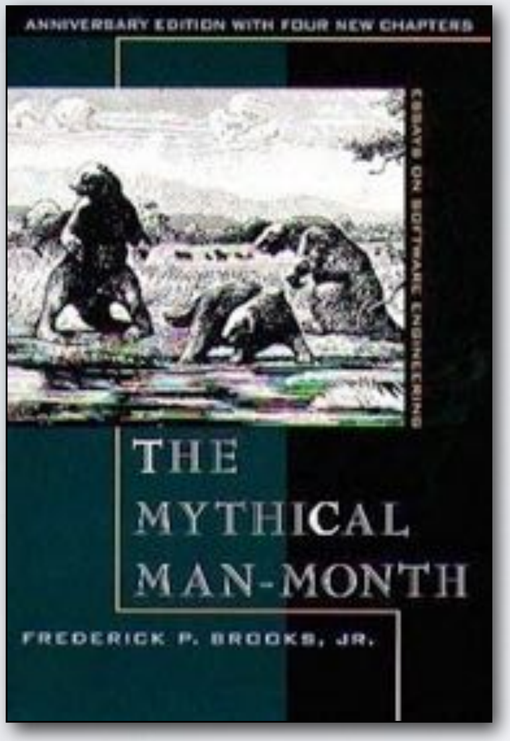
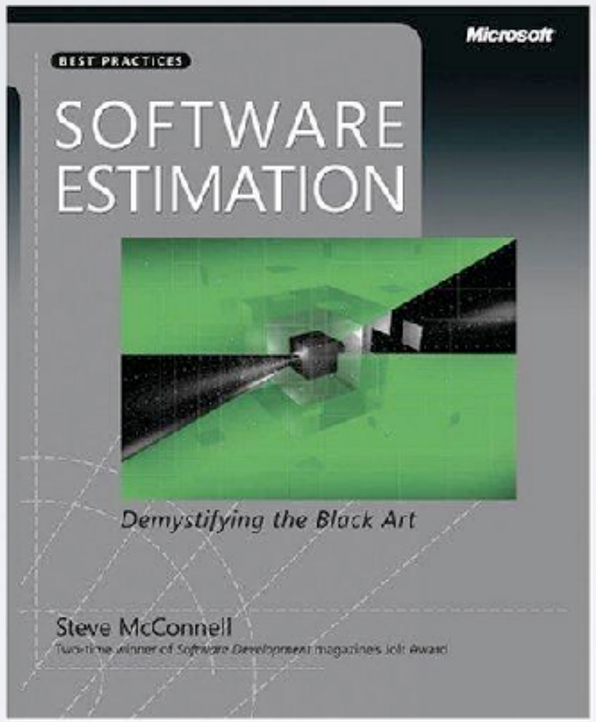
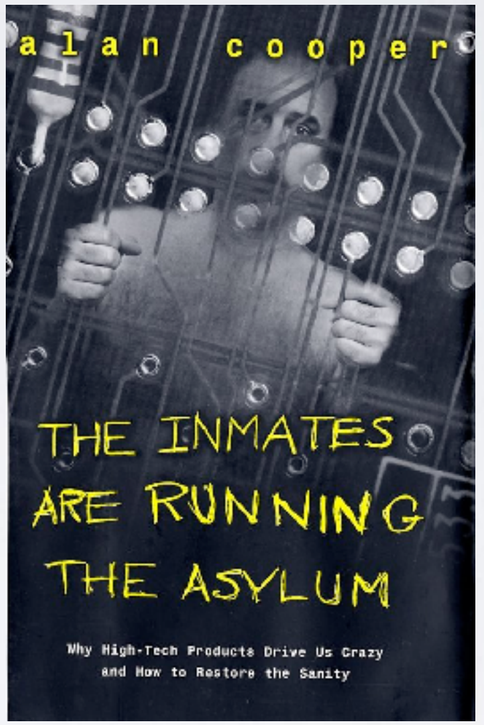
Estimating and Expectations

Paul M. Jones
paul-m-jones.com

Read These


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.


Microsoft
BEST PRACTICES
SOFTWARE ESTIMATION
Demystifying the Black Art
Steve McConnell


alan cooper
THE INMATES ARE RUNNING THE ASYLUM
Why High-Tech Products Drive Us Crazy and How to Restore the Sanity


Peopleware
Productive Projects and Teams
Second Edition Featuring Eight All-New Chapters
Tom DeMarco
&
Timothy Lister

About Me

- Developer, Senior Dev, Team Lead, Architect, VP Engineering
- Military, education, non-profit, startups, products, consulting
- Local and remote, large projects and small ones
- SpikeTV, MTV, Facebook, Microsoft, Apple, etc.

