# CS 428 Week #5 Readings

WINTER 2020, WEEK #5

BRUCE F. WEBSTER

# The Mythical Man-Month, Ch. 2: The Mythical Man-Month

- Root causes of software project delays and failure
  - Our techniques of estimation are [still] poorly developed
  - Our estimation techniques confuse effort with progress (people & months are interchangeable)
  - Because we are uncertain of our estimates, we often lack the courage to say we don't know when we'll be done
  - Schedule progress is poorly monitored and hard to measure
  - When the schedule slips, the impulse is to add staff, which is "like dousing a fire with gasoline"
- What have you observed?

# MMM Chapter 2 (cont.)

- "All programmers are optimists"
  - Only optimists build complex systems. (Adele Goldberg)
  - We too often assume each task will take only as long as it "ought" to take
  - The probability that a given task will go well may be relatively high, but a meaningful software project comprises hundreds if not thousands of such tasks
  - Thus: It is very easy to lose a day; it is impossible to make it up.
  - Additional complication: we tend to focus on the easy tasks first and defer the difficult problems until late in the project – illusion of great progress
- What are some other ways in which we tend to be overly optimistic?

# MMM Chapter 2 (cont.)

- The Man-Month
  - The “man-month” as a unit for measuring the size of a software engineering project is a dangerous and deceptive myth
  - Sequential constraints in development as well as communication requirements make the “man-month” concept unrealistic (and self-deluding)
  - Adding a person to a project not only increases the communication paths and requirements, it also costs time for bringing the new person up to speed
  - Thus, adding more people lengthens, not shortens, the schedule (Brooks Law)
- What do you think the impact of personnel turnover is?

# MMM Chapter 2 (cont.)

- Component debugging and system testing forces sequential constraints
  - Testing is usually the most mis-scheduled (underestimated) part of programming
  - Brooks' rule of thumb: 1/3rd planning, 1/6th coding, 1/4th component test, 1/4th system test
  - "I found that few allowed one-half the project schedule for testing, but that most did indeed spend half of the actual schedule for that purpose."
  - The 90/90 rule: 90% of the work takes the first 90% of the schedule, and the remaining 10% of the work takes the other 90% of the schedule
  - Underestimation of system testing (integration, end-to-end, performance, stress) is particularly damaging since it shows up right when project completion is expected

# MMM Chapter 2 (cont.)

- Gutless estimating
  - Endemic in our industry
  - Completion date is picked because “we have to have it by then” or to meet a “market opportunity”, not based on any rational basis or realistic estimate
  - Upper management often does not want to hear a realistic estimate
- Regenerative schedule disaster
  - So, what happens when the project is late? “Add people to it. Work longer hours.” Both are counter-productive.
  - Only real solution: slip deadline and/or drop features.
- Observations?

# The Mythical Man- Month, Ch. 7: Why Did the Tower of Babel Fail?

- What they did have:
  - A clear mission
  - Manpower
  - Materials
  - Time
  - Technology
- What they lacked?
  - Communication
  - And, as a consequence, organization
- Your observations/experience?

# MMM Chapter 7 (cont.)

- Project workbook: replaced today by online organization (e.g., your project wiki/repo)
- Communication challenge: with n workers on a project, there are $(n^2-n)/2$ possible interfaces and $2^n$ possible sets of workers
- Solution: Division of labor / specialization of function
- Key: project manager and chief architect need to be different people
  - Their priorities conflict
  - Chief architect will tend to be overly optimistic

# The Mythical Man-Month, Ch.14: Hatching a Catastrophe

- "How does a project get to be a year late? One day at a time."
- Milestones must be concrete, specific, measurable events
  - The myth of the "Oh, we're about XX% done" statement
  - 90/90 rule: 90% of the project takes the first 90% of the schedule; the remaining 10% of the project takes the other 90% of the schedule.
- The "three weeks before deadline" rule:
  - "*Underestimates* [of project schedule] do not change significantly during the activity until about three weeks before the scheduled completion."
- Need for a critical-path schedule (e.g., PERT) to show the critical path
- Observations?

