Schlüsseltechnologie #082

Literatur: Vom Mythos des Mann-Monats III

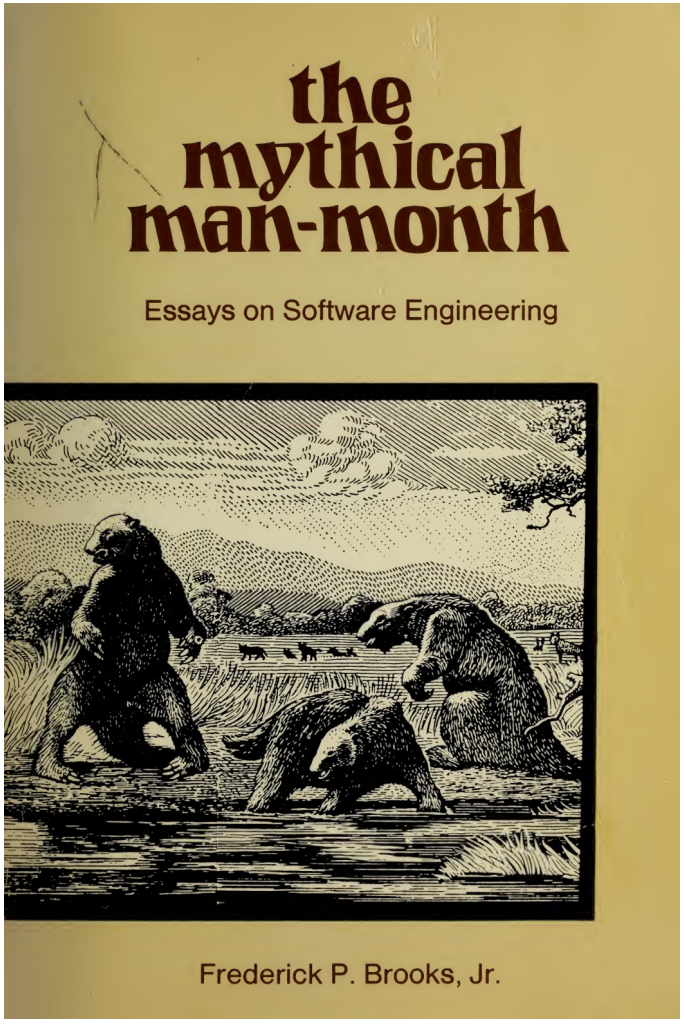
19. September 2025 @ Datenspuren

Wer sind wir?

- Computertechnik von Grund auf erklärt
- für ein technikinteressiertes Laienpublikum
- seit April 2021 alle 3 Wochen

www.schlüsseltechnologie-podcast.de

Wir lesen...



- *Vom Mythos des Mann-Monats*, Frederick K. Brooks (1975).
- 1. Auflage im englischen Original
- digitalisiert durch Internet Archive

Kapitel 1: Die Teergrube

- Zähheit/Trägheit in der Software-Entwicklung aufgrund mehrerer parallel wirkender Gründe

Programme bauen ist vergleichsweise einfach, doch dann:

- Produktisierung: Generalisierung, Testabdeckung, Dokumentation, fortlaufende Wartung
  - mindestens 3x mehr Aufwand
- Programmsystem: konsistenter Stil und definierte Schnittstellen über mehrere Programme hinweg
  - mindestens 3x mehr Aufwand

Kapitel 2: Vom Mythos des Mann-Monats

- Aufwandsschätzungen mit abstrakten Mann-Monaten funktionieren nicht
  - Arbeitsgeschwindigkeit variiert drastisch je nach Kompetenzniveau (bis zu Faktor 10)
  - Ausbremsung durch Kommunikationsaufwand je nach Organisationsgröße
- Programmierer sind schlecht im Schätzen von Zeitaufwand
  - Produktisierung, Testphasen etc. werden oft unterschätzt
- stattdessen Orientierung der Zeitpläne an Kundenwünschen

Kapitel 3: Das OP-Team

- Vorschlag einer Organisationsstruktur für Entwicklungsteams
  - „Chirurg“ erarbeitet das Design und setzt es um
  - „Copilot“ berät den Chirurgen als dessen Stellvertreter
  - außerdem Adminstrator, Editor, Schreiber, Sekretär, Tester, etc.
- heute großteils überholt, da zuarbeitende Rollen durch Technik ersetzt wurden
  - damit aber Arbeitsverdichtung bei den Spezialisten

Kapitel 4: Aristokratie, Demokratie und System-Design

- im Kirchenbau oft mehrere Generationen von Architekten und Erbauern, aber trotzdem gestalterische Kohärenz
- trotzdem auf jeder Ebene Entscheidungskompetenz, je nach Tragweite der einzelnen Entscheidung
  - gute Architekten binden ihre Erbauer fortlaufend ein, besonders wenn es um Praktikabilitätsfragen geht

Kapitel 5: Das Problem des zweiten Systems

- Ideen des Architekten oft wegen Unpraktikabilität verworfen
- doch dann: erstes System erfolgreich, Architekt hat Prestige und Erfahrung gewonnen, Ersatzneubau steht an
  - „Dieses zweite ist das gefährlichste System, dass man jemals entwirft.“

Kapitel 6: Weitergabe von Informationen

- Architekten erstellen vor allem Dokumentation
  - Pläne für Erbauer
  - Handbücher für Benutzer
- Qualitäten guter Dokumentation
  - konzeptionelle Integrität
  - möglichst maschinenlesbare Beschreibung des Verhaltens
  - automatische Validierbarkeit
  - etc.
- Entwurfsprozess
  - Brainstorming
  - Pro/Kontra-Abwägung aller Optionen
  - Dokumentation der Entscheidung

Kapitel 7: Warum ist der Turm zu Babel gefallen?