Part 1:
Everybody's Story

A New Project

- Client/internal customer approaches you
- Has a textual narrative requirements document
- Possibly has an external reference system: “just like Facebook”

Budget and Deadline

- Typically has a certain budget determined in advance
- Typically has a particular deadline in mind. “Or else ...”
  - “... we lose a deal.”
  - “... we miss a trade show.”
  - “... a competitor beats us.”

Due Diligence

- Sit down and talk with client
- Elicit functionality
- Elicit business processes

Functionality, Milestones, Contract

- Talks result in a functionality list
- Perhaps also a milestone list
- You convince yourself that the budget and deadline are fair
- You convince the client and make the sale

Rationalization

- Maybe the deadline is a *little* tight
- Maybe the functionality is a *bit* too extensive
- “Work late nights or weekends as needed to make up.”
- “We’re committed to the project! We can make it happen!”
- “The only limit is ourselves!”

The Project Begins ...

- 3-month project, first 2 weeks go OK
- Environment setup
- System basics
- Early functionality

...To Take Longer Than It "Should"

- Random/personal events
- Technical difficulties
- Client-related issues

Random/Personal Events

- Sickness (self, spouse, kids)
- Emergency leave
- Car trouble
- Weather

Technical Difficulties

- Recompile extensions
- Pick a new library/framework/etc
- External service troubles/failure
- Environment craps out (e.g., HD failure)
- Dropped laptop, spilled coffee, etc.

Client-Related Issues

- Detail disconnects
- Modified functionality
- Added functionality ("but it's implied!")
- Redone work
- More meetings
- System is *much* more complex than

Heroic Measures

- Work weekends like normal days
- Work longer days
- Delegate by expertise
- Add developers/outsource

Difficult Conversations

- Explain to client why schedule is slipping
- Maybe reduce minor functionality
- Still need it after the deadline, though

Late. Late. Late.

- 3 month project is taking 4 months or longer.
- Fixed price? Working at reduced margins.
- Best case: get it all done, just late.
- Common case: finish with reduced scope, quality, and morale.
- Worst case: fired.

Another New Project

- “This time we’ll do better!”
- “We learned so much on the last project!”
- ... but different things go wrong.

There Is A Better Way

Part 2:
Laws

Laws

- Jones' Law
- Hofstadter's Law
- Brooks' Law

Jones' Law

- "If you plan for the worst, then all surprises are good surprises."
- Developers are insufficiently pessimistic.
- They are surprised when things go wrong.
- Cure: Expect for things to go wrong, and plan for it.
- Eventually you can calibrate more accurately what the "worst" will be.

Hofstadter’s Law

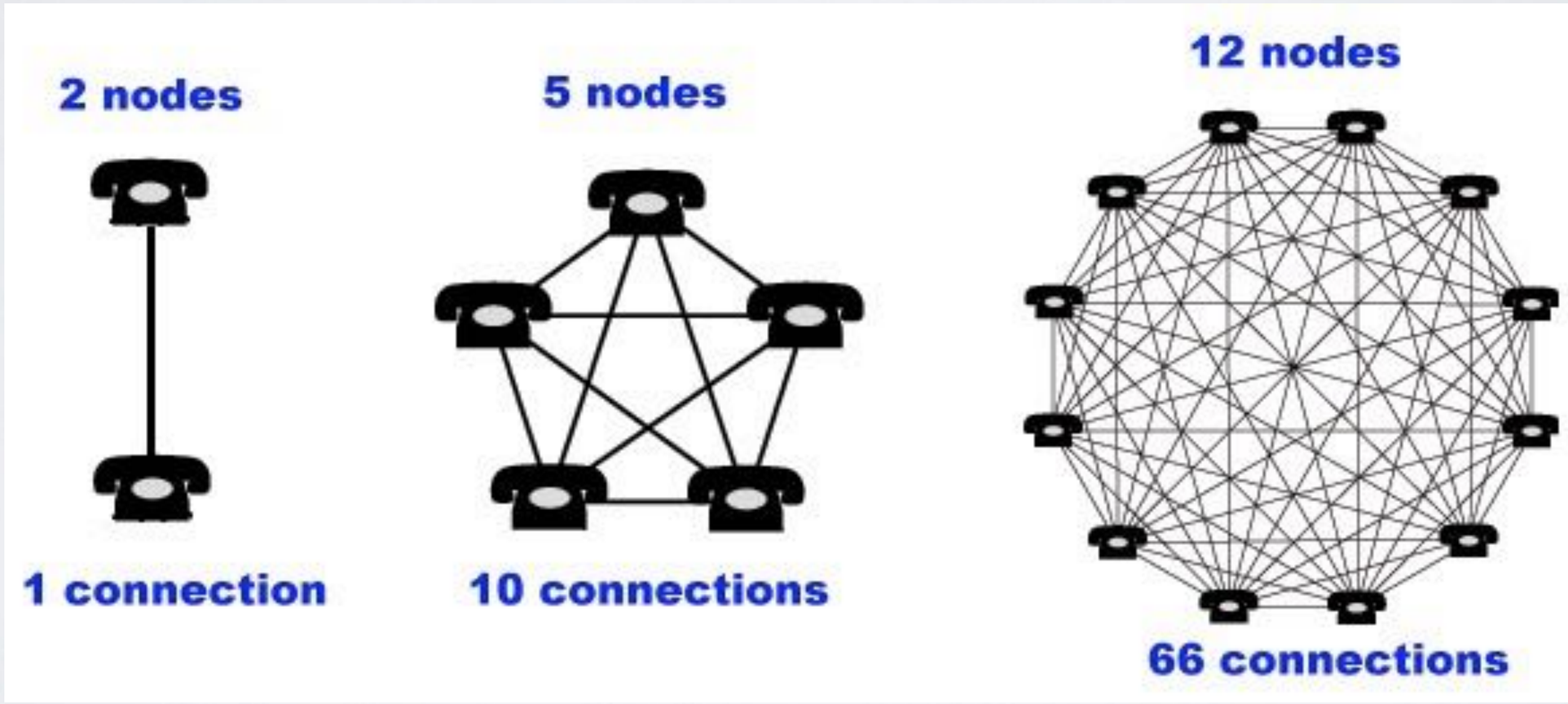
- “It always takes longer than you expect, even when you take into account Hofstadter's Law.”
- There will always be unexpected occurrences.
- “Then what good is it to make plans at all?”
- “Plans are nothing, but planning is everything.”

