The Mythical Man-Month (1975-2025)

Planning, Implementing, and Managing Statistical Software Projects

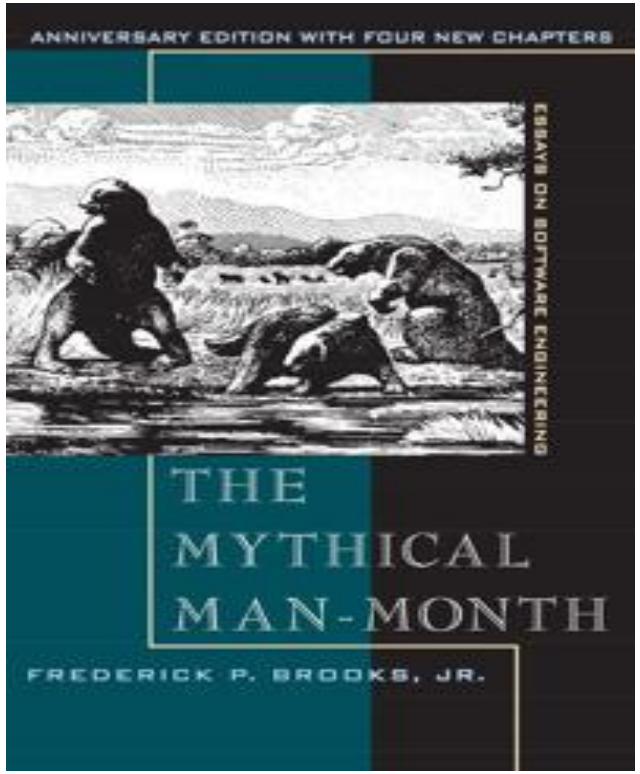
Wilmar Igl, PhD

Veranex

Introduction

- **50th anniversary (1975-2025)**
- "**Bible**" of Software Engineering
- **Overview of the current state-of-the-art** of planning, implementing, and managing software engineering projects
  - established truths
  - novel insights
- **Open-source statistical software** in the pharma industry

Brooks' Law:
"Adding manpower to a late software project makes it later."



Cover with
"Mural of the La Brea Tar Pits"
(C. R. Knight)

50 years of IT Development

IBM System/360 (197x)

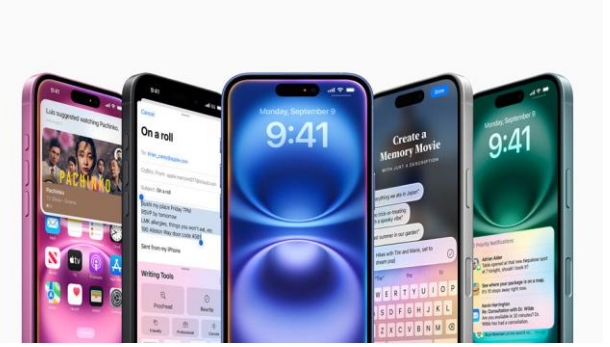
- **1975: Monolithic programs on punch cards for stationary mainframe computers**

…

- **Hardware,** eg, Moore's Law, PCs, Internet via cable/satellite
- **Operating Systems,** eg, Wirth's law, MS Windows, MacOS/iOS, Linux/Android
- **Programming languages,** eg, OOP, FP, Abstract Languages
- **Networking/Internet,** eg, TCP/IP, WWW, XMPP, Cloud, REST APIs
- **Collaboration**, eg DevOps, open-source, eg, Linux, Git(Hub)

…

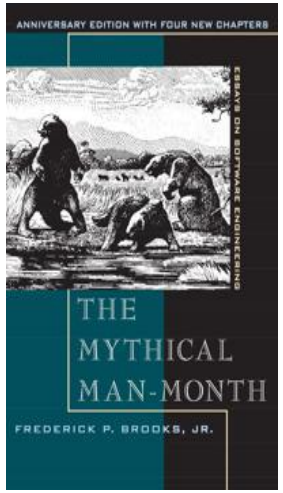
- **2025: Mobile mini-computers (smartphones) communicating with AI-power services on cloud servers via standardized protocols & interfaces**



