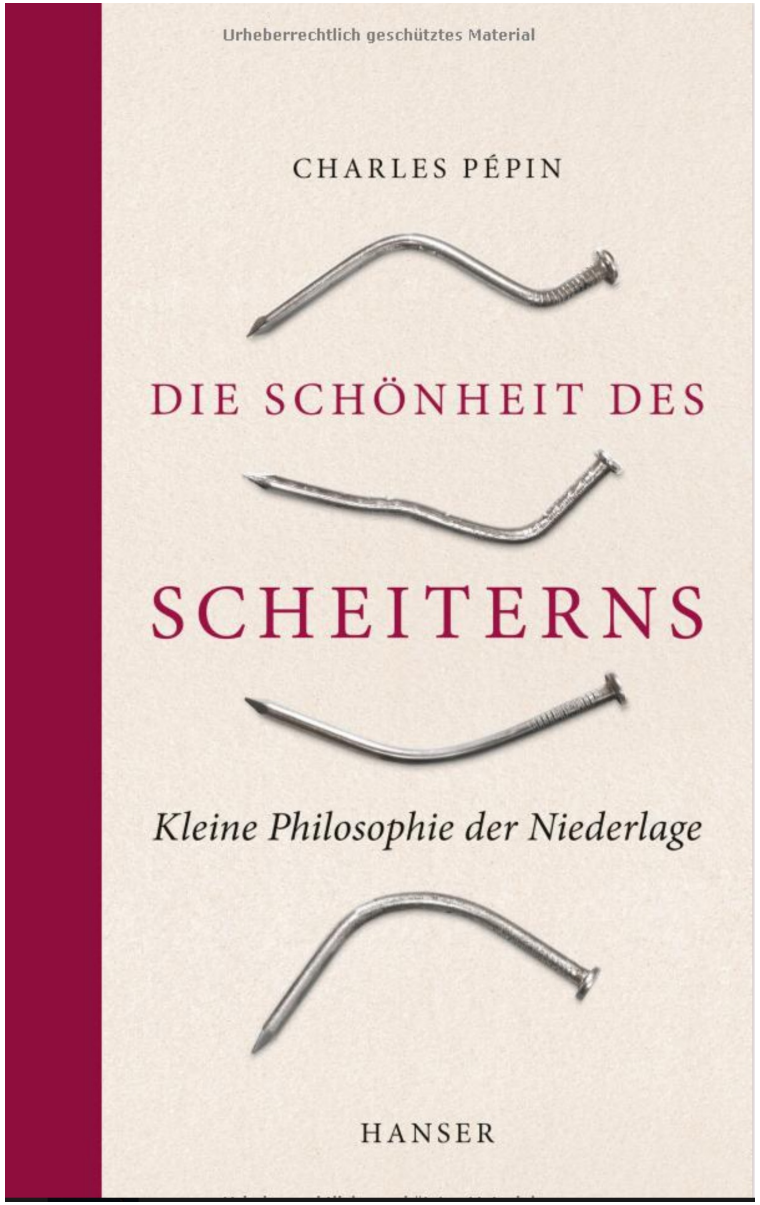
Unter Scheitern versteht man, wenn ein durch eine Handlung intendiertes Ziel endgültig nicht erreicht wird, wenn also etwas misslingt und nicht den erwünschten, angestrebten Erfolg hat.

[...] Das Wort stammt aus dem 17. Jahrhundert; es bildete sich aus dem Substantiv Scheiter (von Scheit und (Holz-)Scheit, in Stücke gehen), was wiederum auf Wendungen wie zu Scheitern gehen (in Trümmer auseinanderbrechen) aus dem 16. Jahrhundert zurückgeht. [...] Scheitern ist also – anders als bei den bedeutungsähnlichen Verben – nicht rückgängig zu machen.

Meist wird das Wort in eher oder deutlich negativer Konnotation verwendet, d. h. als Vorhalt und Vorwurf, dass etwas oder jemand keinen Erfolg hatte. Dabei ist zu beachten, dass in der Regel nur der scheitern kann, der aktiv wird und etwas – oft aus guten Gründen – versucht. Ohne solche Versuche gäbe es Stillstand. Aus einem Scheitern kann aber ein Erfolg werden, meist durch eine neue Anlage und durch veränderte Strategien.

https://de.wikipedia.org/wiki/Scheitern_(Misserfolg)

Die Schönheit des Scheiterns: Kleine Philosophie der Niederlage



1 Wer scheitert, lernt schneller .....
2 Nur wer irrt, kann verstehen .....
– *Die epistemologische Sicht* – .....
3 Die Krise als sich öffnendes Fenster .....
– *Eine Fragestellung unserer Zeit* – .....
4 Scheitern festigt den Charakter .....
– *Die dialektische Sicht* – .....
5 Scheitern lehrt Demut .....
– *Eine christliche Sicht?* – .....
6 Scheitern als Wirklichkeitserfahrung .....
– *Die stoische Sicht* – .....
7 Scheitern als Chance zur Neuerfindung des Selbst
– *Die existenzialistische Sicht* – .....
8 Scheitern als Fehlleistung oder glücklicher Unfall
– *Die psychoanalytische Sicht* – .....
9 Scheitern heißt nicht, ein Versager zu sein .....
– *Warum Scheitern so wehtut* – .....
10 Wagen heißt, das Wagnis des Scheiterns eingehen
11 Ist Wagemut erlernbar? .....

