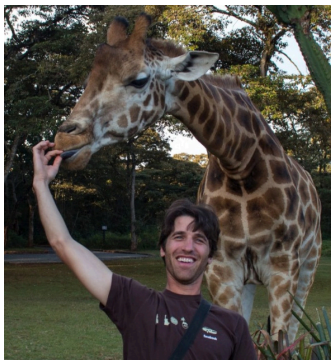
CSE 403

Things are going going badly. What to do?

Guest lecture: Megascale software engineering



- Engineering the Facebook News Feed: architecture, design, implementation, deployment
- With 1 billion users, ~1 million servers: how do you do this?
- Ari Steinberg, formerly from Facebook
- Monday, November 26, 2012

Announcements

- Deliverables for 11/19 release (due 11:59PM)
  - Documents up to date: req, arch, design, schedule
  - Release notes
  - Status report for the **release** (not due until 11/20) -- content should be part of 11/21 presentation
- Presentations on 11/21: 7 minutes, same logistics as before. Deposit presentations (pdf) in dropbox by 11:59PM on 11/20

Release notes

- A description of what you just released
  - High level, thematic description
  - Major functionality
  - Exceptions of what might not be working
  - How to access the release
- Approximately a “README”
- “We’re proud to announce release 1.4 of...”
- Largely for external consumption
- About 1 page

Project Presentations, 11/21 (7 minutes)

- Operations Review by manager or project manager-- similar to weekly status but for the release cycle
  - What did you say you were going to do? What did you do? (1 minute)
  - What are you going to do in the next cycle? (1 minute)
  - What are the issues? (1 minute)
  - **What's your current status on platforms and browsers?**
  - Report on a metric -- or if not in place, scale from 1(disaster) - 10 (exceeding) and why (1 minute)
- Demo (1 minute)
- Questions and Answers (1 minute)

Announcements

- Extension! Your final projects aren't due until **12/08**
- Final moved to **12/07 (Friday)**
- Final presentations on **12/10 (Monday)** -- 14 minutes instead of 7
- Release 2 presentations on Wednesday, 11/21-- should reflect what you are going to do with the additional time from the extension

Final release

- Updated docs, release notes, status report (part of preso) as usual
- Project summary document
  - What did you deliver?
  - Quantity: How much did you deliver?
  - Quality: How good was it? (Design/UI/UX matters, bugs, usability, code)
  - Achievement of vision
- Software engineering principles
  - See wrap up and Joel on Software article
  - Answer all the questions in the wrap up (and more)

Presentation

- What you said you were going to do (~1 minute)
- What you did (~1 minute)
- What you would do next
  - Next release (~1 minute)
  - Strategically (~1 minute)
- Issues (~1 minute)
- Demo (~3 minutes)
- Important things you learned (~2 minutes)
- Discussion of checklist (~2 minutes)
- Questions and answers (~2 minutes)

Presentation logistics

- Date: December 10, 2012
- Time: 8:30a-10:20a (Finals slot for this class)
- Location: TBD
- Basically the same format as in the past, but 14 minutes in length
- Order of presentations to be announced by 10/7
- You only have to attend the one previous to your time slot ("on deck") and your time time slot (but welcome to attend all)

Wrap up write up

- Goal is to understand your product process and infrastructure in addition to what you built
- Answer all the questions (NOTE: Questions are in draft mode until 11/26)
- Your document contains the answers or you make submit a link to an online document
- Okay to hyperlink to the answers
- Anything additional you'd like to add

Correction

SSD vs NVRAM

CSE 403

You're project is going badly. What to do?

Things are going well

- Great!
- Are you sure?
- Are you working hard enough?
- Are there dark days ahead?
- Catch your breathe, but be prepared!

Things are going badly, what to do?

- You're late
- It doesn't work
- Company sucks
- You see the dark clouds on the horizon (code debt?)

You’re late

Why?