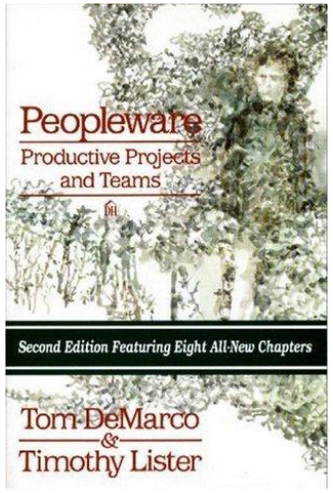
iPhone 16 (2025)

TM3 is still up-to-date

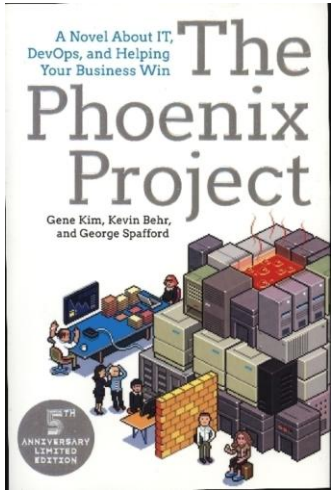
- **Updated 2nd edition (1995)** with 3 additional chapters
- Focus on **people and teams**
- **Style** (holistic, figurative-metaphorical, blunt, open, self-critical)



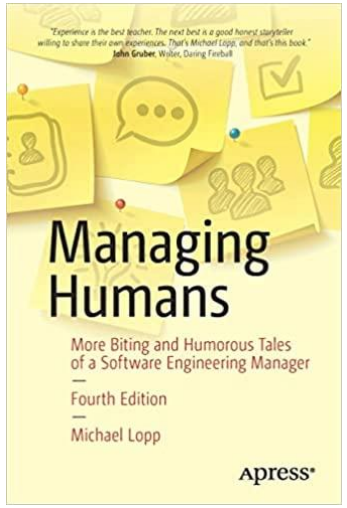
1975/1995



1987/1999



2013/2014



2007/2021

TM3 is out-dated

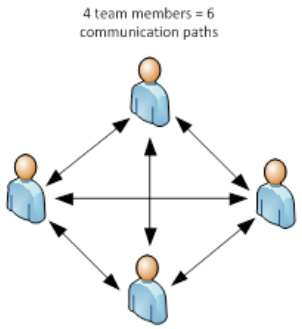
- **Progress of**
  - hardware (cf, Moore's law)
  - software (tools) (cf, Wirth's law, version control, CI/CD)
- **Examples are outdated**
- **Disruptive paradigm shift** by AI,
  eg, Deep Learning, Large Language Models
- **Style** (male-oriented, patriarchal, euro-centric, Christian)
  ("*A team of two, with one leader, is often the best use of minds. [Note God's plan for marriage.]*" Brooks, 1975/1995, p. 232)

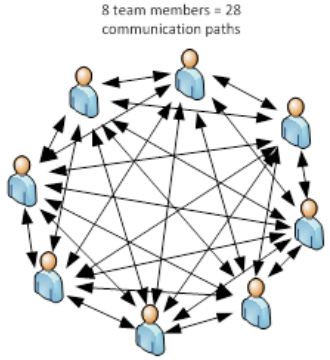
The Mythical Man-Month

- an **artificial unit** of human effort, but not useful for human work
- a **“myth”**, ie a common, but false (& dangerous) idea (cf Brooks law)
- depends on **dependency of subtasks** (cf, complex software)
- **Example**: If a software program is estimated to require **12 man-months**, **365x developers** programming for 1 day is expected to result in a **program of lower quality** (or no functioning program at all) relative to **1x developers** programming for 365 days.
- **Assessment**: still true, but simplistic

Brooks' Law (1)

- *"Adding manpower to a late software project makes it later."*





- Burden of **work & communication,**
  eg, number of communication paths is n(n-1)/2
- Delay depends on **timepoint, personality, process**
- **Extension to "Costs"**
- **Assessment**: true, progress made,
  but often not applied in practice