Niederlagen haben einen schlechten Ruf. Man sieht darin Schwäche statt Erfahrungsgewinn. Und das, obwohl so gut wie keine Erfolgsgeschichte ohne den unvermeidlichen Crash auskommt, das zeigen die Lebensläufe von Steve Jobs, Joanne K. Rowling oder Charles de Gaulle.

Charles Pépin betrachtet das Scheitern neu. Er begreift es im Sinne der Stoiker als privilegierte Begegnung mit der Realität und wie die Existezialisten als Chance zur Neuerfindung.

In seinem charmanten Kompendium entwirft er eine befreiende Philosophie des Scheiterns, die vor Optimismus sprüht und zeigt, was der verpasst, der nie gescheitert ist. Eine wunderbar kluge philosophische Anleitung zur gekonnten Niederlage.

Wo Churchill scheiterte

Wenig später legte der Erste Lord der Admiralität, Winston Churchill, seine Pläne für einen Seeangriff auf die Dardanellen vor. Am 16. Februar 1915 beschlossen die Briten erstmals, ein großes Landungsunternehmen durchzuführen.

Die Schlacht von Gallipoli wurde während des Ersten Weltkriegs vor und auf der türkischen Halbinsel Gallipoli auf der europäischen Seite der Dardanellen, aber auch auf der asiatischen Seite der Meerenge, zwischen Ägäis und Marmarameer ausgetragen.

Die Entente-Mächte wollten später in einer gemeinsamen Operation die Halbinsel besetzen und sie als Ausgangsbasis für die Eroberung der osmanischen Hauptstadt Istanbul nutzen, **scheiterten** jedoch an den Verteidigern.

Die Schlacht forderte auf beiden Seiten insgesamt 100.000 Tote und 250.000 Verwundete

Tragödie

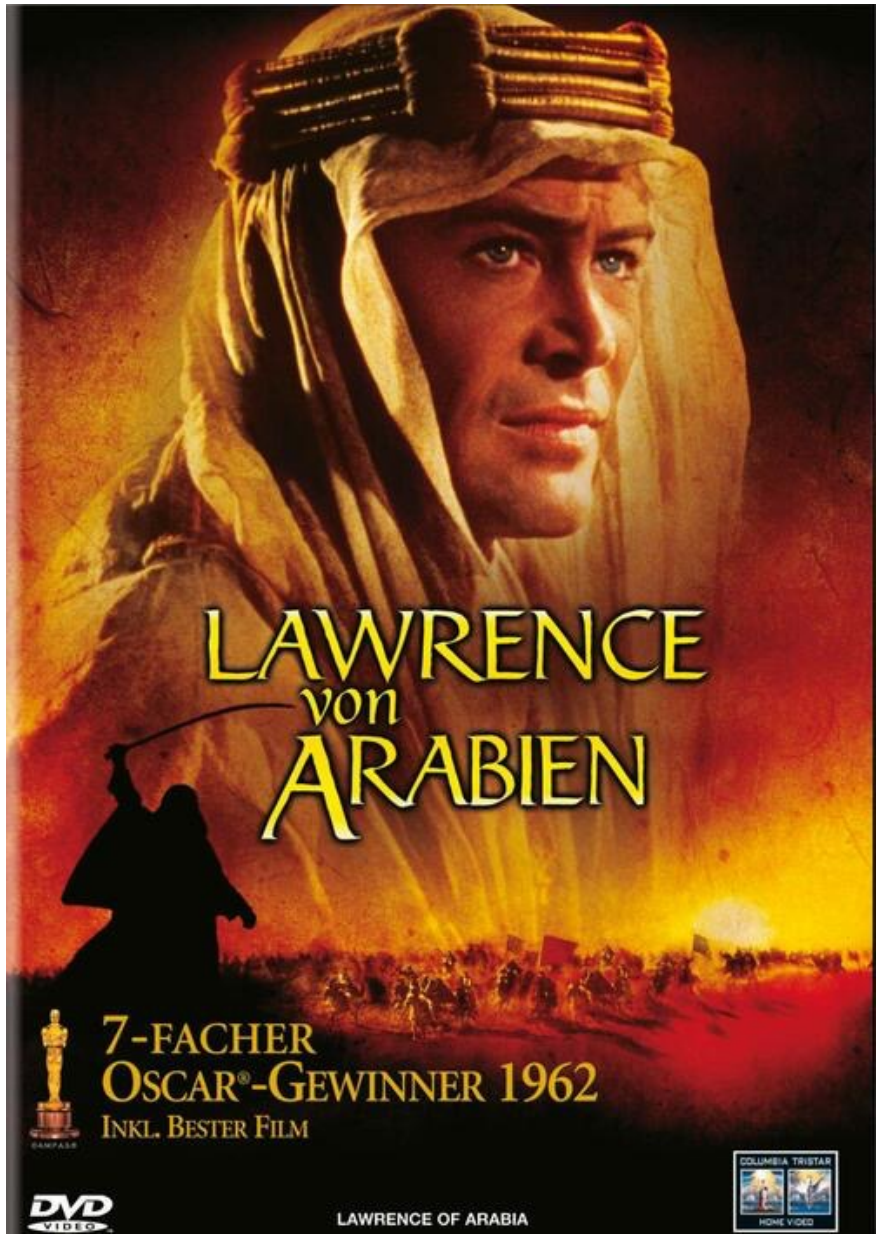
Die Tragödie ist eine Form des Dramas und neben der Komödie die bedeutsamste Vertreterin dieser Gattung. Sie lässt sich bis in das antike Griechenland zurückführen.

