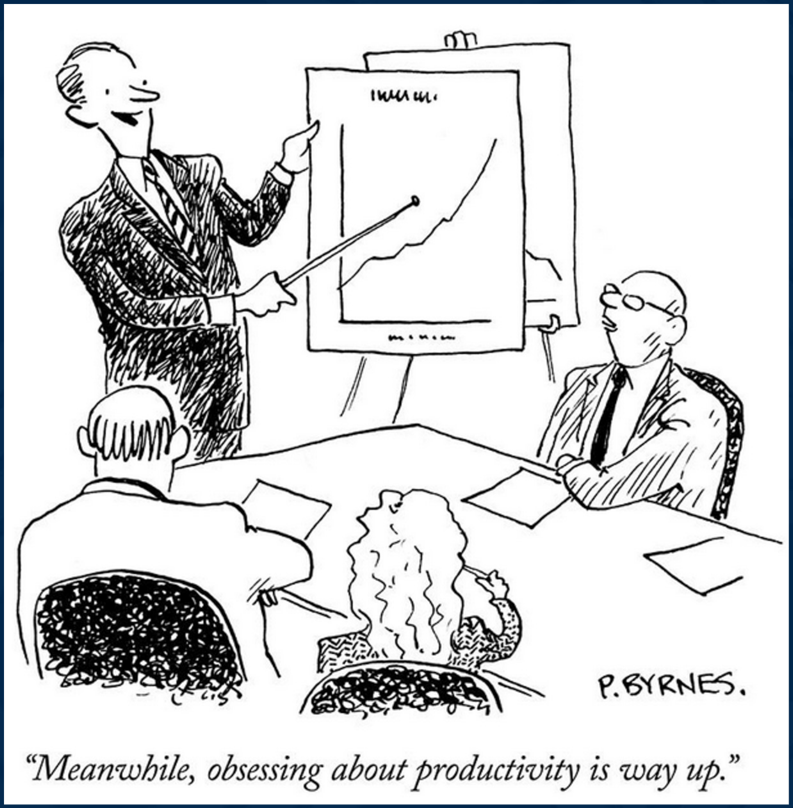
MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Productivity

EECS 481 (W24)



"Meanwhile, obsessing about productivity is way up."

The Story so far…

- We want to deliver and support a quality software product
- Software processes are carried out by humans
  - Humans have biases
- Some humans are more productive than others at software engineering activities
  - How can we understand and improve such human expertise?

One-Slide Summary

- Humans demonstrate different levels of **expertise** (i.e., different productivity rates) at programming tasks.
- We consider a number of hypotheses, including hardware support, slow programmers and programs, abstractions, decompositions, and neural activity. For each, we examine relevant scientific literature.
- Organizations can provide hardware support. Individuals can practice abstractions and decompositions.

Outline, Psychology

- Real-Time Exercise
- Reading Discussion
    - Rapid Response Time
    - Programming Performance
    - Mythical Man-Month
    - Expertise in Problem Solving
    - Expert Bodies, Expert Minds
- Advice

Learning Objectives: by the end of today's lecture, you should be able to…

1. (*knowledge*) explain how hardware affects productivity

2. (*knowledge*) explain how experts and novices approach problem-solving

3. (*knowledge*) explain why adding more people to the project does not always work

Productivity in Software Engineering

Productivity in Software Engineering is a measure of how efficiently and effectively software developers can produce high-quality software products. Productivity can be influenced by various factors, such as:

- The characteristics of the software product, such as its size, complexity, requirements, and quality standards.
- The characteristics of the software process, such as its methodology, tools, techniques, and practices.
- The characteristics of the software development environment, such as its hardware, software, network, and infrastructure.

Productivity in Software Engineering (Cont'd)

- The characteristics of the corporate culture, such as its vision, mission, values, and policies.
- The characteristics of the team culture, such as communication, collaboration, coordination, and cohesion.
- The characteristics of the individual developers, such as their skills, experiences, motivations, and preferences.
- The characteristics of the work environment, such as its physical, social, and psychological aspects.
- The characteristics of the individual project, such as its scope, duration, budget, and stakeholders.

Productivity in Software Engineering (Cont’d)

Different factors may have different impacts on productivity depending on the context and the situation.

- Some factors may have positive effects, such as completing tasks, working with few interruptions, and being happy and satisfied.
- Some factors may have negative effects, such as encountering errors, having meetings, and facing violence and oppression.
- Some factors may have mixed or uncertain effects, such as using existing or new technologies, working remotely or in-person, and having diverse or homogeneous teams.

Improving Productivity in Software Engineering

To improve productivity in software engineering, it is important to understand the factors that affect it and to apply appropriate methods and measures to optimize them. Some of the methods and measures that can help to improve productivity are:

- Using standards-based and interoperable technologies that can ensure compatibility and scalability of the software products and systems.
- Implementing network security and resilience measures that can protect the software products and systems from unauthorized access, tampering, or disruption.

Improving Productivity in Software Engineering (Cont'd)

- Using human-centered methods that can measure productivity from the perspective of the developers and the users, and that can involve them in the design and evaluation of the software products and systems.
- Using biometric sensors that can monitor the physiological and psychological states of the developers and provide feedback and support for their well-being and performance.
- Using team awareness tools can enhance the communication and coordination of the developers and provide visibility and transparency of their work and progress.

https://web.eecs.umich.edu/~movaghar/The_Effect_of_Work_Environments_on_Productivity_and_Satisfaction_of_Software_Engineers%20IEEE-TSE%202021.pdf

https://web.eecs.umich.edu/~movaghar/Software%20Productivity%202019.pdf

https://web.eecs.umich.edu/~movaghar/What_Predicts_Software_Developers_Productivity.pdf

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Real-Time Exercise

YOU EVER HEARD OF 8 MINUTE ABS?
THIS IS GOING TO BLOW IT OUT OF THE WATER...
7. MINUTE. ABS.

Real-Time Exercise

https://dijkstra.eecs.umich.edu/eecs483/shibboleth/productivity/

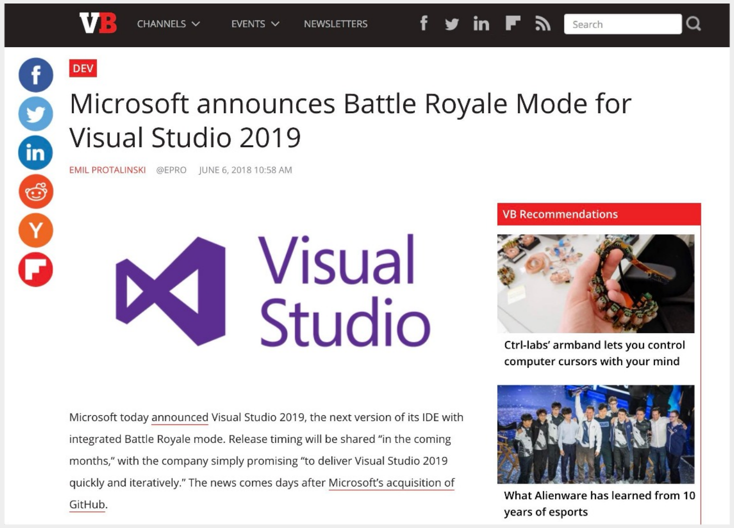
You will be asked to solve a simple problem.

  - ***Get the correct answer as quickly as possible.***
  - ***This counts as the Participation if you submit an answer and explanation by midnight.***

- You will be timed (once you click “start”).
- You can use any program, language or tool available to you.
- Once you have submitted your answer, you must briefly explain what you did.
- I will cut things off after ~10 minutes.

Distribution Times

- How many different tasks were students given?
- What did you observe, roughly,
  as the range and variance of times?



DEV

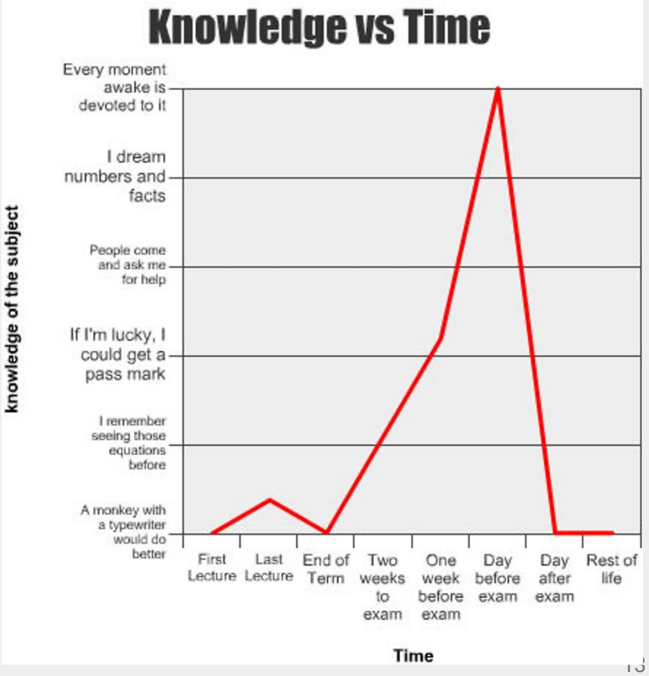
Microsoft announces Battle Royale Mode for Visual Studio 2019

EMIL PROTALINSKI @EPRO JUNE 6, 2018 10:58 AM

Microsoft today announced Visual Studio 2019, the next version of its IDE with integrated Battle Royale mode. Release timing will be shared “in the coming months,” with the company simply promising “to deliver Visual Studio 2019 quickly and iteratively.” The news comes days after Microsoft’s acquisition of GitHub.

Ctrl-labs’ armband lets you control computer cursors with your mind

What Alienware has learned from 10 years of esports

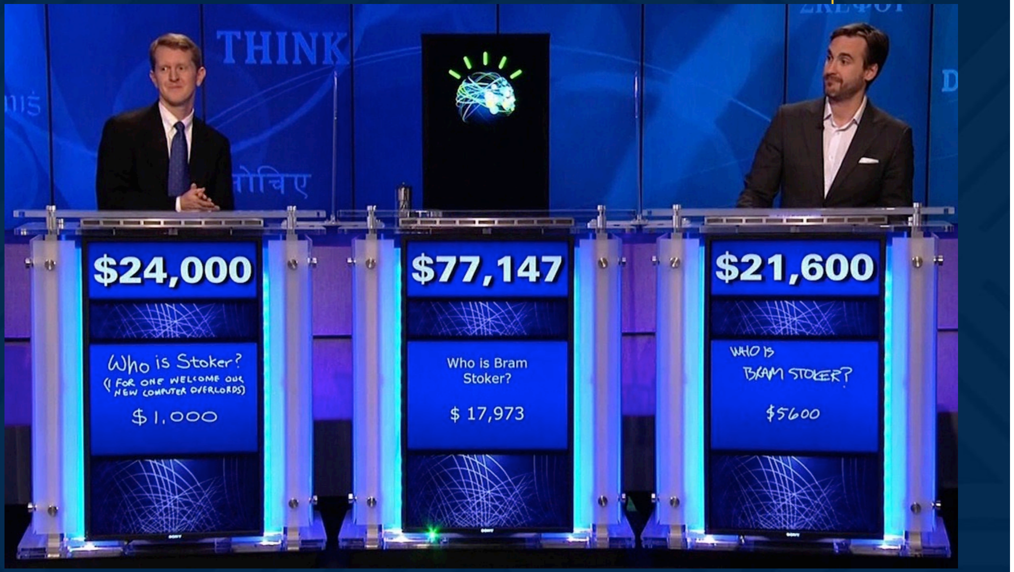


Hypothesis

- My computer is slow.
- I'm slow and so is my program.
- I picked the wrong language/abstraction and couldn't break up the problem.
- I did not recognize the true components of the problem.
- My brain is currently inefficient, requiring much metabolism for little neural activation.

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Rapid Response Time



Rapid Response Time in Customer Service

- In customer service, Rapid Response Time is the average time it takes for a business to answer customer queries or complaints.
- Customers expect fast and helpful solutions, and businesses that provide them can earn customer loyalty and satisfaction.
- Some ways to improve customer service response time are setting goals, collecting feedback, providing self-service resources, and using automation tools.