Brooks' Law (2) – Reasons for delays

- Managers target **too short timelines** (target vs commitment vs estimate)
- **Poor estimation techniques**, e.g. expert opinions
- **Effort confused for progress**, eg, re-work, non-essential features, debugging
- **Poor monitoring** of project progress, eg verbal report at team meetings
- **Inappropriate corrective actions**, eg adding more people
  (vs reducing scope, move timelines, postpone other projects)

Brooks' Law (3) - Rules of thumb

	Brooks (1975/1995)		Orosz (2024)		Jones (2024)
Planning	33%	Planning	20%	Planning	20%
Coding	17%	Coding + Testing	40%	Coding	30%
Testing of components	25%			Testing	25%
Testing of system	25%				
		Code review	20%		
		Release to Production	20%	Release to Production	15%

Orosz, 2024

No Silver Bullet

- ***No single development*** *(technology or management) will result in a* ***10x improvement*** *in productivity, in reliability, in simplicity over the next 10 years* (Brooks, 1986/1995)

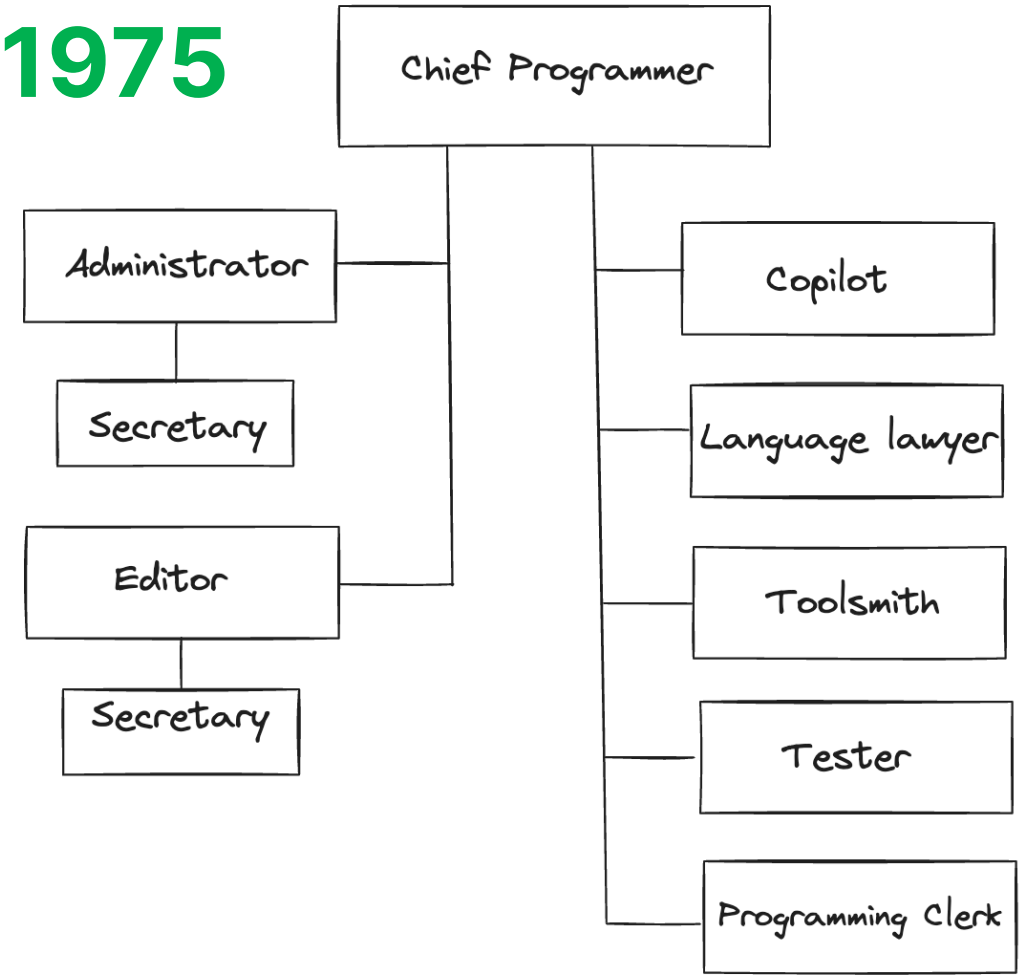
- **Examples:**
  - Multiple small improvements may have a combined effect of a 10-fold improvement (eg, $1.26^{10} \sim 10$)
  - Version control, eg GIT, Subversion
  - Continuous Integration/Continuous Delivery (CI/CD) Model
  - Open-source software development model
  - AI/Deep Learning/Large Language Models