Kennzeichnend für die Tragödie ist der schicksalhafte Konflikt der Hauptfigur. Ihre Situation verschlechtert sich ab dem Punkt, an dem die Katastrophe eintritt. In diesem Fall bedeutet das Wort Katastrophe nur die unausweichliche Verschlechterung für den tragischen Helden. Allerdings bedeutet diese Verschlechterung nicht zwangsläufig den Tod des Protagonisten.

Das Scheitern des Helden ist in der Tragödie unausweichlich; die Ursache liegt in der Konstellation und dem Charakter der Figur. Der Keim der Tragödie ist, dass der Mensch der Hybris verfällt und dem ihm vorbestimmten Schicksal durch sein Handeln entgehen will.

https://de.wikipedia.org/wiki/Trag%C3%B6die

The seven pillars of wisdom



„Die Morgenluft einer zukünftigen Welt berauschte uns. Wir waren aufgewühlt von Ideen, die nicht auszudrücken und die nebulös waren, aber für die gekämpft werden sollte. Wir durchlebten viele Leben während dieser verwirrenden Feldzüge und haben uns selbst dabei nie geschont; doch als wir siegten und die neue Welt dämmerte, da kamen wieder die alten Männer und nahmen unseren Sieg, um ihn der früheren Welt anzupassen, die sie kannten.

Die Jugend konnte siegen, aber sie hatte nicht gelernt, den Sieg zu bewahren; und sie war erbärmlich schwach gegenüber dem Alter. Wir dachten, wir hätten für einen neuen Himmel und für eine neue Welt gearbeitet, und sie dankten uns freundlich und machten ihren Frieden.“

– T. E. Lawrence: Die sieben Säulen der Weisheit, Seite 850

https://de.wikipedia.org/wiki/T._E._Lawrence

Scheitern mit Ansage

Ein wesentlicher Aspekt beim Scheitern ist (hinterher) die Frage, ob es absehbar war und somit die Folge falscher Annahmen oder schlechter Planung, oder ob das Scheitern eher überraschend war: „es hat nicht sollen sein.“

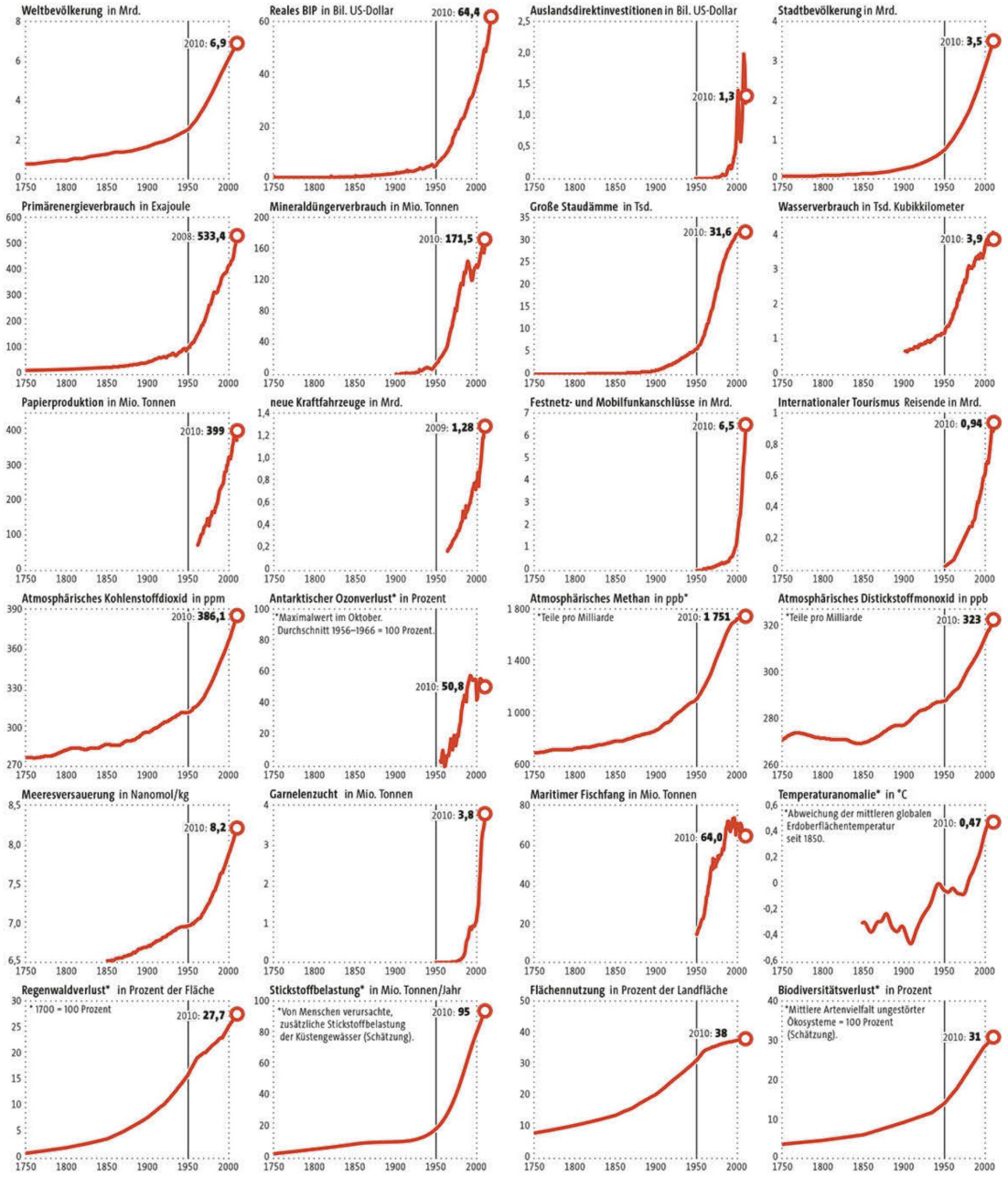
Beispiel: Wenn man sich die Erwartung über den Ausgang beim russischen Angriff auf die Ukraine ansieht, waren die meisten Leute (vermutlich insb. die russische Regierung) der Ansicht, das es schnell vorbei ist.