Rapid Response Time

- The concept of Rapid Response Time in software engineering was first introduced by IBM in the 1980s as part of the Rapid Application Development (RAD) model. This model emphasized the importance of quick development cycles and the ability to rapidly respond to user requirements.
- The RAD model was further developed and formalized by James Martin when he published a book on the subject in 1991.
- The concept of Rapid Response Time is integral to various software development methodologies that prioritize speed and flexibility in response to changing requirements.

Rapid Response Time in Software Engineering

Rapid Response Time in software engineering is a measure of how quickly a software system or application can respond to user requests or inputs. It is an important aspect of software quality, usability, and performance.

Rapid Response Time can also refer to a development methodology that aims to deliver software products faster and with fewer defects by using iterative and incremental processes, such as Rapid Application Development (RAD).

Rapid Response Time in SE (Cont’d)

Some factors that affect Rapid Response Time in software engineering are:

- The complexity and size of the software system or application
- The design and architecture of the software system or application
- The programming language and tools used to develop the software system or application
- The hardware and network resources available to run the software system or application
- The user expectations and requirements for the software system or application

Ways to improve Rapid Response Time in SE

- Using agile and lean principles and practices to deliver software products in small and frequent increments
- Applying design patterns and best practices to reduce coupling and increase cohesion among software components
- Using code analysis and testing tools to identify and fix performance bottlenecks and bugs
- Optimizing the code and algorithms to reduce the computational and memory costs
- Scaling the software system or application horizontally or vertically to handle increased workload and demand

Examples of Rapid Response Time in SE

- In customer service, a web application that provides dynamic and personalized content to users should respond within 1 second to keep the user's flow of thought uninterrupted

https://stackoverflow.com/questions/164175/what-is-considered-a-good-response-time-for-a-dynamic-personalized-web-applicat

- In the military, a software system that coordinates rapid reaction forces should respond within 0.1 seconds to make the user feel that the system is reacting instantaneously

Examples of Rapid Response Time in SE (Cont’d)

- In food safety, a software application that tracks and controls foodborne outbreaks should respond within 10 seconds to keep the user's attention focused on the dialogue.
- In workforce development, a software program that helps workers transition to new jobs should respond within a short time frame (usually 60-90 days) to provide reemployment services to the affected workers

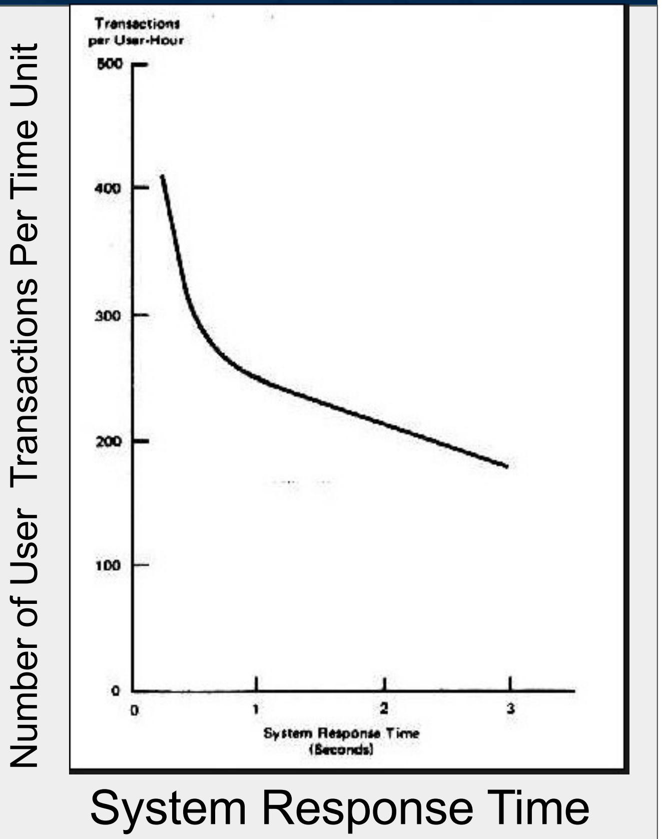
https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/

Rapid Response Time

- Walter Dougherty and Ahrvind Thadani. *The Economic Value of Rapid Response Time*. IBM Systems Journal, 1982.
  - Read chart "backward", from Right to Left.
  - Productivity goes up, then sharply up.

Relationship Between System Response Time and the Number of Transactions a User Can Complete in an Hour

https://jlelliotton.blogspot.com/p/the-economic-value-of-rapid-response.html



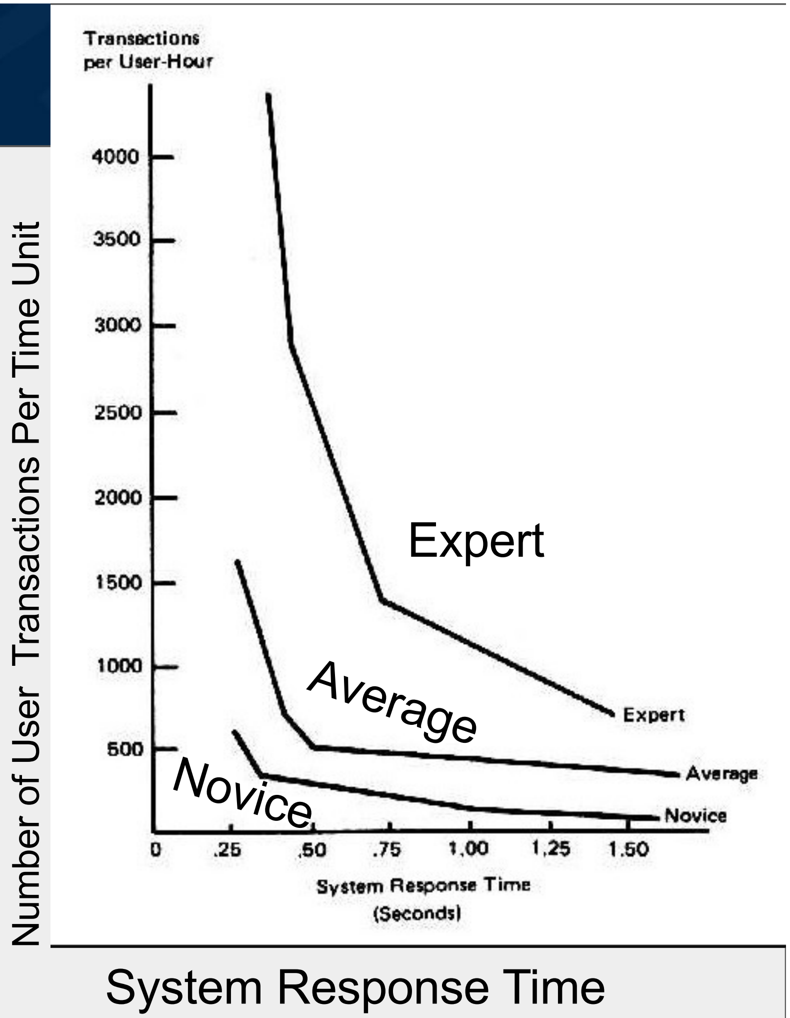
Rapid Response Time

- "...each second of system response degradation leads to a similar degradation added to the user's time for the following [command]. This phenomenon seems to be related to an individual's attention span. The traditional model of a person thinking after each system response appears to be inaccurate. Instead, people seem to have a sequence of actions in mind, contained in a short-term mental memory buffer. Increases in SRT [system response time] seem to disrupt the thought processes, and this may result in having to rethink the sequence of actions to be continued."

Rapid Response Time

High Function Graphics, Transaction Rate versus System Response Time

Figure 7

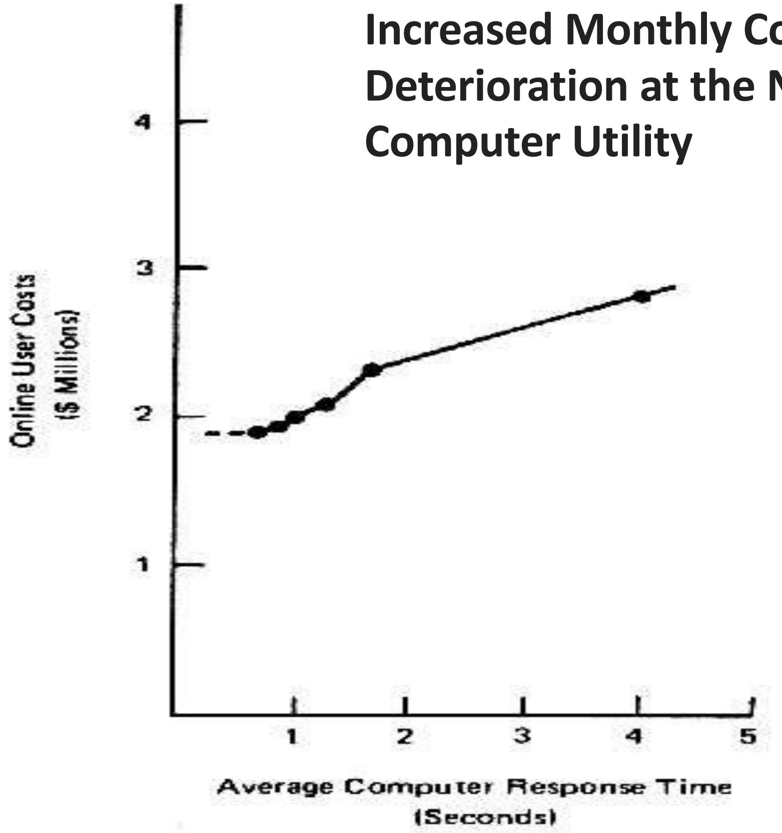


Rapid Response Time

- The System Products Division (SPD) study measured 75 work sessions of 15 engineers at graphic display terminals as they performed various physical design tasks. Their transaction rate data confirmed Thadhani’s curve, (Figure 7). Indeed, it showed considerably more. All users benefited from sub-second response time. In addition, on average, an experienced engineer working with a sub-second response was as productive as an expert with a slower response. A novice's performance became as good as the experienced professional and the productivity of the expert was dramatically enhanced.