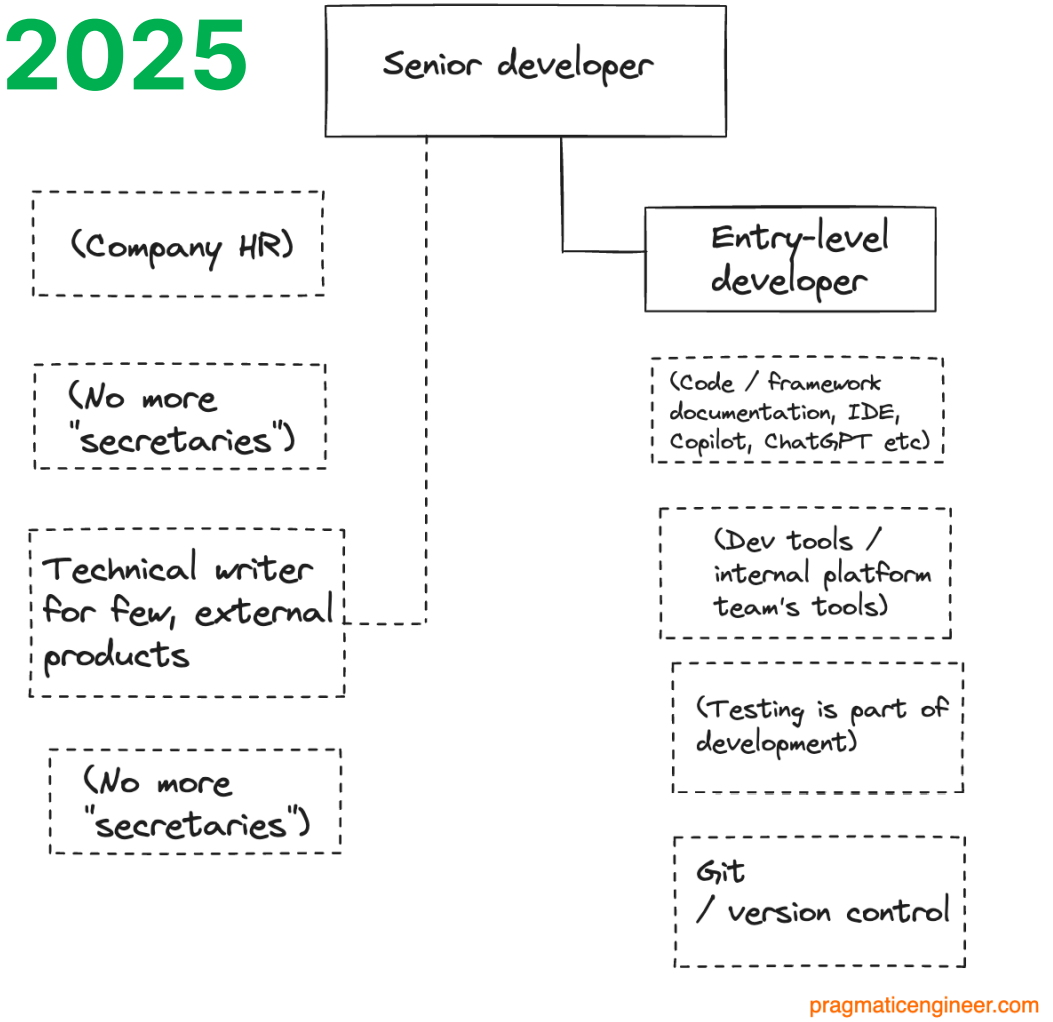
- **Assessment**: true at the time, less relevant today

People – (almost) everything

- The **major problem is sociological,** not technological.
- **Developers team** (> 4x tools) is largest factor for project progress
- **Managers:**
  - Function: to make it possible for people to work
  - Subsidiarity: the power of giving up power
- **Assessment:** still true,
  though tools have improved considerably
  (eg, version control, CI/CD)