Hinterher fällt einem dann dies dazu ein:

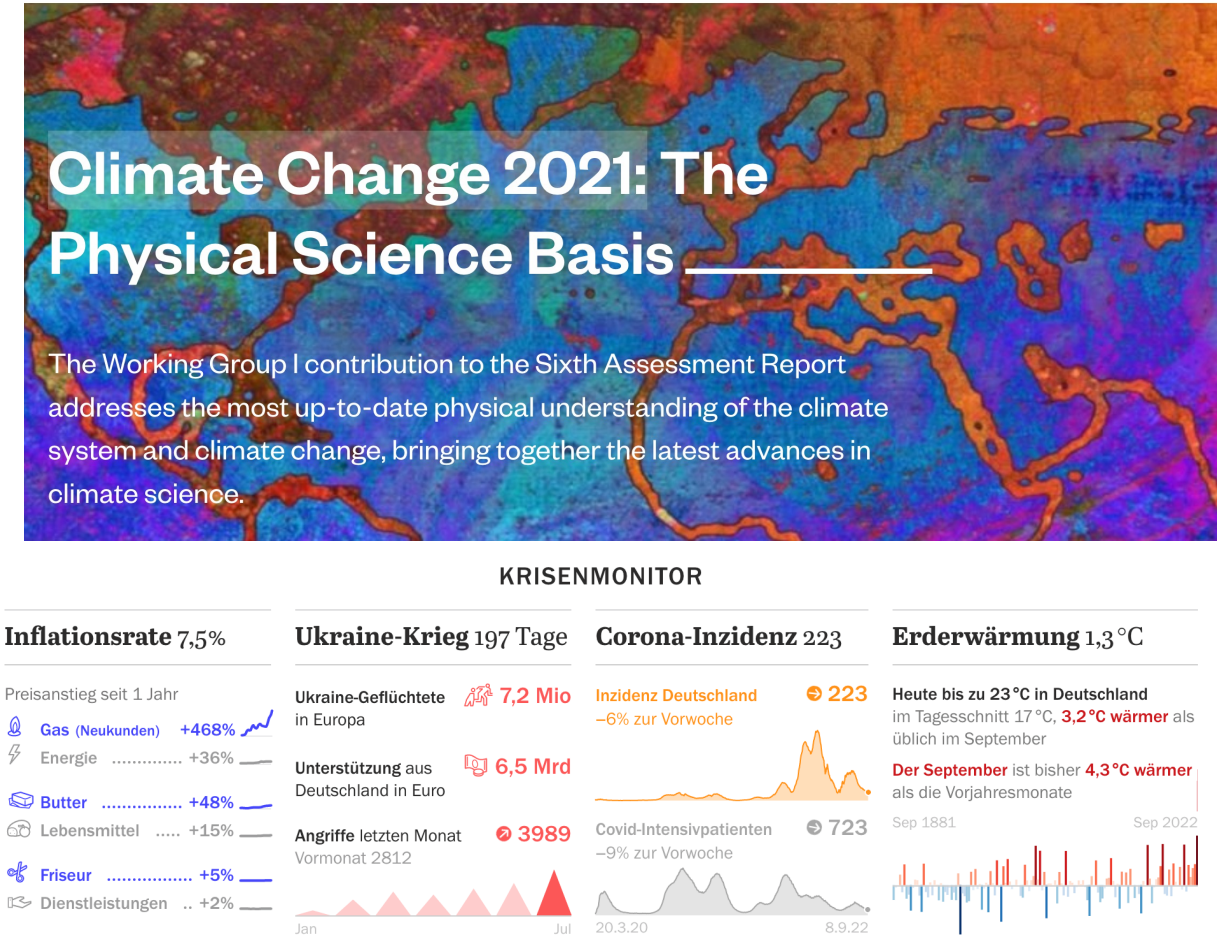
„Wenn du dich und den Feind kennst, brauchst du den Ausgang von hundert Schlachten nicht zu fürchten. Wenn du dich selbst kennst, doch nicht den Feind, wirst du für jeden Sieg, den du erringst, eine Niederlage erleiden. Wenn du weder den Feind noch dich selbst kennst, wirst du in jeder Schlacht unterliegen.“