Brooks' Law

- "Adding manpower to a late software project makes it later."
- Software development is a knowledge profession. Knowledge has to be communicated and shared.
- Organizational differences: going from 2 developers to 4 or 6 means different management approaches
- Training time means at least one productive developer is lost while training and helping new developers

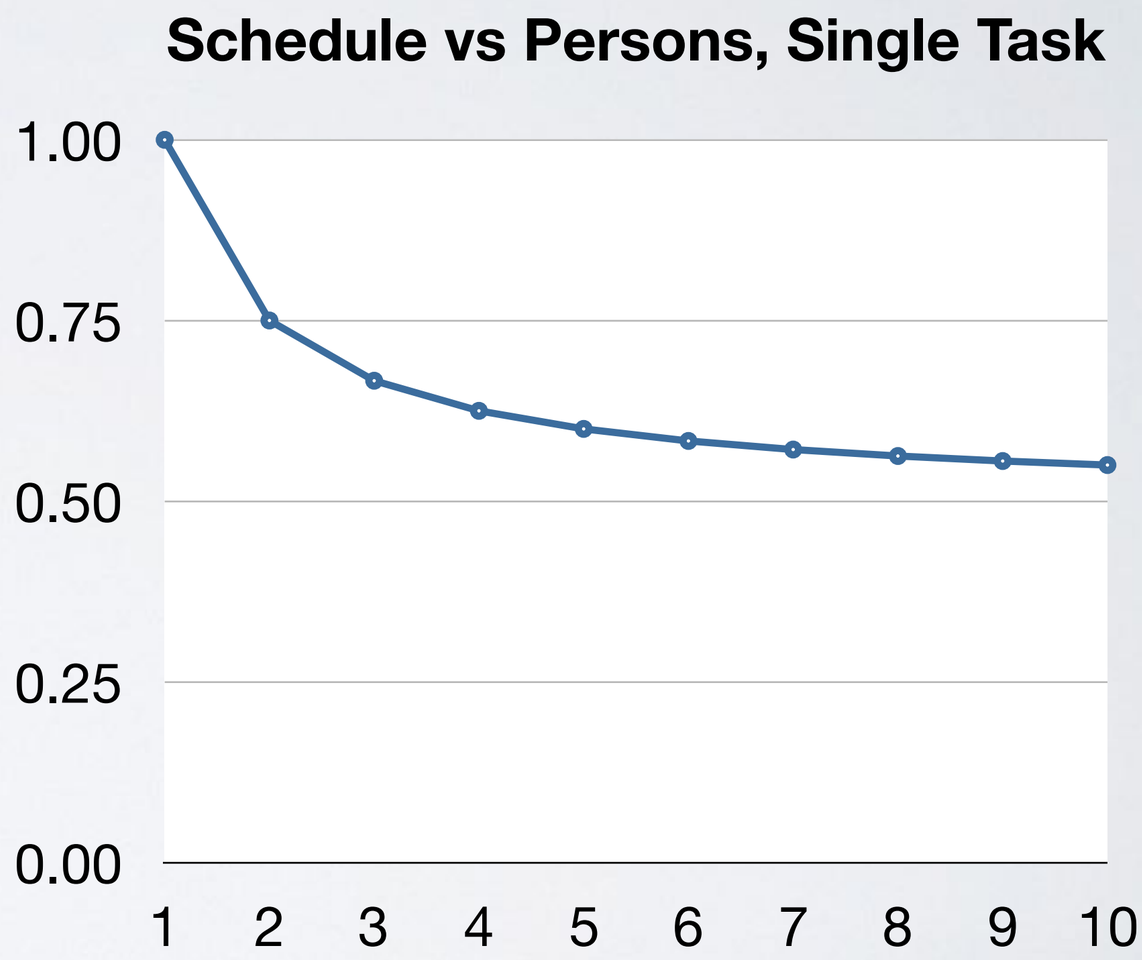
Effort vs Communication

- Available effort increases linearly,
  but communication increases exponentially.



Partitioning is Key

- For a task that cannot be partitioned, no matter how many people you add, you will never cut development time in half.
- The key is to be able to partition into independent tasks that can function separately and be integrated



Worker Pairs

- The biggest gain is to add one programmer (from 100% of remaining schedule to 75% of remaining schedule, given perfect knowledge)
- But don't add at the end; add at the beginning, once you have a good estimate.

Part 3:
Planning and Estimating

Estimates, Targets, Commitments

- Estimate: a prediction about cost or schedule
- Target: a desirable business objective
- Commitment: a promise to deliver by a certain date
- A “plan to commit to a target” is *not* the same thing as an “estimate”
- An estimate can tell you if a target commitment is realistic

You Are Terrible At Estimating

- Give a low value and a high value so that the true answer is 90% certain to fall in that range.
- "What is the surface temperature of the sun?"
- "What year was Alexander the Great born?"
- "What was the weight of the heaviest blue whale ever recorded?"

You Are Terrible At Estimating

- Sun: 10,000 deg F (6,000 deg C)
- Alexander: 356 BC
- Whale: 386,000 lbs/190 English tons (170,000 kg/170 metric tons)
- Even on questions of fact, estimating is difficult. How much more difficult is it for software development?

Estimating is Hard

- We want to "look smart" and that gets in the way.
- We are unwilling to say "I don't know" because we want to look smart.
- We are over-optimistic because we believe we are smart.
- Too easy to let political/managerial/social considerations get in the way.

Knowledge Is Paramount

- It's not enough to be smart; you have to actually know things.
- It's not enough to be a good programmer; you have to know what the client's business needs are.
- It's not enough to have good judgment; it's much better to be able to count.
- More knowledge about the project means a better estimate.