- Großprojekte gelingen nur bei sauberer Aufgabenteilung entlang klarer Schnittstellen
  - damit effektive Kommunikation extrem wichtig
- Organisationsstruktur: Brooks präferiert baumförmige Strukturen mit klaren Entscheidungskompetenzen
- feste Rollen in jedem einzelnen Entwicklungsteam
  - „Producer“: Arbeitsaufteilung, Zeitpläne, Kommunikation *außerhalb* des Teams
  - „Architect“: Design, Entwicklungsleitung, Kommunikation *innerhalb* des Teams

Kapitel 8: "Calling the Shot"

- Aufwandsschätzung für den reinen Code einigermaßen machbar, aber dann:
  - Interaktionen mit anderen Teilen des Systems (vgl. Kapitel 1)
  - Kommunikationsaufwand im Team (vgl. Kapitel 2)
  - unerwartete Hindernisse (z. B. langwierige Fehlersuche)
- Erkenntnisse aus quantitativen Studien
  - Schätzungen werden im Laufe des Projektes immer schlechter
  - starke Abhängigkeit von der reinen Menge an Code, relativ unabhängig von der Sprache

Kapitel 9: Zehn Pfund in einem Fünf-Pfund-Sack

Wie kann man Code reduzieren?

- Hochsprachen (aber: intrinsische Komplexität)
- Budgetierung von Ressourcen (Code belegt Speicherplatz)
- Fokus auf Datenstrukturen statt auf Algorithmen
  - „Zeige mir Flussdiagramme ohne [Datenbank-]Tabellen, und ich werde verwirrt bleiben. Zeige mir Tabellen, und ich werde die Flussdiagramme wahrscheinlich nicht brauchen; sie werden offensichtlich sein.“

Kapitel 10: Die dokumentarische Hypothese

Für Manager ist gute Dokumentation das zentrale Arbeitswerkzeug sowie Arbeitsergebnis.

- Festhalten von Entscheidungen
  - Aufdecken von Widersprüchen, Denkfehler etc.
- Kommunikation von Entscheidungen
- Erinnern von Entscheidungen
  - Rückschau, Abwägung von Richtungsänderungen

Kapitel 11: Plane damit, eines wegzuwerfen

- Systeme werden obsolet durch…
  - Skalierungsprobleme: Analogie zu ”Pilotanlagen”
  - Änderung der Umstände: z. B. neue Kundenanforderungen
- Maintenance laut Brooks hauptsächlich aufgrund von „Änderungen, die Designfehler beheben“
  - jede Änderung erzeugt potentiell Folgefehler und erhöht Komplexität

Kapitel 12: Scharfe Werkzeuge

- Manager sollen Ressourcen abstellen für *Werkzeugmacher*
  - damals für Debugging-Werkzeuge
  - heute für CI/CD oder Linting
- Brooks wünscht sich stärkere Hochsprachen und mehr interaktive Programmierung

Kapitel 13: Das Ganze und die Teile

Wie erhält man beim Zusammenfügen mehrerer Teilprogramme die konzeptionelle Integrität des Gesamtsystems?

- Niklaus Wirth: Entwicklung durch „schrittweises Verfeinern“
  - Probleme beim Verfeinern entlarven Designfehler auf gröberen Ebenen
- klare Markierung von manuellen Abweichungen vom Plan
- nicht zu viele Änderungen auf einmal

Kapitel 14: Wie man eine Katastrophe ausbrütet

Zeitpläne scheitern aufgrund Tod durch tausend Schnitte.

- Verspätungen erkennen mittels fester Zeitpläne mit spezifischen Meilensteinen
- Aufmerksamkeit lenken mittels Analyse des kritischen Pfades

Kapitel 15: Das andere Gesicht

- Programme sind für Maschinen da (zum Ausführen), aber auch für Menschen (zum Verstehen)
- drei Arten von Dokumentation eines Programms:
  - wie man es verwendet (z. B. Manpage)
  - wie man ihm glaubt (z. B. Testfälle)
  - wie man es verändert (z. B. Codekommentare)
- Brooks präferiert Strukturdiagramme anstelle von Flowcharts, und wünscht sich mehr *Literate Programming* (vgl. Knuth)

Epilog

„Die Teergrube der Software-Entwicklung wird noch eine ganze Weile klebrig bleiben. Man kann davon ausgehen, dass die Menschheit auch weiterhin Systeme in Angriff nehmen wird, die gerade so innerhalb oder gerade so außerhalb des Möglichen liegen; und Softwaresysteme sind vielleicht die vertracktesten und komplexesten aller menschlichen Schöpfungen. Das Verwalten dieses komplexen Handwerks wird von uns fordern: den bestmöglichen Einsatz neuer Sprachen und Systeme, die optimale Anwendung bewährter Methoden des Ingenieurmanagements, großzügige Mengen gesunden Menschenverstandes, und die gottgegebene Demut, die eigene Fehlbarkeit und Beschränkungen zu erkennen.“

Podcast-Empfehlungen

- Gilda con Arne – Der Politik-Podcast
- Dlf Doku Serien
  - einige Trailer für Dokus
  - Tech Bro Topia
  - Rechtsextreme vor Gericht
  - Neuland
- Die Geschichte geht weiter – Victor Klemperers Tagebücher 1918-1958