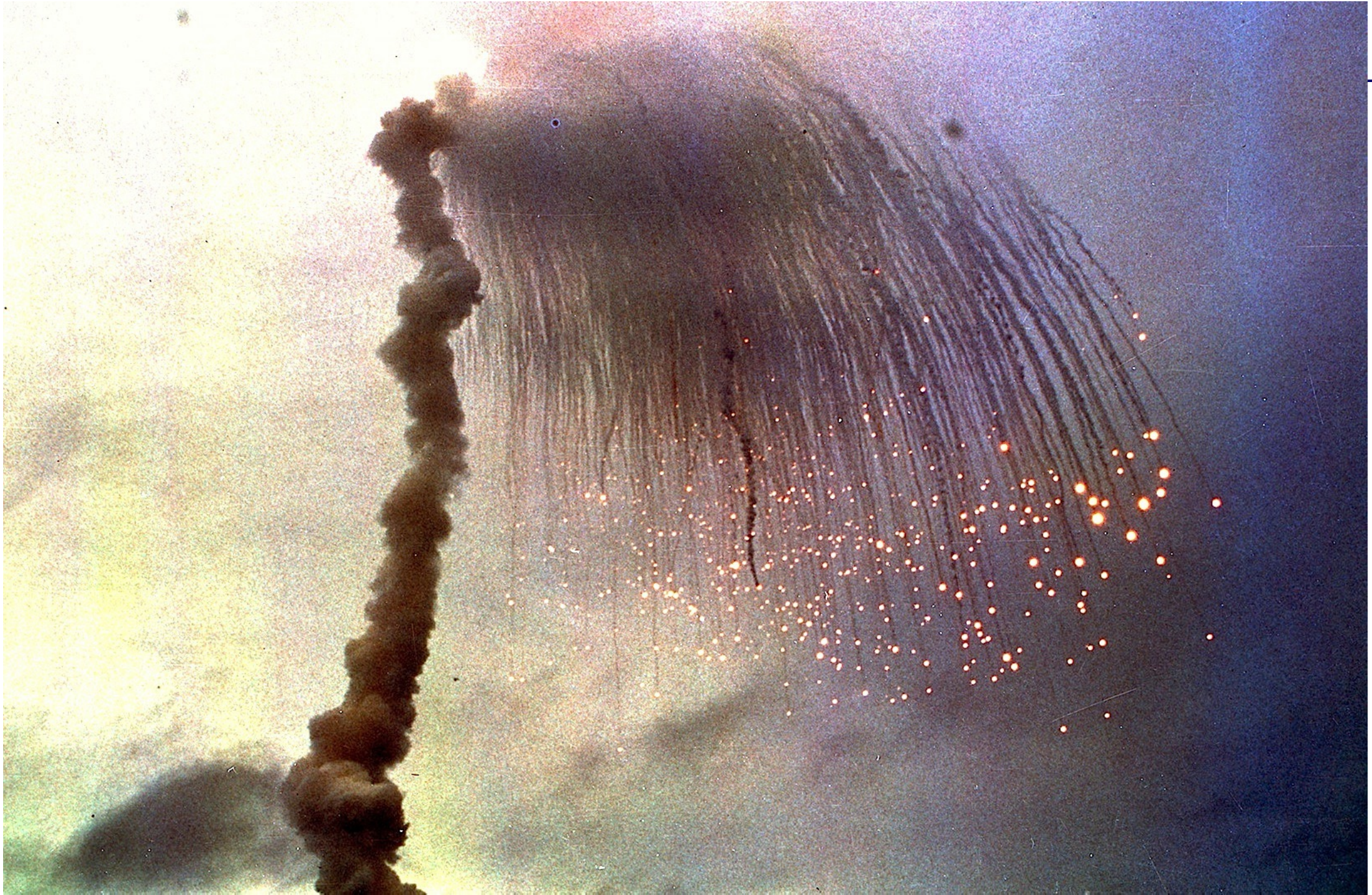
— Sunzi, Die Kunst des Krieges (ca. 500 v. Chr.)

Atlas der Globalisierung 2015: Weniger wird mehr



Und wenn die Folgen der Klimaerwärmung sichtbar werden, wird es heißen:
Damit haben wir nicht gerechnet.





kobv

Kooperativer Bibliotheksverbund
Berlin-Brandenburg

Ariane 5

Am 4. Juni 1996 endete der Jungfernflug der Ariane-5-Trägerrakete etwa 40 Sekunden nach Beginn der Flugsequenz mit einem Fehlschlag. In einer Höhe von etwa 3700 m kam die Trägerrakete von ihrer Flugbahn ab, zerbrach und explodierte. Die Ursache für das Versagen war der „vollständige Verlust der Leit- und Fluglageinformationen“ 30 Sekunden nach dem Start.