Getting Project Knowledge

- One way to gain knowledge about a project is to do the same one over and over ... but that never happens
- Another way is to do similar projects over and over, and that happens often in some kinds of work.
- Most commonly the project is mostly-new (to you). The best way to get knowledge is...

Design First

- Textual narratives are not enough.
- Design the project screens/pages first, before building an estimate.
- These need not be pixel-perfect. Wireframes/mocks are sufficient.
- Every form, every link, every interaction.
- From this will rise everything else you need to know about the project.

Objections

- “This is waterfall! This is big-bang design up front! Waterfall is bad!”
- “That's going to take too much time!”
- “We have to show the client something useful quickly!”
- “That's really expensive!”

Estimation Technique Guidelines

- Broad/coarse techniques are just as good as fine-grained ones
- Techniques based on historical data are better than judgment
- Shared/compared estimates are better than individual ones
- Spend your time gaining knowledge, not on complex estimation games

Two Workers, One Day, Per Controller Method

- Count up how many pages/controller in the design
- Estimate one day per controller method
- Estimate two workers to be involved in completing it
- Covers backend (DB/PHP/HTML) and frontend (HTML/JS/CSS)
- Graphic design is *not* generally covered; that's a parallel task
- http://paul-m-jones.com/archives/1837

Objections

- "That means a CRUD controller takes 2 workers 4 days."
- Look back at your own projects and count up the number of controller methods, then divide by how long it actually took and how many people were involved.
- If not 2 workers one day per controller, then close or worse.
- Warning: "This is for posterity, so, be honest."
- Warning: Does not cover setup, deployment, maintenance, etc.

Part 4:
Managing Expectations

Whose Expectations?

- Your own expectations for yourself
- The client's expectations for the project (McConnell, ch 23)

Guidelines

- Communicate early and often
- Estimate in the largest units possible
- Beware of ranges
- Estimate against coherent feature sets
- Be ready to re-estimate
- Maintain consistent pacing

Recovery Methods

- Cone of uncertainty
- Reduce features or extend deadline
- Schedule compression limits

Conclusion

- Everybody's Story
- General Laws
- Planning and Estimating
- Managing Expectations
- http://paul-m-jones.com/