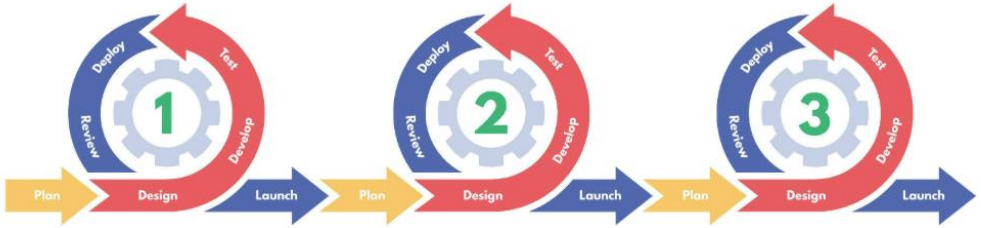
Surgical Team Concept





Progressive Refinement & Central Argument

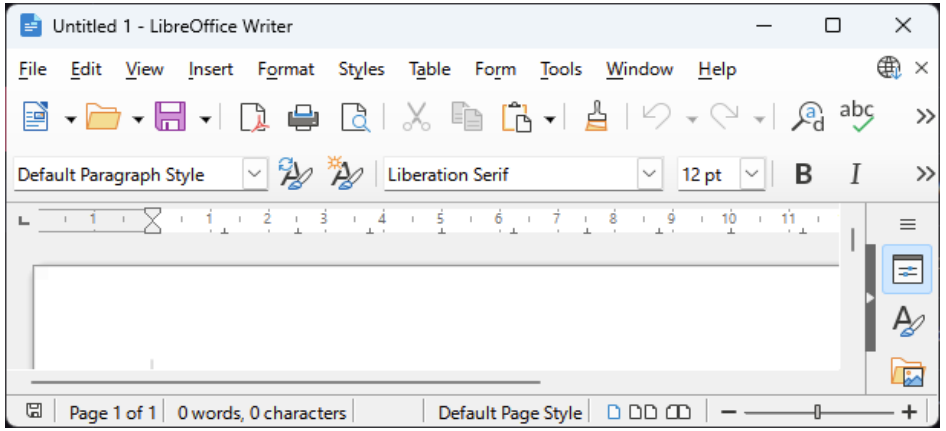
- **Progressive Refinement**
  - End-to-end skeleton system
  - Family of related products
  - Incremental builds



Example: Agile Development

- **The Central Argument**
  - Conceptual Integrity
  - Architecture vs Implementation
  - The Architect
  - Recursion of architects

- **Assessment**: true, though terminology has changed



Example: WIMP Interface

AI tools

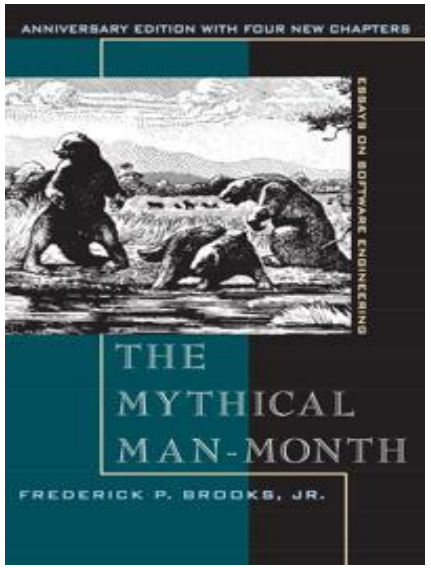
- **Release of Large Language Models**
  (ie, OpenAI/ChatGPT, DeepSeek, MS/CoPilot, Google/Gemini,..)

- **Pros:**
  - **Automatic creation, translation, debugging of programs**
    based on human, natural-language prompts
  - **Impressive performance**

- **Cons:**
  - **Misinterpretation** of prompts, bugs, hallucinations
  - **Additional QC review** of (overly) complex code
  - **Violation of intellectual property rights**

Conclusion

- *“**The Mythical Man-Month**” is still a classic*
  - *some anachronisms*
  - *(simplified) truths*
  - *prophetic insights*



- ***Large progress** in technology and management, but often not applied.*

- **Human personalities and social behaviors** have not fundamentally changed.