Das Problem wurde durch einen „Operandenfehler“ bei der Umwandlung von Daten in einem Unterprogramm von 64-Bit-Gleitkommazahlen in 16-Bit-Ganzzahlen mit Vorzeichen verursacht. Ein Wert war zu groß, um konvertiert zu werden, was den Operandenfehler verursachte. Dieser Fehler wurde im Programm nicht explizit behandelt (wohl aber andere potenzielle Operandenfehler), so dass der Computer, das Inertial Reference System (SRI), angehalten wurde, wie in anderen Anforderungen angegeben. Es gibt zwei SRIs, ein „aktives“ und ein „Hot-Backup“, und das aktive hielt kurz nach dem Backup an, wegen desselben Problems. Da nun keine Trägheitsführung mehr zur Verfügung stand und das Steuerungssystem davon abhängt, kann man sagen, dass die zerstörerische Folge das Ergebnis von „Garbage in, garbage out“ (GIGO) war.

Der Konvertierungsfehler trat in einer Routine auf, die von der Ariane 4 wiederverwendet worden war, deren Startbahn sich von der der Ariane 5 unterschied. Die Variable, die die Berechnung des Horizontal Bias (BH), einer Größe, die mit der horizontalen Geschwindigkeit zusammenhängt, enthielt, geriet daher außerhalb der „geplanten“ Grenzen (die für die Ariane 4 „geplant“ waren) und verursachte den Operandenfehler.

Anforderungen und Tests

a) Der Operandenbereich des Moduls wurde absichtlich nicht geschützt;

b) Der Grund dafür war, dass eine technische Analyse für die Verwendung in der Ariane 4 ergeben hatte, dass der Operand niemals außerhalb der Grenzen liegen würde;

c) Die sich aus dieser Analyse ergebende Bereichsanforderung wurde nicht auf die Anforderungen für die Ariane 5 übertragen;

d) die Tests wurden anhand der Anforderungen durchgeführt

Dies ist eher als Anforderungsfehler denn als Programmierfehler zu werten. Das Programm wurde auf der Grundlage von Ariane-4-Anforderungen geschrieben; diese Anforderungen wurden nicht in die Ariane-5-Anforderungsspezifikation übertragen; die Ariane-5-Anforderungen enthielten daher nicht die Reichweitenanforderung; die (in der Ariane 5 implizite) Reichweitenanforderung stand im Konflikt mit dem Verhalten der Ariane 5 (wie in anderen Ariane-5-Anforderungen erläutert); die Anforderungen stießen auf das Verhalten und die Rakete wurde zerstört.

(Es ist nicht überraschend, dass es sich um einen Anforderungsfehler handelte - über 90 % der Ausfälle von sicherheitskritischen Systemen sind Anforderungsfehler, wie eine inzwischen zur Folklore gewordene JPL-Studie zeigt)

MULTICS

Multics ("Multiplexed Information and Computing Service") ist ein einflussreiches frühes Time-Sharing-Betriebssystem, das auf dem Konzept eines einstufigen Speichers basiert.[4][5] Nathan Gregory schreibt, dass Multics „alle modernen Betriebssysteme seither beeinflusst hat, von Mikrocomputern bis zu Großrechnern“.

Die ersten Planungen und Entwicklungen für Multics begannen 1964 in Cambridge, Massachusetts. Ursprünglich war es ein Kooperationsprojekt unter der Leitung des MIT (Projekt MAC mit Fernando Corbató), zusammen mit General Electric und Bell Labs. Es wurde auf dem speziell dafür konzipierten Computer GE 645 entwickelt; der erste wurde im Januar 1967 an das MIT geliefert. GE bot seine früheren 635-Systeme mit einem frühen Timesharing-System namens „Mark I“ an und beabsichtigte, den 645 mit Multics als größeren Nachfolger anzubieten. Bell zog sich 1969 aus dem Projekt zurück, als klar wurde, dass es kurzfristig kein funktionierendes System liefern würde. Kurz darauf beschloss GE, sich ganz aus der Computerbranche zurückzuziehen, und verkaufte den Geschäftsbereich 1970 an Honeywell. Honeywell bot Multics kommerziell an, jedoch mit begrenztem Erfolg.

https://en.wikipedia.org/wiki/Multics

https://www.multicians.org/myths.html

R. L. Glass: **Facts** and Fallacies of Software Engineering 1/5

Menschen

1. Der wichtigste Faktor bei der Softwarearbeit ist die Qualität der Programmierer.
2. Die besten Programmierer sind bis zu 28 Mal besser als die schlechtesten Programmierer.
3. Wenn man einem verspäteten Projekt Leute hinzufügt, wird es später.
4. Die Arbeitsumgebung hat einen großen Einfluss auf die Produktivität und Qualität.

Werkzeuge und Techniken

5. Der Hype (um Tools und Techniken) ist die Plage im Haus der Software.
6. Neue Tools/Techniken führen zunächst zu Produktivitäts- und Qualitätseinbußen.
7. Softwareentwickler reden viel über Tools, benutzen sie aber nur selten.

Schätzungen

8. Eine der beiden häufigsten Ursachen für aus dem Ruder laufende Projekte sind schlechte Schätzungen.
9. Die Schätzung von Software erfolgt meist zum falschen Zeitpunkt.
10. Die Software-Schätzung wird in der Regel von den falschen Personen durchgeführt.
11. Software-Schätzungen werden selten im Laufe des Projekts korrigiert.
12. Es ist nicht überraschend, dass Software-Schätzungen schlecht sind. Aber wir leben und sterben trotzdem mit ihnen!
13. Die Verbindung zwischen dem Softwaremanagement und den Programmierern ist gestört.
14. Die Antwort auf eine Machbarkeitsstudie lautet fast immer „Ja“.