# MMM Chapter 14 (cont.)

- Not being willing to pass bad news uphill
  - Webster: The Thermocline of Truth (2008)
- Not knowing the news is bad
  - Webster: Lies, Damned Lines, and Metrics (parts I through III) (2008)
  - Project progress metrics need to be objective, repeatable, and informative
  - Weinberg's Law of Metrics: That which gets measured gets fudged.
  - The Metric Law of Least Resistance: "The more human effort required to calculate a metric, the less often (and less accurately) it will be calculated, until it is abandoned or ignored altogether."
- Thoughts and observations?

# Peopleware, Ch 5: Parkinson's Law Revisited

- "Work expands to fill the time allotted." – was actually a satirical observation
- "Parkinson's Law almost certainly doesn't apply to your people."
  - They have too many other things they want to do.
- Bad estimates tend to lower productivity; good/credible estimates tend to raise it
  - Death march vs. achievable goal
- Organizational busy work tends to expand to fill the working day
- Observations and feedback?

- Same problem as project estimation, but for a project already underway
  - Most organizations are very bad at predicting when a given project will ship
  - Usually rely on ‘metrics’ that aren’t at all useful
- A meaningful, useful project metric has three key qualities:
  - Informative/predictive: tells you something important and/or when you will deliver
  - Objective: should yield the same value regardless of who is doing the measuring
  - Automated: can be done quickly and without direct human intervention
- Almost all major metrics used in IT projects lack one, two, or all three qualities

# Lies, damned lies, and project metrics [Part I, Part II, Part III] (Baseline, 2008)

# WEB #4 (cont.): Metric "Laws"

- **Weinberg's Law of Metrics:** "That which gets measured, gets fudged."
  - We will distort work and reporting to achieve required or valued metrics
- **The Metric Law of 90s:** "The first 90 percent of a development project takes 90 percent of the schedule. The remaining 10 percent of the project takes the other 90 percent of the schedule."
  - We tend to focus on low-hanging fruit in order to make metrics look good
- **The Metric Law of Least Resistance**: "The more human effort required to calculate a metric, the less often (and less accurately) it will be calculated, until it is abandoned or ignored altogether."
  - Hence the need for automation (cf. classic joke about drunk looking for keys)
- Must-read book: ***Measuring and Managing Performance in Organizations*** by Robert D. Austin (Dorset House, 1996)

# Web #4 (cont.): Visual Assessment in Real Life


- Why is project completion so hard to predict?
  - The amount of analysis (gathering relevant subject-matter information) that still has to occur
  - The amount of invention (novel problem solving) that still has to occur (cf Armour, as usual)
  - The amount of discovery (e.g., running into roadblocks and dead ends) that still has to occur (again, Armour)
  - The adequacy of the current architecture, design and implementation
  - The amount of actual coding that still has to occur
  - The amount of quality engineering (testing, reviews, etc.) that still has to occur
  - Unexpected turnover among engineering personnel
  - Changes in market requirements and/or opportunities
  - Changes in external systems upon which you depend

# WEB #4 (cont.): The challenge

# WEB #4 (cont.): Potential approach to useful metrics

- First, instrumentation: automated collection of wide range of metrics/characteristics over time
  - Result: time-stamped history for each metric/characteristic
  - These should be automated and objective
  - Can be tied to configuration management system and run on a regular basis
- Second, heuristics: use data collected
  - After project is done and with known timeline, use Bayesian analysis to see which combination of metrics best anticipate milestone completion
  - Use human analysis as well to look for correlations between metrics and actual progress (or lack thereof)
  - Refine set of metrics/characteristics for next project and see how well they predict progress
- **NOTE**: Check out gitprime.com for a system that appears to do just this