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

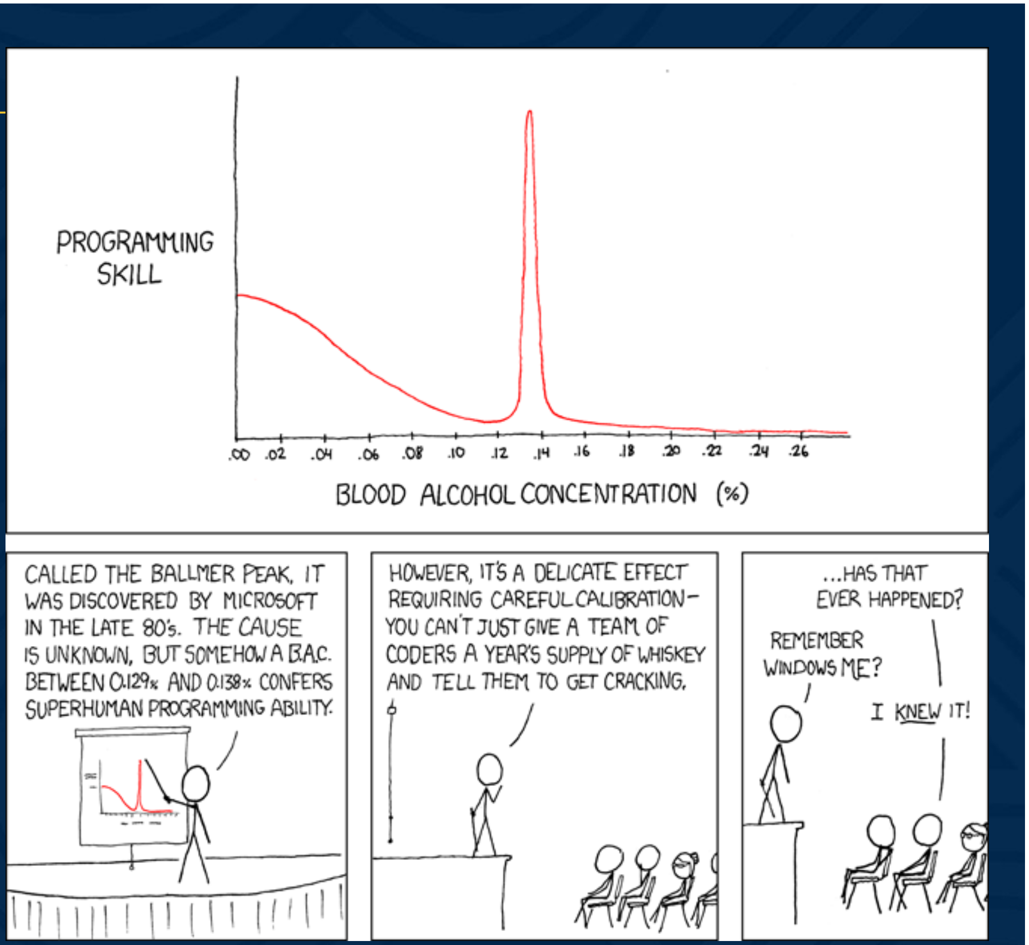
Increased Monthly Costs with Response Time Deterioration at the National Institutes of Health Computer Utility



Rapid Response Time

- Example implication, from the reading:
  - "The system and user cost for this time were estimated at $900,000 monthly (Figure 6), 15 times the incremental cost of a new processor capable of providing sub-second response time to 500 simultaneous users. For the National Institutes of Health, the cost of upgrading their processor was more than justified by the savings in user time and the restoration of their low task costs.
  - The engineers use display terminals specifically designed for the high transaction rates necessary to manipulate graphic images."

Programming Performance


PROGRAMMING SKILL
.00 .02 .04 .06 .08 .10 .12 .14 .16 .18 .20 .22 .24 .26
BLOOD ALCOHOL CONCENTRATION (%)
CALLED THE BALLMER PEAK, IT WAS DISCOVERED BY MICROSOFT IN THE LATE 80's. THE CAUSE IS UNKNOWN, BUT SOMEHOW A B.A.C. BETWEEN 0.129% AND 0.138% CONFERS SUPERHUMAN PROGRAMMING ABILITY.
HOWEVER, IT'S A DELICATE EFFECT REQUIRING CAREFUL CALIBRATION– YOU CAN'T JUST GIVE A TEAM OF CODERS A YEAR'S SUPPLY OF WHISKEY AND TELL THEM TO GET CRACKING.
...HAS THAT EVER HAPPENED?
REMEMBER WINDOWS ME?
I KNEW IT!

Programmer Performance in Software Engineering

Programmer performance in software engineering is a measure of how well a programmer can write, test, debug, and maintain software code.

It is an important aspect of software quality, productivity, and efficiency.

Metrics for Programmer Performance

- Code quality: the degree to which the software code meets the functional and non-functional requirements, follows the coding standards and best practices, and is readable, maintainable, and reusable.
- Code quality can be measured by tools such as code analyzers, code reviewers, and code coverage tools, which can detect and report code issues, such as bugs, errors, vulnerabilities, complexity, duplication, and style violations

https://blog.pragmaticengineer.com/performance-reviews-for-software-engineers/

https://www.effy.ai/blog/developer-performance-review

Metrics for Programmer Performance

- Code productivity: the amount of software code that a programmer can produce or modify in a given time frame
- Code productivity can be measured by tools such as code counters, code trackers, and code estimators, which can calculate and report code metrics, such as lines of code, function points, cyclomatic complexity, and code churn

https://www.shakebugs.com/blog/kpi-software-development/

https://insights.sei.cmu.edu/blog/programmer-moneyball-challenging-the-myth-of-individual-programmer-productivity/

.

Metrics for Programmer Performance

- Code efficiency: the degree to which the software code can perform its intended functions and tasks with optimal use of resources, such as CPU, memory, disk, network, and power
- Code efficiency can be measured by tools such as code profilers, code optimizers, and code benchmarkers, which can analyze and report code performance, such as execution time, memory usage, disk space, bandwidth, and power consumption

https://youteam.io/blog/software-engineer-performance-review-the-best-process-and-metrics/

How to Improve Programmer Performance?

- Performance feedback: the process of providing and receiving constructive and timely feedback on the software code and the programming skills.
- Performance feedback can be done by using tools such as code review platforms, code collaboration tools, and code feedback systems, which can facilitate and automate code reviews, code comments, code suggestions, and code ratings

How to Improve Programmer Performance?

- Performance coaching: the process of mentoring and guiding a programmer to improve their software code and their programming skills.
- Performance coaching can be done by using tools such as code learning platforms, code mentoring tools, and code coaching systems, which can offer and deliver code courses, code challenges, code exercises, code quizzes, and code tips

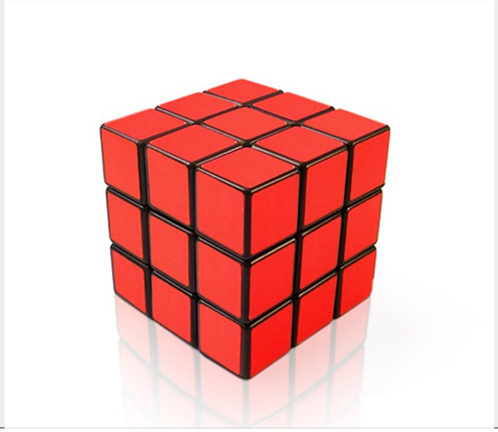
How to Improve Programmer Performance?

- Performance recognition: the process of acknowledging and rewarding a programmer for their software code and their programming skills
- Performance recognition can be done by using tools such as code recognition platforms, code reward tools, and code gamification systems, which can create and manage code badges, code points, code levels, code leaderboards, and code rewards

Programming Performance

- H. Sackman, W. J. Erikson and E. E. Grant. *Exploratory Experimental Studies Comparing Online and Offline Programming Performance.* Communication of the ACM, 1968.
- Two exploratory experiments were conducted at System Development Corporation to compare the debugging performance of programmers working under conditions of online and offline access to a computer. These were the first known studies that measured programmers' performance under controlled conditions for standard tasks.

https://web.eecs.umich.edu/~movaghar/Sackman 1968.pdf



- Summary?

Programming Performance

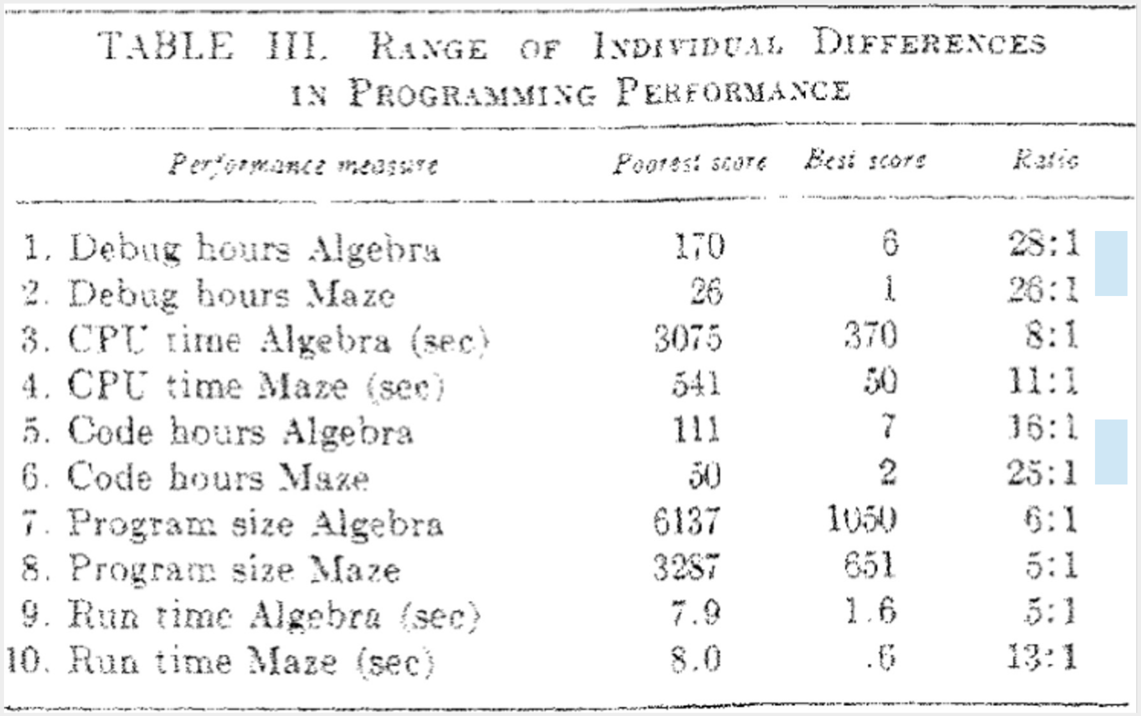


TABLE III. RANGE OF INDIVIDUAL DIFFERENCES IN PROGRAMMING PERFORMANCE

Performance measure	*Poorest score*	*Best score*	*Ratio*
1. Debug hours Algebra	170	6	28:1
2. Debug hours Maze	26	1	26:1
3. CPU time Algebra (sec)	3075	370	8:1
4. CPU time Maze (sec)	541	50	11:1
5. Code hours Algebra	111	7	16:1
6. Code hours Maze	50	2	25:1
7. Program size Algebra	6137	1050	6:1
8. Program size Maze	3287	651	5:1
9. Run time Algebra (sec)	7.9	1.6	5:1
10. Run time Maze (sec)	8.0	.6	13:1

Programming Performance

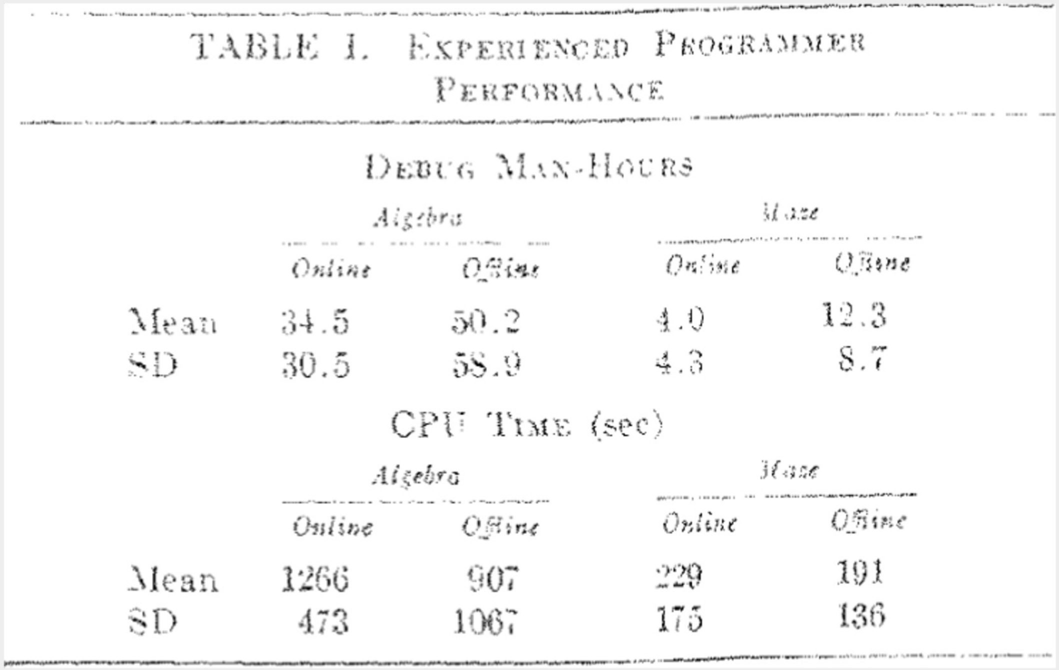


TABLE I. EXPERIENCED PROGRAMMER PERFORMANCE

DEBUG MAN-HOURS

	Algebra		Maze	
	Online	Offline	Online	Offline
Mean	34.5	50.2	4.0	12.3
SD	30.5	58.9	4.3	8.7

CPU TIME (sec)

	Algebra		Maze	
	Online	Offline	Online	Offline
Mean	1266	907	229	191
SD	473	1067	175	136

Programming Performance

- A substantial performance factor designated as "programming speed," associated with faster coding and debugging, less CPU time, and the **use of a higher order language**.
  - WRW: This is new, but not the whole story.
- A well-defined "program economy" factor marked by shorter and faster running programs, associated to some extent with greater programming experience and with the use of machine language rather than higher order language.
  - WRW: Similar explanation to the previous paper.