- Optimism
- Poor specifications
- Incomplete knowledge
- Changes in requirements
- Couldn't hire fast enough/at the right time
- Misjudged the capabilities of people
- People leave the team
- People aren't getting along
- Bad management

Why? (continued)

- Not enough process
- Too much process
- Fire-fighting issues from a previous release
- Personnel emergencies
- Buggy (third party) software
- Hardware problems
- Didn't know how hard it was going to be
- Didn't acknowledge problems earlier

Why? (continued)

- "Just one more feature"
- Badgered by management to do something else (or more)
- Cynicism
- Knowing dependencies are going to be late
- Conflict of ideas
- Paralysis by analysis
- Didn't think of "everything" (Under thought what needs to be done)
- Too much time on infrastructure
- Not enough time on infrastructure (code debt accumulation)

Why? (continued)

- Things are going well
- Dependencies are perceived to be late, so you adjust too
- Changes in strategies
- Lay offs
- Change in management
- Rumors
- Code debt
- Failure to accurately estimate

Why? (continued)

- Bad architecture
- Bad design
- Bad code
- Bad integration

Philosophically, these “problems” are just the common case!

Why? Failure to accurately estimate

- Engineers are typically optimistic
- For all the (external) causes for lateness, burden often falls on engineers to estimate how long something will take
- How long it takes to implement/debug is often times hard to estimate too (internal cause)
- Engineers are bad at estimating
- External pressures make engineering estimates unacceptable
- Back to the drawing board, perhaps with unrealistic constraints
- Re-estimation (which takes more time)

Responses

- Engineering manager doubles (or triples) the estimate of the engineering team
- Engineers "sand bag" their estimates
- Results in unrealistic estimation
- Change to train model or continuous release (just a panacea?)
- "Scapegoating"
- Break throughs in thinking (multi-month estimate reduced to minutes?)

Boeing 787: Not just software

- Initial first flight scheduled 7/07
- 9/07, announce 3 month slip
- 10/07, announce another 3 month slip
- 1/08, another 3 month slip
- 4/08, slip to 4Q08
- 11/08, another slip
- 12/08, slip to 2Q09
- 6/09, slip to end of 2009
- First flight December 15, 2009

In construction

Sagrada Familia

- Large Catholic church in Barcelona
- Unesco World Heritage site
- Started in 1882
- Gaudi involved in 1883
- Many changes/project slips over tim
- Still unfinished!

Largely due to same problems we are discussing!

Late? What does that mean anyway?

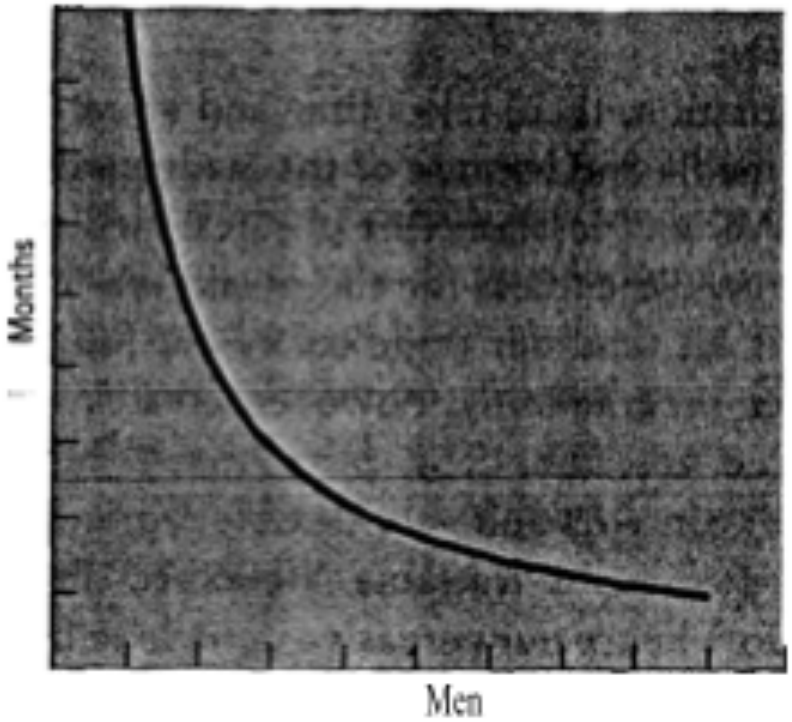
- Feature release model: Miss the date
- Train release model: Less on the train
- Continuous release model: Less stuff projected to be done by a given date