R. L. Glass: **Facts** and Fallacies of Software Engineering 2/5

Wiederverwendbarkeit

15. Wiederverwendung im Kleinen ist ein gut gelöstes Problem.
16. Die Wiederverwendung im Großen bleibt ein weitgehend ungelöstes Problem.
17. Die Wiederverwendung im Großen funktioniert am besten bei Familien verwandter Systeme.
18. Wiederverwendbare Komponenten sind 3 mal so schwer zu erstellen und sollten in 3 Umgebungen ausprobiert werden.
19. Die Modifikation von wiederverwendetem Code ist besonders fehleranfällig.
20. Die Wiederverwendung von Entwurfsmustern ist eine Lösung für die Probleme bei der Wiederverwendung von Code.

Komplexität

21. Für jede 25% Zunahme der Problemkomplexität gibt es eine 100% Zunahme der Lösungskomplexität.
22. 80% der Softwarearbeit ist intellektuell. Ein beträchtlicher Teil davon ist kreativ. Wenig davon ist bürokratisch.

Anforderungen

23. Eine der beiden häufigsten Ursachen für aus dem Ruder laufende Projekte sind instabile Anforderungen.
24. Fehler in den Anforderungen sind die teuersten, die während der Produktion zu beheben sind.
25. Fehlende Anforderungen sind die am schwersten zu korrigierenden Anforderungsfehler.

Entwurf

26. Explizite Anforderungen „explodieren“, wenn sich implizite (Design-)Anforderungen für eine Lösung entwickeln.
27. Es gibt selten nur eine beste Designlösung für ein Softwareproblem.
28. Design ist ein komplexer, iterativer Prozess. Anfängliche Lösungen sind in der Regel falsch und sicherlich nicht optimal.

R. L. Glass: **Facts** and Fallacies of Software Engineering 3/5

Kodierung

29. Die „Primitive“ des Designers entspricht selten der „Primitiven“ des Programmierers.
30. COBOL ist eine sehr schlechte Sprache, aber alle anderen (für Geschäftsanwendungen) sind noch viel schlechter.

Fehlerbeseitigung

31. Die Fehlerbeseitigung ist die zeitaufwändigste Phase des Lebenszyklus.

Testen

32. Software wird in der Regel bestenfalls mit einem Abdeckungsgrad von 55%-60% Prozent (Zweig) getestet.
33. 100% Abdeckung sind noch lange nicht genug.
34. Testwerkzeuge sind unerlässlich, aber viele werden nur selten eingesetzt.
35. Testautomatisierung ist selten. Die meisten Testaktivitäten können nicht automatisiert werden.
36. Vom Programmierer erstellter, integrierter Debug-Code ist eine wichtige Ergänzung zu Testwerkzeugen.

Überprüfungen und Inspektionen

37. Strenge Inspektionen können bis zu 90% der Fehler beseitigen, bevor der erste Testfall ausgeführt wird.
38. Strenge Inspektionen sollten jedoch das Testen nicht ersetzen.
39. Überprüfungen nach der Auslieferung („Retrospektiven“) sind wichtig, werden aber nur selten durchgeführt.
40. Überprüfungen sind sowohl technischer als auch soziologischer Natur, und beide Faktoren müssen berücksichtigt werden.

R. L. Glass: **Facts** and Fallacies of Software Engineering 4/5

Wartung

41. Die Wartung verschlingt in der Regel 40-80 Prozent der Softwarekosten. Sie ist wahrscheinlich die wichtigste Lebenszyklusphase von Software.
42. Erweiterungen machen etwa 60 Prozent der Wartungskosten aus.
43. Wartung ist eine Lösung, nicht ein Problem.
44. Das bestehende Produkt zu verstehen ist die schwierigste Aufgabe der Wartung.
45. Bessere Methoden führen zu MEHR Instandhaltung, nicht zu weniger.

Qualität

46. Qualität IST: eine Sammlung von Eigenschaften.
47. Qualität ist NICHT: Benutzerzufriedenheit, Erfüllung von Anforderungen, Einhaltung von Kosten und Zeitplänen oder Zuverlässigkeit.

Verlässlichkeit

48. Es gibt Fehler, zu denen die meisten Programmierer neigen.
49. Fehler neigen dazu, sich zu häufen.
50. Es gibt keinen einzigen besten Ansatz zur Beseitigung von Softwarefehlern.
51. Restfehler werden immer bestehen bleiben. Das Ziel sollte sein, schwere Fehler zu minimieren oder zu beseitigen.

Effizienz

52. Effizienz entsteht eher durch gutes Design als durch gute Kodierung.

R. L. Glass: Facts and **Fallacies** of Software Engineering 5/5

Management

1. Man kann nicht managen, was man nicht messen kann.
2. Man kann Qualität in ein Softwareprodukt hineinmanagen.

Menschen

3. Programmieren kann und sollte egolos sein.

Werkzeuge und Techniken

4. Werkzeuge und Techniken: One size fits all
5. Software braucht mehr Methoden.

Kostenschätzung

6. Um Kosten und Zeitplan abzuschätzen, müssen zuerst die Codezeilen geschätzt werden.

Testen und Überprüfen

7. Zufällige Testeingaben sind ein guter Weg zur Optimierung von Tests.
8. „Wenn man genug Augen hat, sind alle Fehler oberflächlich“.