THANK YOU
Bedankt
Danke
धन्यवाद
谢谢
ありがとう
Teşekkürler
Bedankt
ευχαριστώ
gracias
СПАСИБО
TERIMA KASIH
MULȚUMESC
MERCI
TEŞEKKÜRLER
DANKE
Спасибо
GRATIAS TIBI
ΕΥΧΑΡΙΣΤΩ
HVALA VAM
BEDANKT
متشكرم
GRATIAS TIBI
Hvala vam
Teşekkürler
BEDANKT
MULȚUMESC
TERIMA KASIH
ขอบคุณ
გმადლობთ
THANK YOU

References (1) – Books

- Brooks, F. (1975/1995). The Mythical Man-month: Essays on Software Engineering. Addison-Wesley.
- McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction.
- Cohn, M. (2005). Agile Estimating and Planning. Prentice Hall.
- McConnell, S. (2006). Software Estimation: Demystifying the Black Art. Microsoft Press US.
- Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Pearson.
- Kim, G., Behr, K., & Spafford, G. (2013). The Phoenix Project. IT Revolution Press.
- DeMarco, T., & Lister, T. (2016). Peopleware: Productive Projects and Teams (3rd ed.). Addison Wesley.
- Thomas, D., & Hunt, A. (2019). The Pragmatic Programmer: Your journey to mastery (2nd ed.). Addison-Wesley.
- Kim, G., et al. (2021). The DevOps Handbook (2nd ed.). IT Revolution Press.
- Lopp, M. (2021). Managing Humans: More Biting and Humorous Tales of a Software Engineering Manager (4th ed.). Apress. Link: https://managinghumans.com
- Farley, D. (2021). Modern Software Engineering: Doing What Works to Build Better Software Faster.
- Larson, W. (2019). An Elegant Puzzle: Systems of Engineering Management. Stripe Press.
- Janson, S. (2024). Agile Project Management. Best of HR – Berufebilder.de.

References (2) – Articles & Blogs

- Orosz, G. (2024). What Changed in 50 Years of Computing: Part 1? https://newsletter.pragmaticengineer.com/p/what-changed-in-50-years-of-computing
- Orosz, G. (2024). What Changed in 50 Years of Computing: Part 2? https://newsletter.pragmaticengineer.com/p/what-changed-in-50-years-of-computing-8d0
- Orosz, G (2023). The productivity impact of AI coding tools, https://newsletter.pragmaticengineer.com/p/ai-coding-tools
- Zheng, Z., et al. (2025). Towards an understanding of large language models in software engineering tasks. Empir Software Eng 30, 50 (2025). https://doi.org/10.1007/s10664-024-10602-0
- Xinyi Hou, et al. (2024). Large Language Models for Software Engineering: A Systematic Literature Review. ACM Trans. Softw. Eng. Methodol. 33, 8, Article 220 (November 2024), 79 pages. https://doi.org/10.1145/3695988, https://arxiv.org/abs/2308.10620
- Ozkaya, I. et al. (2023). Application of Large Language Models (LLMs) in Software Engineering: Overblown Hype or Disruptive Change? https://insights.sei.cmu.edu/blog/application-of-large-language-models-llms-in-software-engineering-overblown-hype-or-disruptive-change/

References (3) – Articles & Blogs

- Fan, A. *et al*., "Large Language Models for Software Engineering: Survey and Open Problems," *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, Melbourne, Australia, 2023, pp. 31-53, doi: 10.1109/ICSE-FoSE59343.2023.00008. https://ieeexplore.ieee.org/document/10449667

- Fallman, D. for Forbes Technology Council (2024). How To Leverage Large Language Models For Engineering And More. Forbes Magazine. https://www.forbes.com/councils/forbestechcouncil/2024/03/06/how-to-leverage-large-language-models-for-engineering-and-more/

- Lopp, M. (2005). The single most productive engineer - Free Electron, https://randsinrepose.com/archives/free-electron/

Veranex