Options

- Add more people
- Cut a few features
- Cut a lot of features
- Slip the date by a little
- Slip the date by a lot
- Go back to the “drawing board” (major replan)
- Remove a few people
- Work harder, longer, smarter

Adding more people

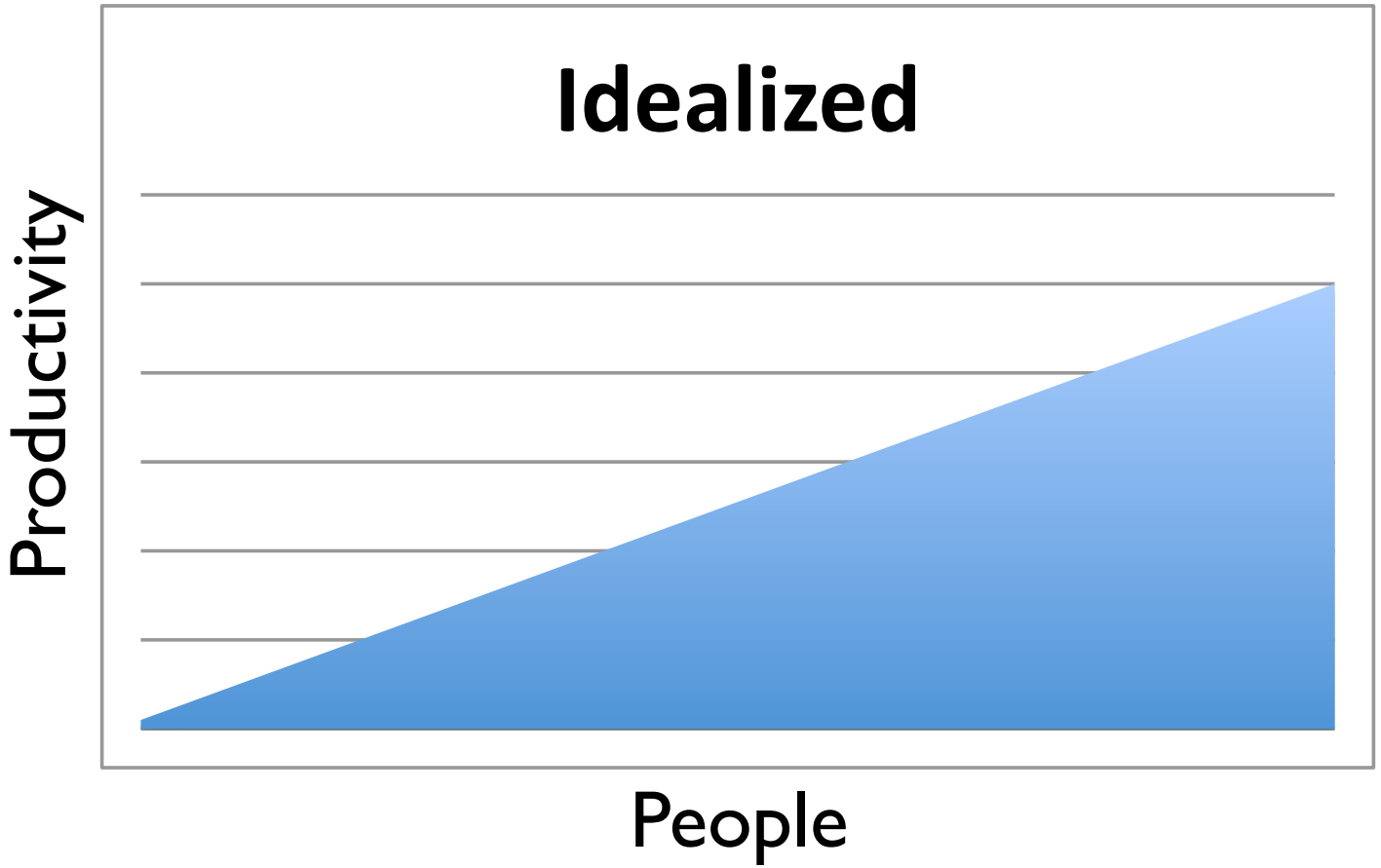
Adding more people (our hope)