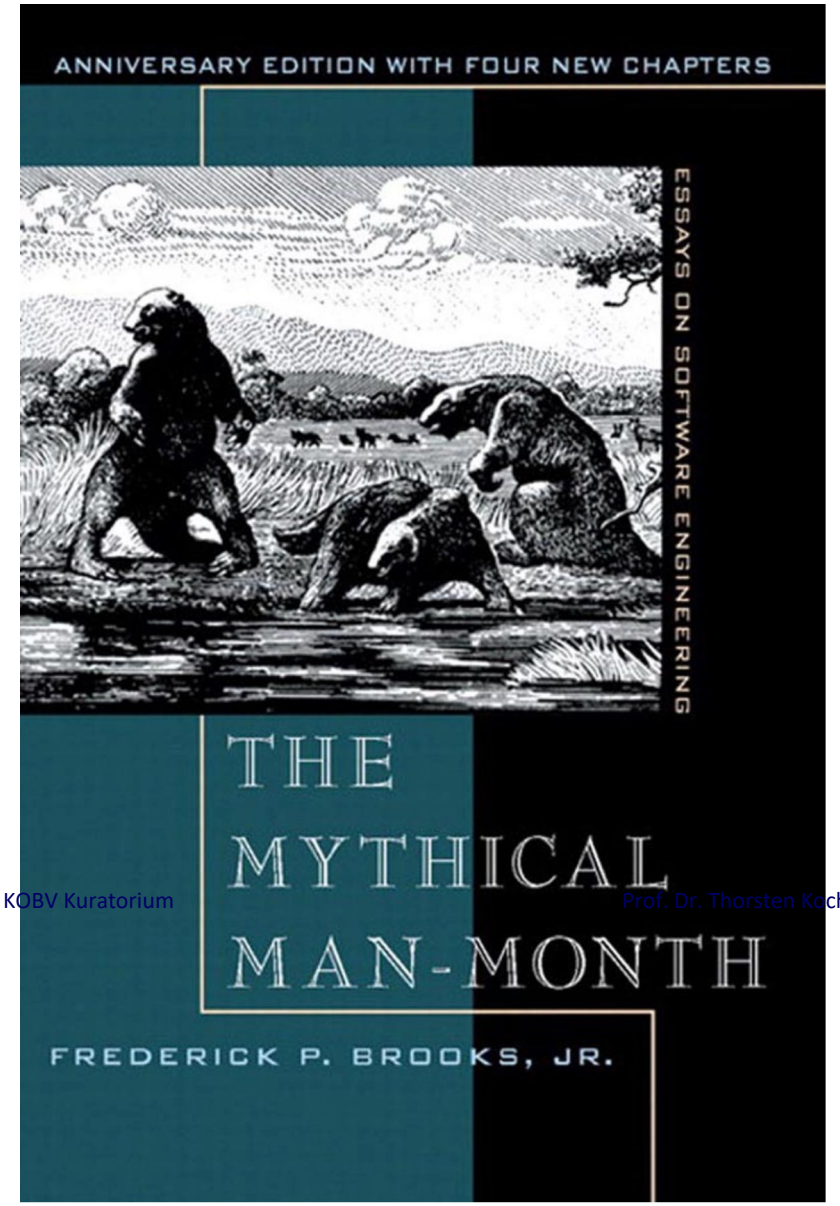
Wartung

9. Die Vorhersage künftiger Wartungskosten und die Entscheidung über den Austausch von Produkten lässt sich nur anhand von Kostendaten aus der Vergangenheit treffen.

Bildung

10. Man bringt Menschen das Programmieren bei, indem man ihnen zeigt, wie man Programme schreibt.

Der Klassiker zum Thema Software Engineering/Management



Dedication

Dedication of the 1975 edition

To two who especially enriched my IBM years:

Thomas J. Watson, Jr.,

whose deep concern for people still permeates his company,

and

Bob O. Evans,

whose bold leadership turned work into adventure.

Dedication of the 1995 edition

To Nancy,

God's gift to me.

ISBN 0-201-83595-9

Text printed in the United States on recycled paper at RR Donnelley Crawfordsville in Crawfordsville, Indiana.

Printing 35th January 2010

The essay entitled, No Silver Bullet, is from Information Processing 1986, the Proceedings of the IFIP Tenth World Computing Conference, edited by H.-J. Kugler, 1986, pages 1069–1076. Reprinted with the kind permission of IFIP and Elsevier Science B.V., Amsterdam, The Netherlands.

Library of Congress Cataloging-in-Publication Data

Brooks, Frederick P., Jr. (Frederick Phillips)
The mythical man-month : essays on software engineering / Frederick P. Brooks, Jr. — Anniversary ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-83595-9
1. Software engineering. I. Title.
QA76.758.B75 1995
)68—dc20 94-36653
CIP

Venture Capital vs. Forschung

…

Das Scheitern des Ziels kann auch unwichtig sein

„Der Weg ist das Ziel.“

— Konfuzius?

Und ansonsten: *Aut viam inveniam aut faciam*

Vielen Dank!

https://de.toonpool.com/cartoons/Wachstum_177057