Programming Performance

- “Data were gathered on the subject's grades in the SDC programmer training class … and they were also given the Basic Programmer Knowledge Test. Correlations between all experimental measures, adjusted scores, grades, and the BPKT results were determined. … The results showed no consistent correlation between performance measures and the various grades and test scores.”

Programming Performance

- “It is apparent from the spread of the data that very substantial savings can be effected by successfully detecting low performers. Techniques measuring individual programming skills should be vigorously pursued ...”

- Why do CS companies use Skill-Based Interviews instead of just using your class grades?
  - See other lecture!

Fault Localization Accuracy

- Zachary P. Fry, Westley Weimer: *A Human Study of Fault Localization Accuracy*. International Conference on Software Maintenance (ICSM) 2010

https://web.eecs.umich.edu/~movaghar/A_human_study_of_fault_localization_accuracy.pdf

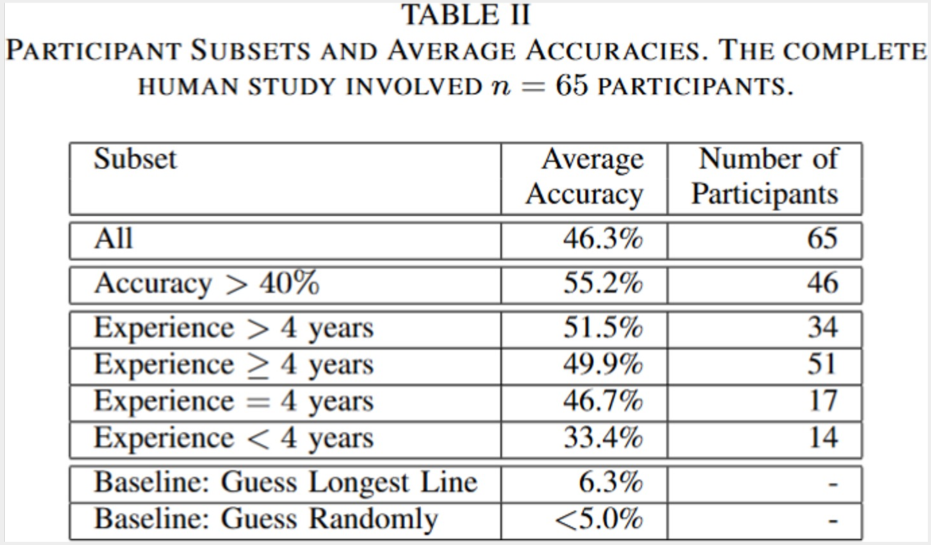


TABLE II
PARTICIPANT SUBSETS AND AVERAGE ACCURACIES. THE COMPLETE HUMAN STUDY INVOLVED $n = 65$ PARTICIPANTS.

Subset	Average Accuracy	Number of Participants
All	46.3%	65
Accuracy $> 40\%$	55.2%	46
Experience > 4 years	51.5%	34
Experience ≥ 4 years	49.9%	51
Experience $= 4$ years	46.7%	17
Experience < 4 years	33.4%	14
Baseline: Guess Longest Line	6.3%	-
Baseline: Guess Randomly	<5.0%	-

The Mythical “Man” Month

I HIRED ALL OF YOU BECAUSE THE PROJECT WILL TAKE 300 MAN DAYS TO COMPLETE.
scottadams@aol.com
www.dilbert.com
THERE ARE 300 OF YOU, SO I WANT YOU TO FINISH BY FIVE O'CLOCK AND CLEAN OUT YOUR DESKS. YOU'RE ALL FIRED.
©2007 Scott Adams, Inc./Dist. by UFS, Inc.
IF IT TAKES MORE THAN ONE MEETING TO MANAGE A PROJECT, I LOSE INTEREST.

© Scott Adams, Inc./Dist. by UFS, Inc.

Fred Brooks

- **Frederick Phillips Brooks Jr.** (April 19, 1931 – November 17, 2022) was an American computer architect, software engineer, and computer scientist, best known for managing the development of IBM's System/360 family of computers and the OS/360 software support package, then later writing candidly about those experiences in his seminal book *The Mythical Man-Month*.
- In 1976, Brooks was elected to the National Academy of Engineering for "contributions to computer system design and the development of academic programs in computer sciences".
- Brooks received many awards, including the National Medal of Technology in 1985 and the Turing Award in 1999.

The Mythical Man-Month: Essays on Software Engineering

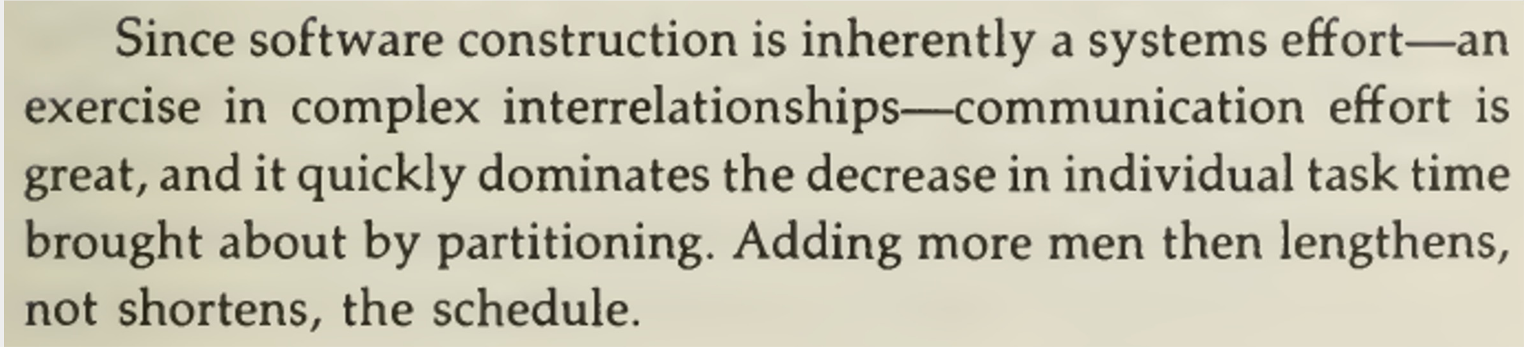
- The Mythical Man-Month: Essays on Software Engineering is a book on software engineering and project management by Fred Brooks first published in 1975, with subsequent editions in 1982 and 1995.
- Its central theme is that adding manpower to a software project that is behind schedule delays it even longer.
- This idea is known as Brooks's law and is presented along with the second-system effect and advocacy of prototyping.
- Brooks's observations are based on his experiences at IBM while managing the development of OS/360.

https://en.wikipedia.org/wiki/The_Mythical_Man-Month

The Mythical Man-Month

- Frederick Brooks. The Mythical Man-Month. Addison-Wesley, 1975/1995.

- Summary?



Since software construction is inherently a systems effort—an exercise in complex interrelationships—communication effort is great, and it quickly dominates the decrease in individual task time brought about by partitioning. Adding more men then lengthens, not shortens, the schedule.

The Mythical Man-Month

- Brooks: SE is non-partitionable.

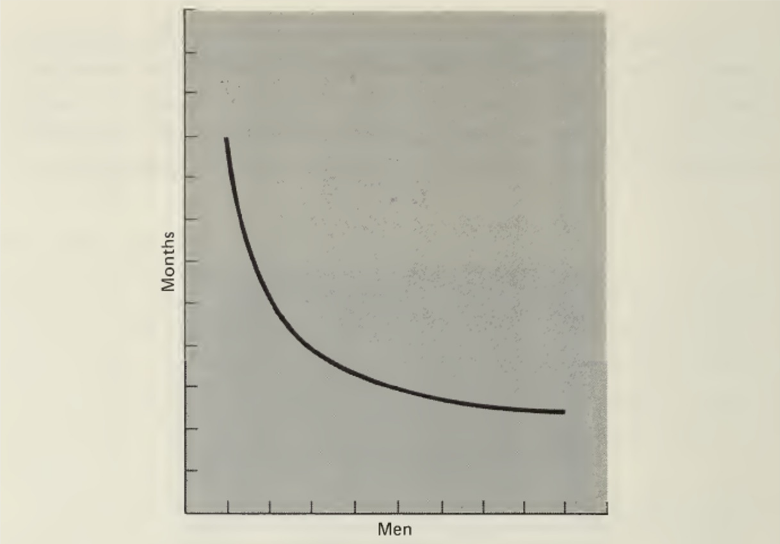


Fig. 2.3 Time versus number of workers—partitionable task requiring communication

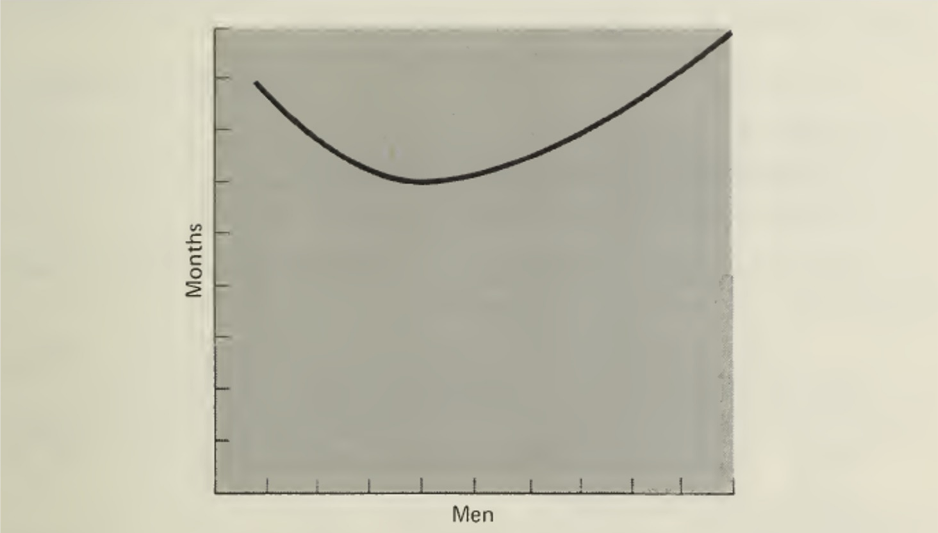
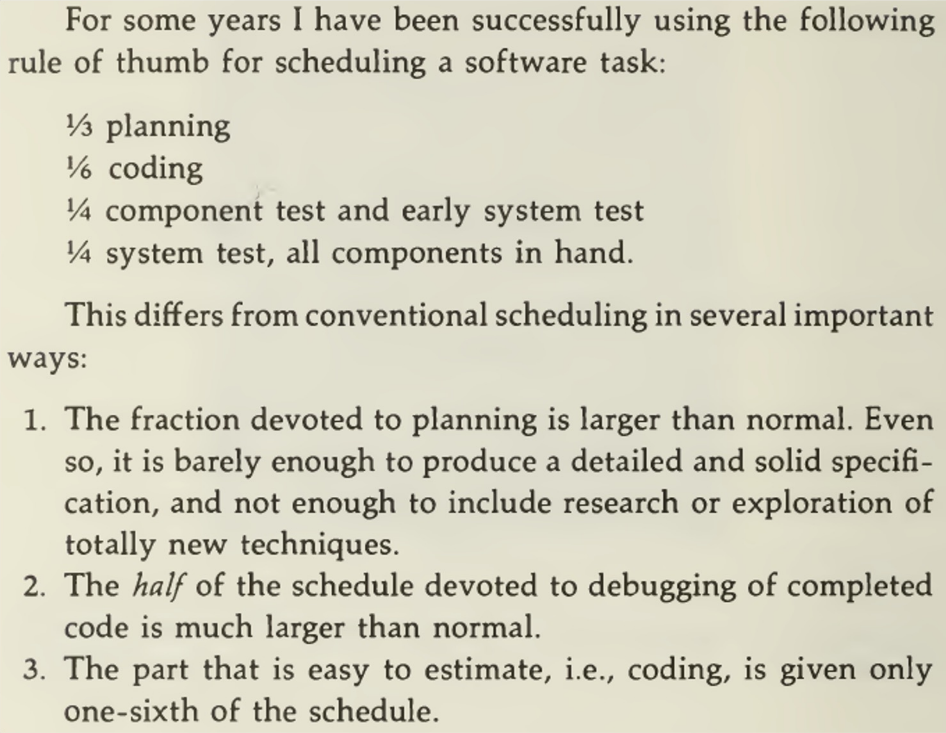


Fig. 2.4 Time versus number of workers—task with complex interrelationships

The Mythical Man-Month



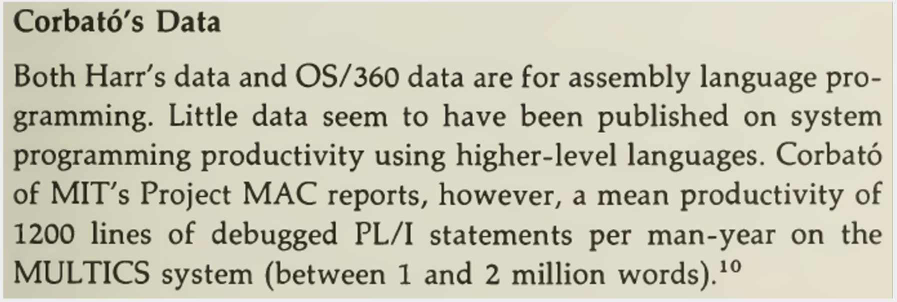
For some years I have been successfully using the following rule of thumb for scheduling a software task:

⅓ planning
⅙ coding
¼ component test and early system test
¼ system test, all components in hand.

This differs from conventional scheduling in several important ways:

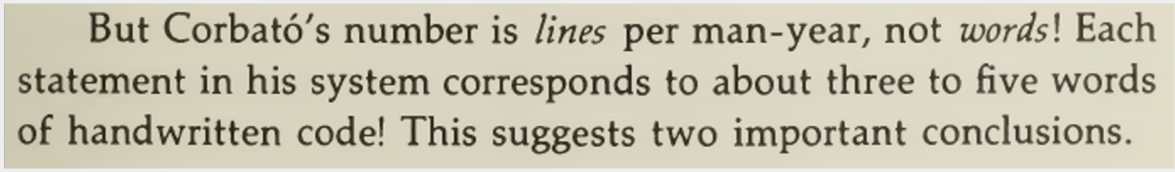
1. The fraction devoted to planning is larger than normal. Even so, it is barely enough to produce a detailed and solid specification, and not enough to include research or exploration of totally new techniques.
2. The *half* of the schedule devoted to debugging of completed code is much larger than normal.
3. The part that is easy to estimate, i.e., coding, is given only one-sixth of the schedule.

The Mythical Man-Month

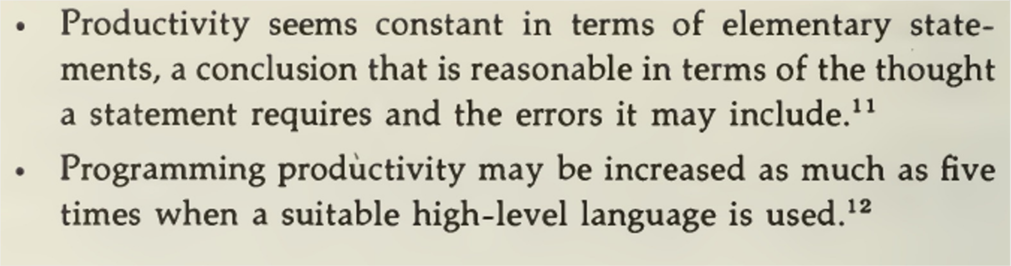


Corbató's Data

Both Harr's data and OS/360 data are for assembly language programming. Little data seem to have been published on system programming productivity using higher-level languages. Corbató of MIT's Project MAC reports, however, a mean productivity of 1200 lines of debugged PL/I statements per man-year on the MULTICS system (between 1 and 2 million words).[10]



But Corbató's number is *lines* per man-year, not *words*! Each statement in his system corresponds to about three to five words of handwritten code! This suggests two important conclusions.



- Productivity seems constant in terms of elementary statements, a conclusion that is reasonable in terms of the thought a statement requires and the errors it may include.[11]
- Programming productivity may be increased as much as five times when a suitable high-level language is used.[12]

The Mythical Man-Month

- 1200 lines / year = 3 lines of code per day
  - ***What?***
- Recall: “debugged code”
  - This includes coding, testing, debugging, etc.
  - Basically the entire software lifecycle
- More modern estimates: 10 LOC / day
- The real insight is the observation of **language invariance**.
  - You can get 10 lines of ASM or 10 lines of Python.

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Trivia Break

Trivia: Names

- Originally called Catholepistemiad, this institution was established in 1817. Its board of regents was formed later in 1837. However, a local justice called that name “neither Greek, Latin, nor English, [but merely] a piece of language gone mad.” At a speech there in 1960, President Kennedy announced his intention to establish the Peace Corps.

Trivia: Poetry

- Name the reclusive American poet and Amherst graduate associated with these works:
  - Because I could not stop for Death
    He kindly stopped for me
  - I'm nobody! Who are you?
    Are you nobody, too?
  - Tell all the Truth but tell it slant —
    Success in Circuit lies
  - My Life had stood — a Loaded Gun —
    In Corners — till a Day

Trivia: Gaming Metrics

- This term refers to the rate at which video game players can select units or otherwise issue orders. It is primarily associated with real-time strategy and fighting games such as StarCraft; a high value for this metric is associated with skill and expertise:
  - Beginner: ~50
  - Professional: ~300
  - Competition: ~400+

Trivia: Cuisine

- This fresh cheese is common in South Asia, especially in India. It is a non-melting, acid-set farmer cheese made by curdling heated milk with lemon juice or vinegar or yogurt, separating out the excess water, and cooling. It is commonly used in dishes in India, Nepal, Bangladesh and Pakistan.

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Expertise in Problem Solving

THERE'S A FINE LINE BETWEEN MAINTAINING SOMETHING AND HAVING TO FIX IT... FOR EVERYTHING IN BETWEEN THERE'S :
DUCT TAPE

Expertise in Problem-Solving

- Expertise in problem-solving is the ability to solve complex and novel problems efficiently and effectively
- Expertise in problem-solving is not a fixed or innate trait, but a dynamic and learnable one.
- It can be developed and improved by engaging in deliberate practice, receiving feedback, and reflecting on one's own problem-solving process

Effects of Expertise in Problem-Solving

- It enhances the performance and productivity of individuals and organizations by enabling them to achieve their goals and overcome challenges
- It improves the quality and creativity of solutions by allowing them to generate more ideas, evaluate more alternatives, and apply more principles
- It increases the confidence and satisfaction of problem solvers by making them feel more competent, autonomous, and motivated

Effects of Expertise in Problem-Solving

It facilitates the learning and development of problem solvers by helping them acquire new knowledge, skills, and strategies

It fosters the collaboration and communication of problem solvers by enabling them to share their perspectives, insights, and feedback

Expertise in Problem Solving

- MTH Chi, PJ Feltovich, R Glaser, Categorization and representation of physics problems by experts and novices, Cognitive science 5 (2), 121-152

https://web.eecs.umich.edu/~movaghar/Cognitive%20Science%20-%20April%201981%20-%20Chi%
%20Categorization%20and%20Representation%20of%20Physics%20Problems%20by%20Experts%20

- Summary?

I Am Devloper
@iamdevloper

manager: we need to design an admin system for a veterinary centre

dev: ok, this is it, remember your training

class Dog extends Animal {}

Expertise in Problem Solving

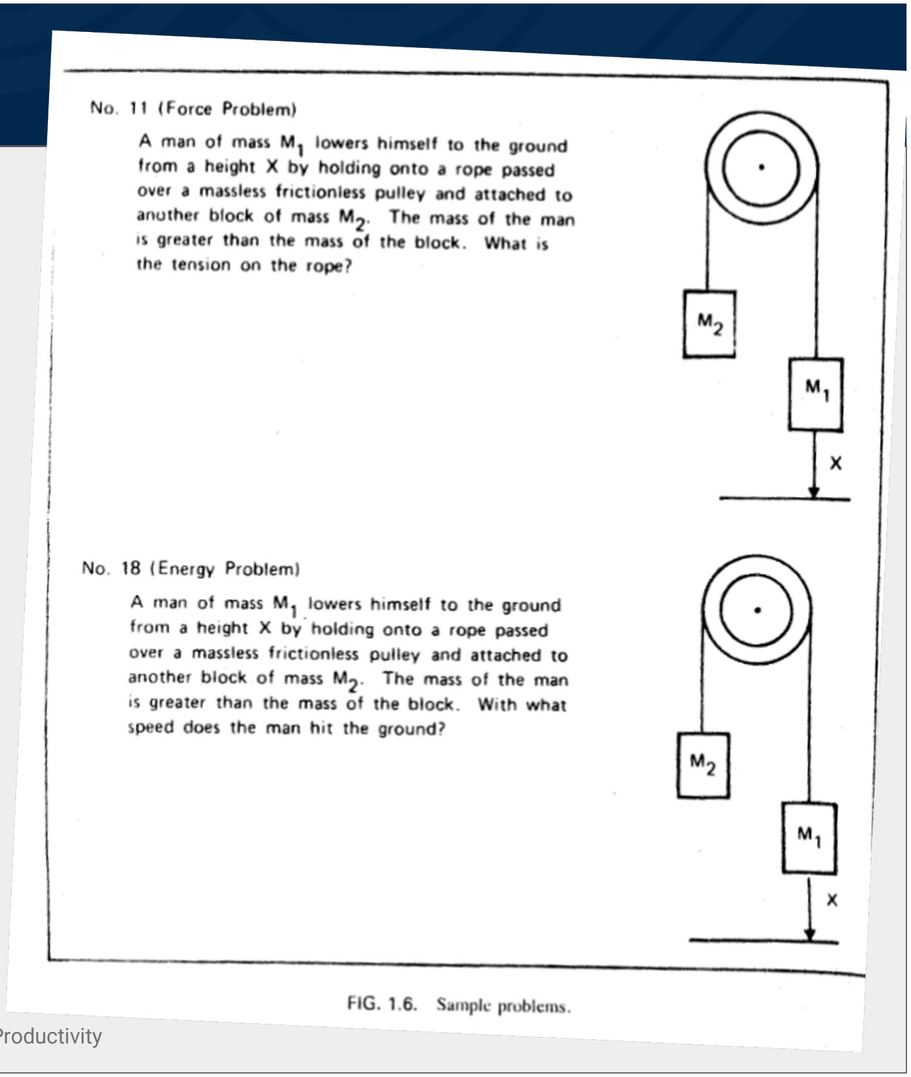
- “Both expert and novice proceed to solution by evoking the appropriate physics equations and then solving them. The expert often does this in one step, however …”
- “The speed with which a problem can be solved depends a great deal on the skill of the individual. Simon and Simon noted a 4:1 difference … Larkin also reported a similar difference between her experts and novices.”

Expertise in Problem Solving

- “Another interesting aspect of novice problem solving is not only that they commit more errors than experts but that, even when they do solve a physics problem correctly, their approach is quite different.”

Expertise in Problem Solving

- These two problems have a similar superficial structure



No. 11 (Force Problem)

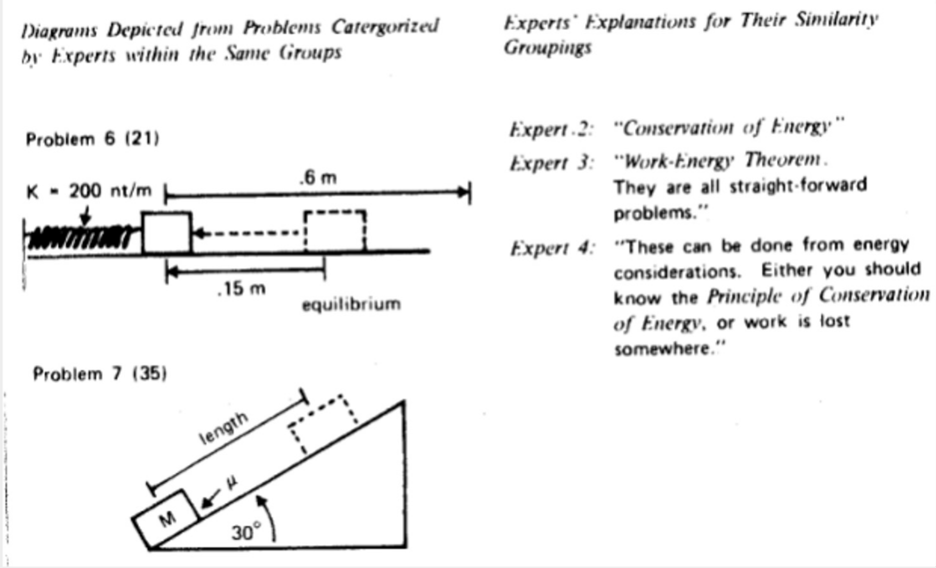
A man of mass M_1 lowers himself to the ground from a height X by holding onto a rope passed over a massless frictionless pulley and attached to another block of mass M_2. The mass of the man is greater than the mass of the block. What is the tension on the rope?

No. 18 (Energy Problem)

A man of mass M_1 lowers himself to the ground from a height X by holding onto a rope passed over a massless frictionless pulley and attached to another block of mass M_2. The mass of the man is greater than the mass of the block. With what speed does the man hit the ground?

FIG. 1.6. Sample problems.

Expertise in Problem Solving



Diagrams Depicted from Problems Catergorized by Experts within the Same Groups

Experts' Explanations for Their Similarity Groupings

Problem 6 (21)

K = 200 nt/m
.6 m
.15 m
equilibrium

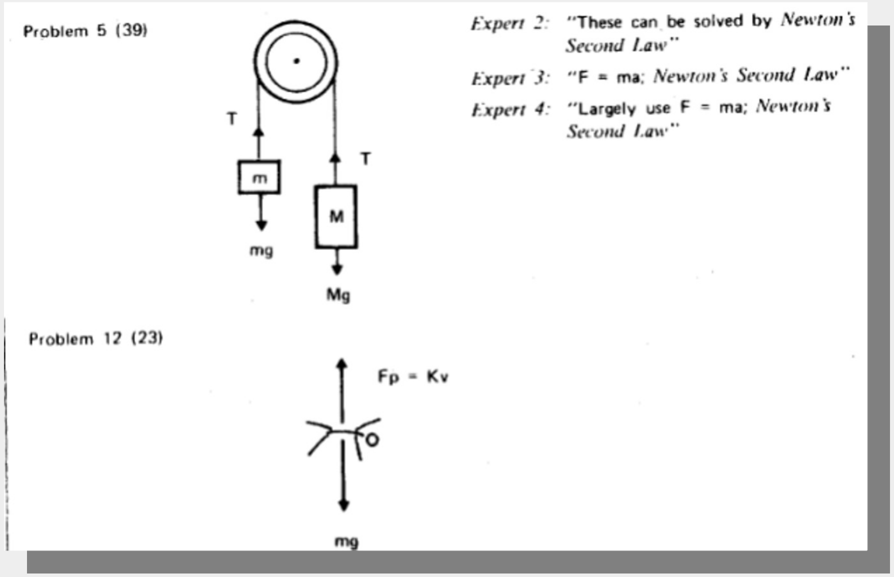
Expert 2: "*Conservation of Energy*"

Expert 3: "*Work-Energy Theorem*. They are all straight-forward problems."

Expert 4: "These can be done from energy considerations. Either you should know the *Principle of Conservation of Energy*, or work is lost somewhere."

Problem 7 (35)

length
μ
M
30°



Problem 5 (39)

T
m
mg
T
M
Mg

Expert 2: "These can be solved by *Newton's Second Law*"

Expert 3: "F = ma; *Newton's Second Law*"

Expert 4: "Largely use F = ma; *Newton's Second Law*"

Problem 12 (23)

Fp = Kv
mg

Expertise in Problem Solving

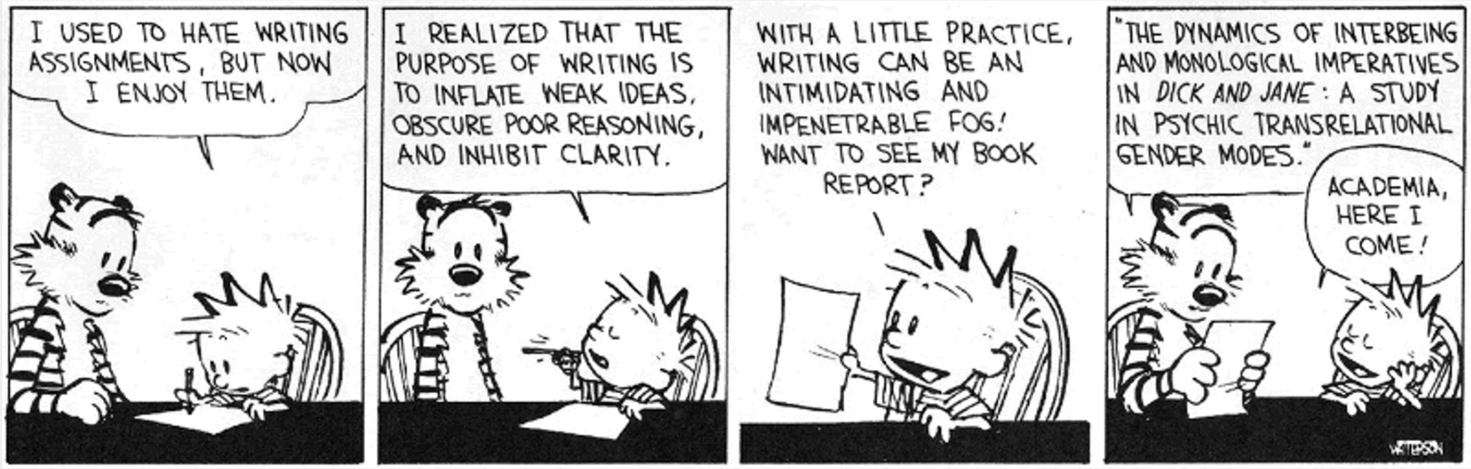
- "In this study, we specially designed a set of 20 problems to test the hypothesis that novices are more dependent on surface features, whereas experts focus more on the underlying principles. … We were able to replicate the initial findings that experts categorize problems by physical laws, whereas novices categorize problems by the literal components."

Expertise in Problem Solving

- “If we assume that such categories reflect knowledge schemata, then our results from the person at the intermediate skill level suggest that, with learning, there is a gradual shift in organization of knowledge --- from one centering on the physical components, to one where there is a combined reliance on the physical components and the physics laws, and finally, to one primarily unrelated to the physical components.”

Expertise in Problem Solving

- “Improved ability to learn would be developed through a knowledge strategy in which individuals would be taught ways in which their available knowledge can be recognized and manipulated.”
  - Do we do this in school?


I USED TO HATE WRITING ASSIGNMENTS, BUT NOW I ENJOY THEM.
I REALIZED THAT THE PURPOSE OF WRITING IS TO INFLATE WEAK IDEAS, OBSCURE POOR REASONING, AND INHIBIT CLARITY.
WITH A LITTLE PRACTICE, WRITING CAN BE AN INTIMIDATING AND IMPENETRABLE FOG! WANT TO SEE MY BOOK REPORT?
"THE DYNAMICS OF INTERBEING AND MONOLOGICAL IMPERATIVES IN DICK AND JANE: A STUDY IN PSYCHIC TRANSRELATIONAL GENDER MODES."
ACADEMIA, HERE I COME!
WATTERSON

Expert Bodies, Expert Minds

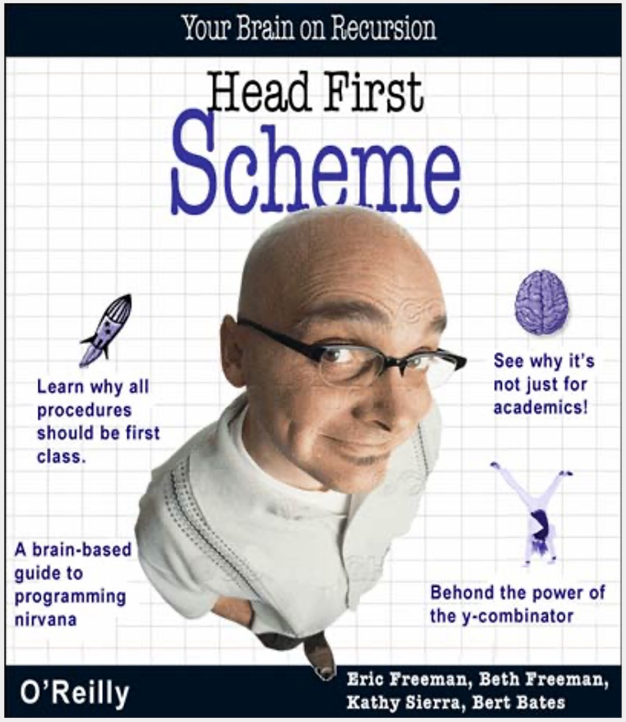
Spud Webb (5’7”) 1986 NBA Slam Dunk Contest

Expert Bodies, Expert Minds

- U. Debarnot, M. Sperduti, F. Di Rienzo, and A Guillot. *Experts bodies, experts minds: How physical and mental training shape the brain*. Frontiers in Human Neuroscience, 2014.

https://web.eecs.umich.edu/~movaghar/fnhum-08-00280.pdf

- Summary?

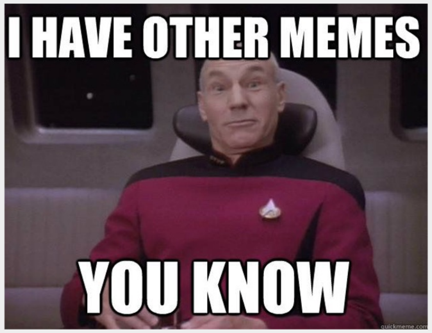

Your Brain on Recursion
Head First
Scheme
Learn why all procedures should be first class.
See why it's not just for academics!
A brain-based guide to programming nirvana
Behond the power of the y-combinator
O'Reilly
Eric Freeman, Beth Freeman, Kathy Sierra, Bert Bates

Expert Bodies, Expert Minds

- “These results suggest that the disparity between the quality of the performance of novice and expert golfers lies at the level of the functional organization of neural networks during motor planning. More generally, Patel et al. (2013) demonstrated that spatially distributed cortical networks and subcortical striatal regions may serve as neural markers of practice interventions.”
  - What's a “practice intervention”?

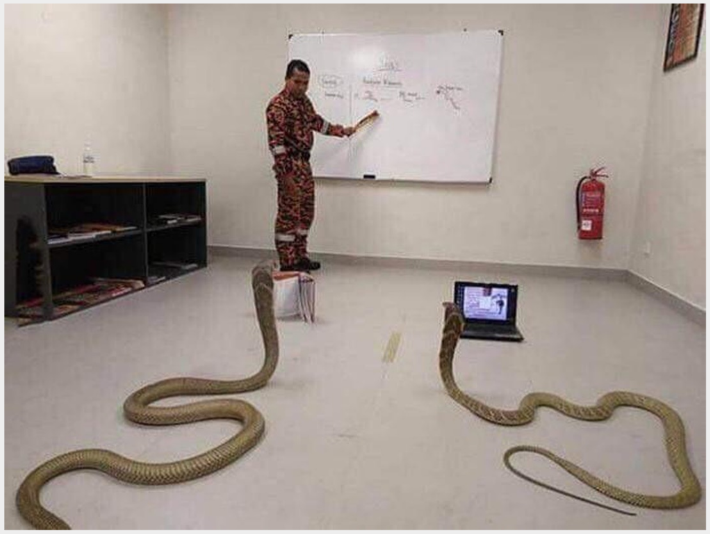
Expert Bodies, Expert Minds

- "Recently, Picard et al. (2013) examined the consequence of practice-dependent motor learning on the metabolic and neural activity in M1 of monkeys who had extensive training (~1–6 years) on sequential movement tasks. They found that practicing a skilled movement and the development of expertise lead to lower M1 metabolic activity, without a concomitant reduction in neuron activity. In other terms, they showed that less synaptic activity was required to generate a given amount of neuronal activity."
    - What does this mean?



Expert Bodies, Expert Minds

- Scholz et al. (2009) reported experience-induced changes in white matter architecture following a short period of practice. Practically, it was found that 6 weeks of juggling practice protracted an increased fractional anisotropy in a region of white matter underlying the intraparietal sulcus.



Taxi Cab Drivers

- If the brain anatomy parts are a bit opaque, it may be easier to interpret a famous study of London taxi cab driver brains [http://www.scientificamerican.com/article/london-taxi-memory/]. Memorizing and navigating that spatial problem (London is not laid out on a clean grid) causes growth in the hippocampus. Quote:
  - “These navigational demands stimulate brain development, concludes a study five years in the making. With the new research, scientists can definitively say that London taxi drivers not only have larger-than-average memory centers in their brains, but also that their intensive training is responsible for the growth.”

Back To The Time I Exercised

- What are other ways to solve this?
  - Hint: Many did not “write a program” at all in the conventional sense.
- If this were a contest (*and it is not*!), the key decision/mistake happened in the first seconds when you decided to write a program.
  - “C vs. Python” is a red herring: to phrase things as pejoratively as possible, that determines the winner of the loser's bracket.

What Predicts Software Developers' Productivity

- E. Murphy-Hill, C. Jaspan, C. Sadowski, D. C. Shepherd, M. Phillips, C. Winter, A. K. Dolan, E. K. Smith, M. A. Jorde. *What Predicts Software Developers' Productivity?* Transactions on Software Engineering, 2019.
- " … a survey that asked 622 developers across 3 companies [Google, ABB, National Instruments] about these productivity factors and self-rated productivity"

https://web.eecs.umich.edu/~movaghar/What_Predicts_Software_Developers_Productivity%20IEEE-TSE%202019.pdf

Research Paper

- The previous article is a research paper that investigates the factors that influence software developers' productivity. The authors conducted a large-scale survey of over 622 developers from 3 companies (Google, ABB, and National Instruments) and analyzed their responses using statistical methods.
- The paper concludes that software developers' productivity is a complex and multidimensional phenomenon that requires a holistic and empirical approach to measure and improve.
- The paper also provides some implications and recommendations for software engineering research and practice

Research Conclusions

- Software developers' productivity is not only related to the amount of code they write, but also to the quality, impact, and usefulness of their code
- Software developers' productivity is affected by various personal, social, and environmental factors, such as their motivation, satisfaction, collaboration, feedback, tools, and processes
- Software developers' productivity is not a static or fixed attribute, but a dynamic and context-dependent one, that can vary across different tasks, projects, and domains

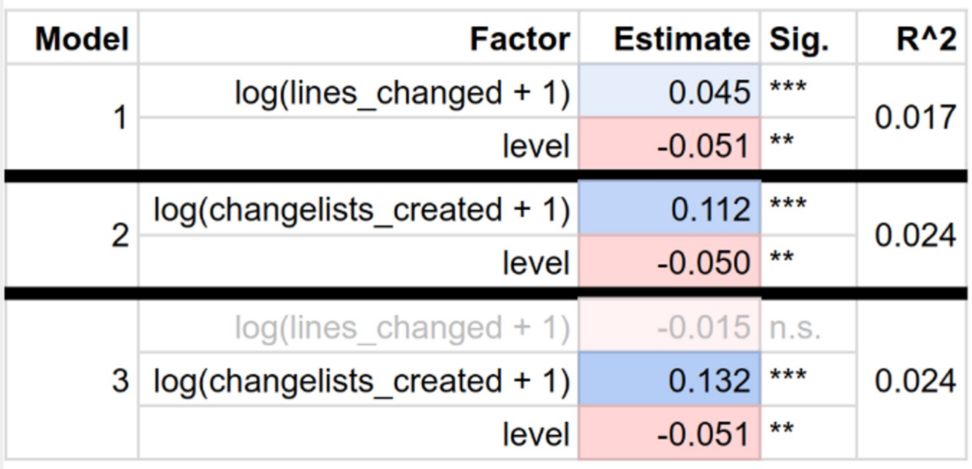
COCOMO Factors

- COCOMO factors are the parameters that affect the cost, effort, and schedule of software development projects, according to the COCOMO (Constructive Cost Model) developed by Barry W. Boehm
- There are two types of COCOMO factors: scaling drivers and effort multipliers

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

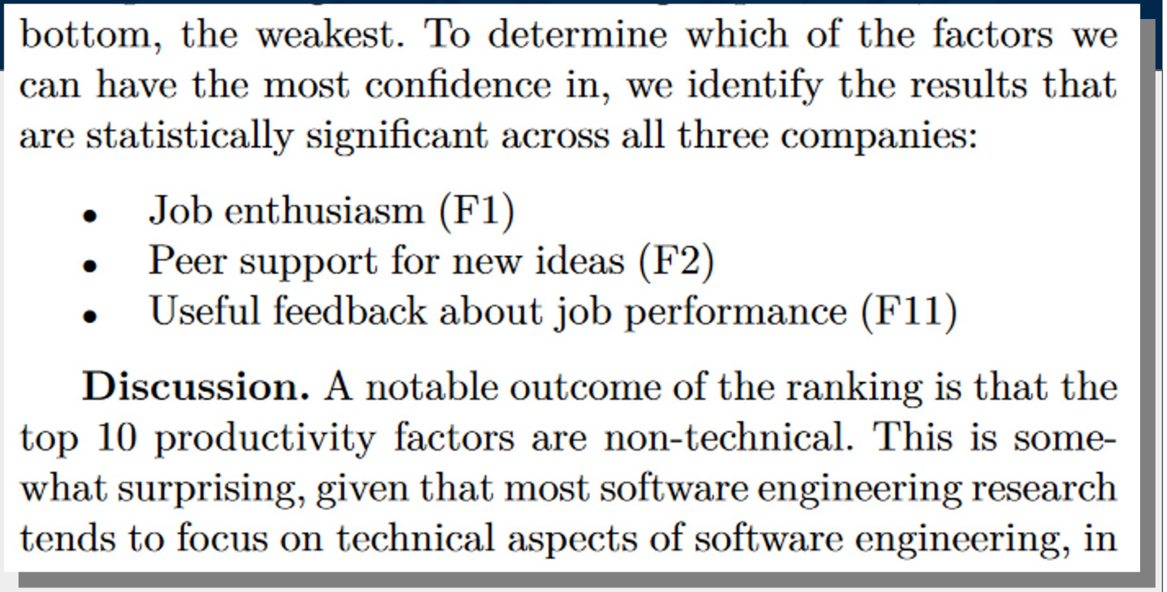
Self-Reported?

- “I regularly reach a high level of productivity.”
- Correlate with some objective measures at Google (n=3344)
  - Senior devs self-report less productivity



Model	Factor	Estimate	Sig.	R^2
1	log(lines_changed + 1)	0.045	***	0.017
	level	-0.051	**	
2	log(changelists_created + 1)	0.112	***	0.024
	level	-0.050	**	
3	log(lines_changed + 1)	-0.015	n.s.	0.024
	log(changelists_created + 1)	0.132	***	
	level	-0.051	**	

The Results



bottom, the weakest. To determine which of the factors we can have the most confidence in, we identify the results that are statistically significant across all three companies:

- Job enthusiasm (F1)
- Peer support for new ideas (F2)
- Useful feedback about job performance (F11)

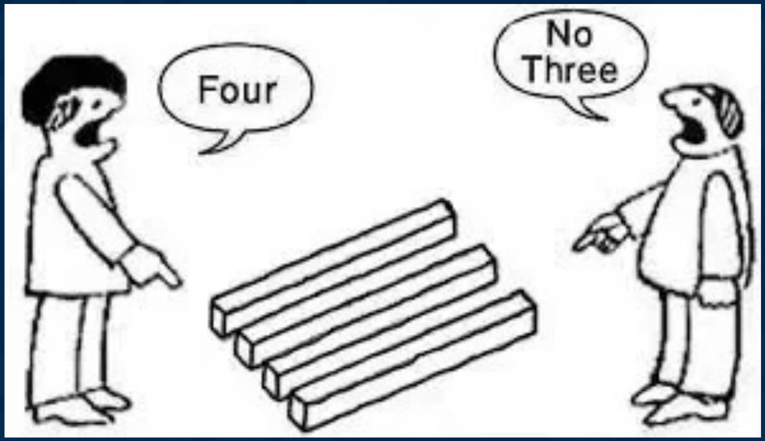
Discussion. A notable outcome of the ranking is that the top 10 productivity factors are non-technical. This is somewhat surprising, given that most software engineering research tends to focus on technical aspects of software engineering, in

- They also included COCOMO factors (what are those again?) and found that they didn't matter
  - Either COCOMO isn't accurate
  - Or it's accurate at the project, not the person, level

Hypothesis

- ~~My computer is slow.~~
- I'm slow ~~and so is my program.~~
- I picked the wrong ~~language~~/abstraction and couldn't break up the problem.
- I did not recognize the true components of the problem.
- My brain is currently inefficient, requiring much metabolism for little neural activation.

Everyone is entitled to their opinion

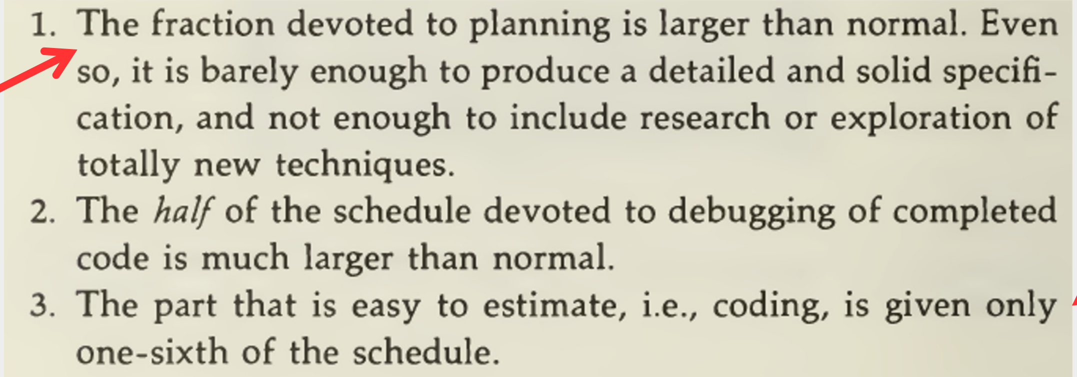


My Opinion: Programming Performance

- A substantial performance factor designated as "programming speed," associated with faster coding and debugging, less CPU time, and the use of a higher order language.
  - Programming Speed = Common Mistaken Belief!
  - Use of Abstraction = The Real Deal
    - The language is just one way to get abstraction. Abstraction (so that you can break up the problem and re-use existing solutions) is the relevant insight.

My Opinion: Mythical M-M

- “Planning” includes deciding whether write a standard program or whether to try something different (“totally new techniques”)
  - Coding is much less relevant than many think.



1. The fraction devoted to planning is larger than normal. Even so, it is barely enough to produce a detailed and solid specification, and not enough to include research or exploration of totally new techniques.
2. The *half* of the schedule devoted to debugging of completed code is much larger than normal.
3. The part that is easy to estimate, i.e., coding, is given only one-sixth of the schedule.

I Am Devloper
@iamdevloper

Always enjoy seeing someone trying to exit Vim for the first time.

Lady Gaga @ladygaga
AAAAAAAAAAAAAHHHHRHRGRGRGRRRGUR
BHJB
EORWPSOJWPJORGWOIRGWSGODEWPGOHE
PW09GJEDPOKSD!!!!!!!!!!!!!!!!
0924QU8T63095JRGHWPE09UJ0PWHRGW

12:37 PM · 18 Sep 18

157 Retweets 386 Likes

My Opinion: Mythical M-M

- “The real insight is the observation of **language invariance.**
  - You can get 10 lines of ASM or 10 lines of Python.”
- All keystrokes in my solution to this problem
  - [Ctrl]-A cat > foo [Enter] [Ctrl]-V [Ctrl]-D vim foo [Enter] Vjjjjjjjjjd :%s/$/+/g [Enter] :0VGJA0 [Enter] V!bc -l [Enter] A/10000 V!bc -l [Enter]
- You can solve this by typing less, not faster.
  - Would typing 100% faster or slower have mattered?

The notion of language invariance

- The observation of language invariance in Fred Brooks' "The Mythical Man-Month" refers to the idea that the productivity of software engineers does not significantly vary with the programming language used. This insight suggests that the time required to develop software is relatively constant regardless of the language, because the complexity and challenges of software engineering are primarily due to the inherent difficulties of the tasks themselves, rather than the tools used to accomplish them.
- Brooks argues that the main factors affecting software development time are the conceptual and communicative work involved, rather than the specific syntax or features of a programming language. This concept is part of a broader discussion in the book about the fallacy of measuring productivity in "man-months" and the complexities of software project management.

Criticisms about The Mythical Man-Month

Outdated Practices: Some argue that the book's insights, while revolutionary at the time, are less applicable in today's agile and fast-paced software development environment.

Overemphasis on Large Systems: Brooks' experiences were primarily with large, complex systems at IBM. Critics say this may translate poorly to smaller projects or different types of software development.

Brooks' Law Misinterpretation: The idea that "adding manpower to a late software project makes it later" can be misinterpreted as an argument against scaling teams, whereas Brooks intended it as a caution against poor planning and coordination.

Criticisms about The Mythical Man-Month (Cont'd)

Neglect of Modern Tools: The book predates many modern tools and practices that can mitigate some of the issues Brooks described, such as version control systems, continuous integration, and comprehensive automated testing.

Underestimation of Human Factors: While Brooks acknowledges the human element in software engineering, some feel he underestimates the impact of team dynamics, motivation, and individual skill levels.

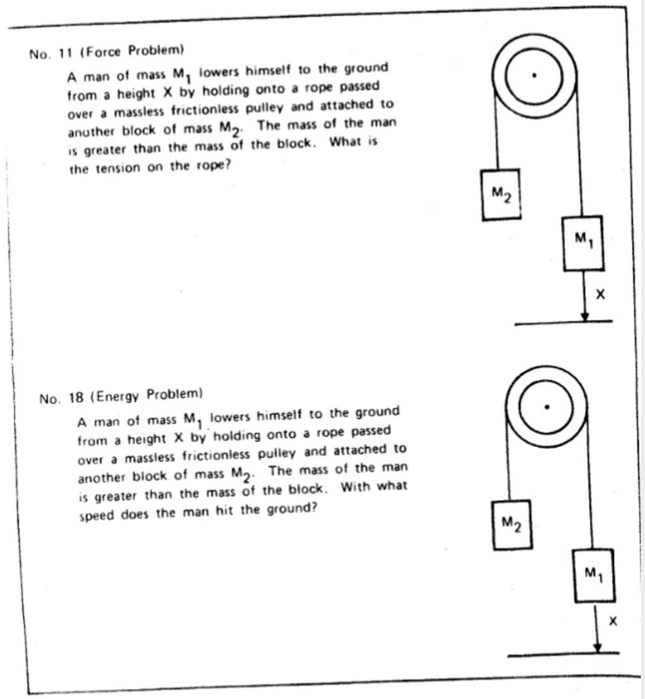
No Silver Bullet: Brooks famously argued that no single technology or practice would produce a tenfold improvement in productivity within a decade. Critics have pointed to the rise of high-level programming languages, development frameworks, and methodologies that have significantly boosted productivity.

My Opinion: Expertise in Problem Solving

- “Another interesting aspect of novice problem solving is not only that they commit more errors than experts but that, even when they do solve a physics problem correctly, their approach is quite different.”
- Story time: “I've seen this one before.”
  - Linux OOM Killer.
- “approach is quite different” cf. “new techniques”
  - Is “calculate math” a primitive in your language?

My Opinion: Problem Solving

- Many of you looked at the problem and, despite the instructions, saw that it looked similar to programming tasks you'd been given before.
- Those are “it looks like a pulley” surface features (file access then loop to compute total then divide).
- You wanted “it uses Newton's 2nd Law” deep features (compute the average).



No. 11 (Force Problem)

A man of mass M_1 lowers himself to the ground from a height X by holding onto a rope passed over a massless frictionless pulley and attached to another block of mass M_2. The mass of the man is greater than the mass of the block. What is the tension on the rope?

No. 18 (Energy Problem)

A man of mass M_1 lowers himself to the ground from a height X by holding onto a rope passed over a massless frictionless pulley and attached to another block of mass M_2. The mass of the man is greater than the mass of the block. With what speed does the man hit the ground?

FIG. 1.6. Sample problems.

My Opinion: Expert Bodies, Expert Minds

- On Page 6 (= Page 17) the Chi reading talks about three quantifiable (!) differences between experts and novices when solving problems.
  - The first is raw solution time (which we already saw in the Sackman reading).
  - The second is pauses in retrieving chunks of the correct equation. This is more interesting (cf. "chunking"): "experts group their equations in chunks so that the eliciting of one equation perhaps activates another related equation, and thus it can be retrieved faster". For programming, replace "equation" with "program fragment".
- One difference that previous students noted after watching my "how I did it" explanation was that I never really seemed to stop and think about what to do next, whereas a student might write the code to read in lines, stop and think, write the code to iterate over them and sum them, stop and think, write the print-and-divide code, etc. If you've observed that in yourself, the psych research summarized in the Chi reading suggests that one area for improvement is to get better at chaining from one fragment to the next.

Difference between Experts and novices

There are three main quantifiable differences between experts and novices when solving problems:

- The first difference is that experts can recognize and categorize problems more quickly and accurately than novices, based on their prior knowledge and experience
- The second difference is that experts can represent and organize problems more effectively and efficiently than novices, using their superior memory and mental models
- The third difference is that experts can search and select solutions more rapidly and reliably than novices, using their refined strategies and heuristics

My Opinion

- My “plan” breakdown:
  - This problem is regular expressions plus a calculator.
    - Use regular expressions to turn the input into an arithmetic expression (“into a program”)
    - Feed that to a pre-existing calculator
- Students who said “I will pass this to Excel” also did well.
  - Why are you re-inventing the wheel? Your boss wanted the right answer as fast as possible.

Story Time (They’re Fables)

I am the only one who remembers this fairy tale?

no one:
absolutley no one:
the grimm brothers:

Abstraction

- Abstraction is the process of generalizing concrete details, such as attributes, away from the study of objects and systems to focus attention on details of greater importance
- Abstraction is a fundamental concept in computer science and software engineering, especially within the object-oriented programming paradigm
- Abstraction can be achieved by using various features and techniques, such as abstract data types, subroutines, modules, polymorphism, inheritance, design patterns, architectural styles, and software components

Benefits of Abstraction

- It reduces code duplication and complexity by reusing common functionality and hiding irrelevant details
- It improves code quality and maintainability by following coding standards and best practices
- It enhances code performance and efficiency by optimizing the use of resources, such as CPU, memory, disk, network, and power.
- It facilitates code collaboration and communication by using clear and consistent interfaces and protocols

Benefits of Abstraction

- It enables code scalability and reliability by supporting concurrency and fault-tolerance
- It fosters code creativity and innovation by allowing the creation of new abstractions and languages

Story Time: Abstraction

- One of the classical elements of magical fantasy is the ability to transform one object or creature into another. This spans cultures, from the Greek myth of Circe turning sailors to beasts to the magical transformation duel in Disney's The Sword in the Stone (http://video.disney.com/watch/wizards-duel-4be36b86f6d55e5bc7f6b2d6). Indeed, many fantasy games feature this notion under the "formal" name of polymorph. One of my favorite roleplaying systems codifies this nicely: http://www.d20srd.org/srd/spells/polymorph_AnyObject.htm. To the suitably prepared and devious mind, a polymorph spell is much more deadly than the usual combat fireball or lightning bolt. You will make a much bigger explosion by polymorphing your foe's 40 pound suit of armor into 40 pounds of nitroclygerin than you will with any standard fireball. Indeed, many such systems must implicitly or explicitly disallow such "chemistry" lest it break the balance and challenge of the game.

Story Time: Abstraction

- You could take a moment to actually read that spell description linked above. In one sense, an innocuous line is actually the most interesting:
- Target: One creature, or one nonmagical object of up to 100 cu. ft./level
- The spell can transform a single object. One object, eh? What exactly is a single object? It turns out that this is a difficult -- and effectively unsolved -- question. If you haven't run into it in your philosophy courses, check out http://en.wikipedia.org/wiki/Ship_of_Theseus. For example, in Norse Mythology there is a magical ship that can be transformed into folded up cloth (http://en.wikipedia.org/wiki/Sk%C3%AD%C3%B0bla%C3%B0nir). So it seems that "one ship" is sometimes "one object". But could just the mast or the sail of the ship also be one object?

Advice 1/3: Small Potatoes

- Try to learn a shell-based editor, such as vim or emacs, and practice suspending the editor (ctrl-z, fg) rather than restarting it. If you must use something like Eclipse for a project, start it once and never quit it.
- Inasmuch as extra hand actions on your part are isomorphic to the computer delaying before giving you what you really want, master "focus follows mouse" (yes, even Windows supports it) and editors that don't involve new windows. Similarly, master keyboard shortcuts and favor an editor that allows you to make your own macros. Memorize the common ones shared across many interfaces, like ctrl-a (beginning of line) and ctrl-e (end of line -- those both work in the shell as well).
- Buy fast storage.

Advice 2/3

- Students often overemphasize the effect of low-level notions like typing speed but underemphasize high-level decisions (like breaking down a problem so its components can be solved in terms of transformations on existing solutions). When adding numbers, we demonstrated this concretely by taking what was to some a unitary atomic problem ("sum a list of numbers") into smaller parts ("turn a list of numbers into an arithmetic expression with regular expressions" and "invoke a calculator").
- This is non-obvious for a few reasons, not the least of which is that the parts actually appear to be larger, not smaller! So one trick is to gain enough felicity with various small problems in computer science that you can solve them quickly (see Sackman reading), as well as to retrieve them quickly and do the chunking to break down the big problem in terms of those parts (see Chi reading) without your machine setup actually getting in the way (see Dougherty reading).

Advice 3/3

- Ultimately, the bottleneck productivity limitation is not your typing speed. The real obstacle is more a conceptual limitation related to abstraction -- and there may be no shortcut to years of practice, the sort of study that ultimately changes the organization of your brain.
- Good luck.

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Questions?

- HW 5 is due today!
- HW 6a is due next Wednesday.

My cousin just got a job programming AI software.

I'm jealous of his ability to make friends at work.

@TheChrisSchmidt