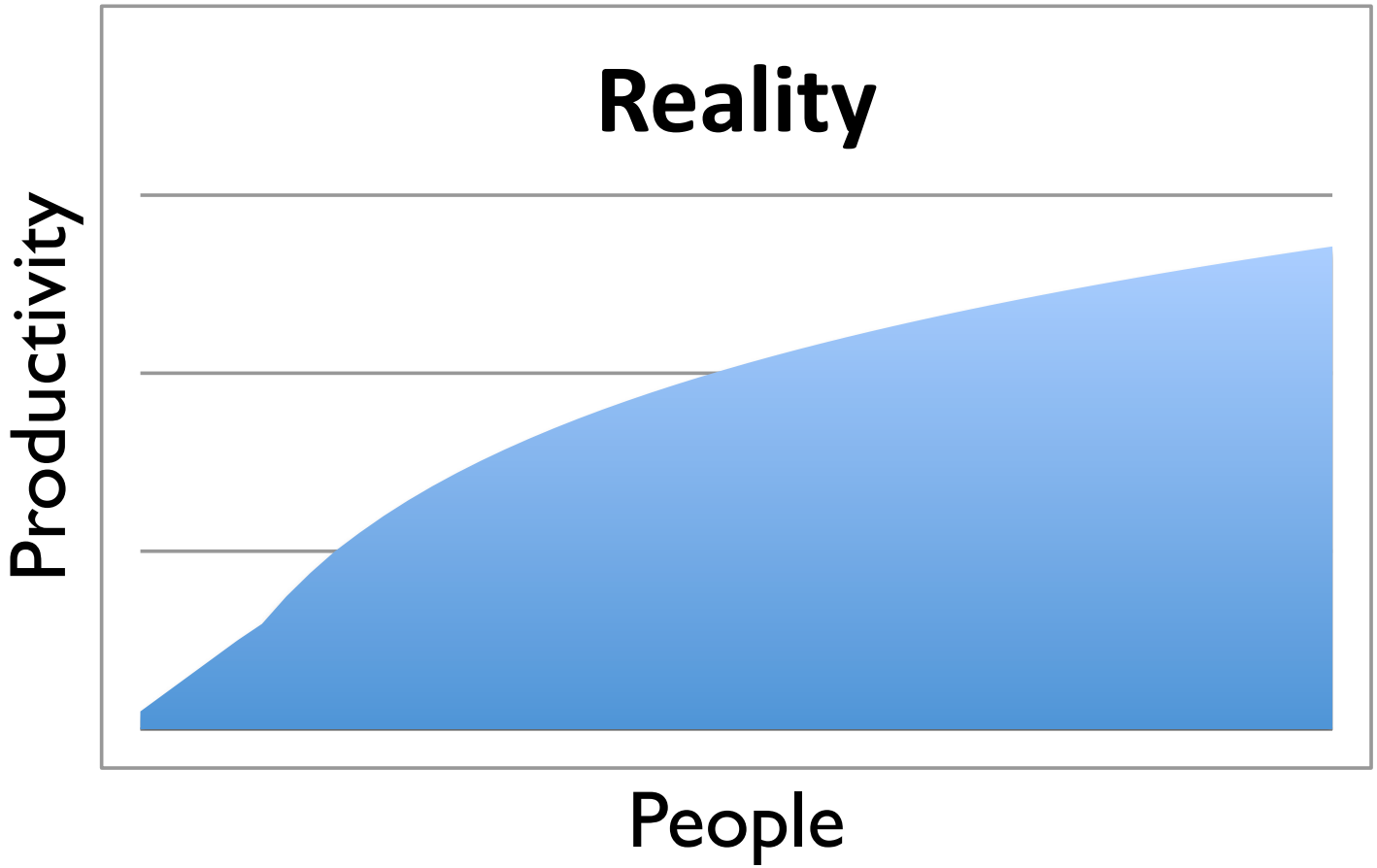
Adding more people (potential reality)

"Nine women can't have a baby in a month"

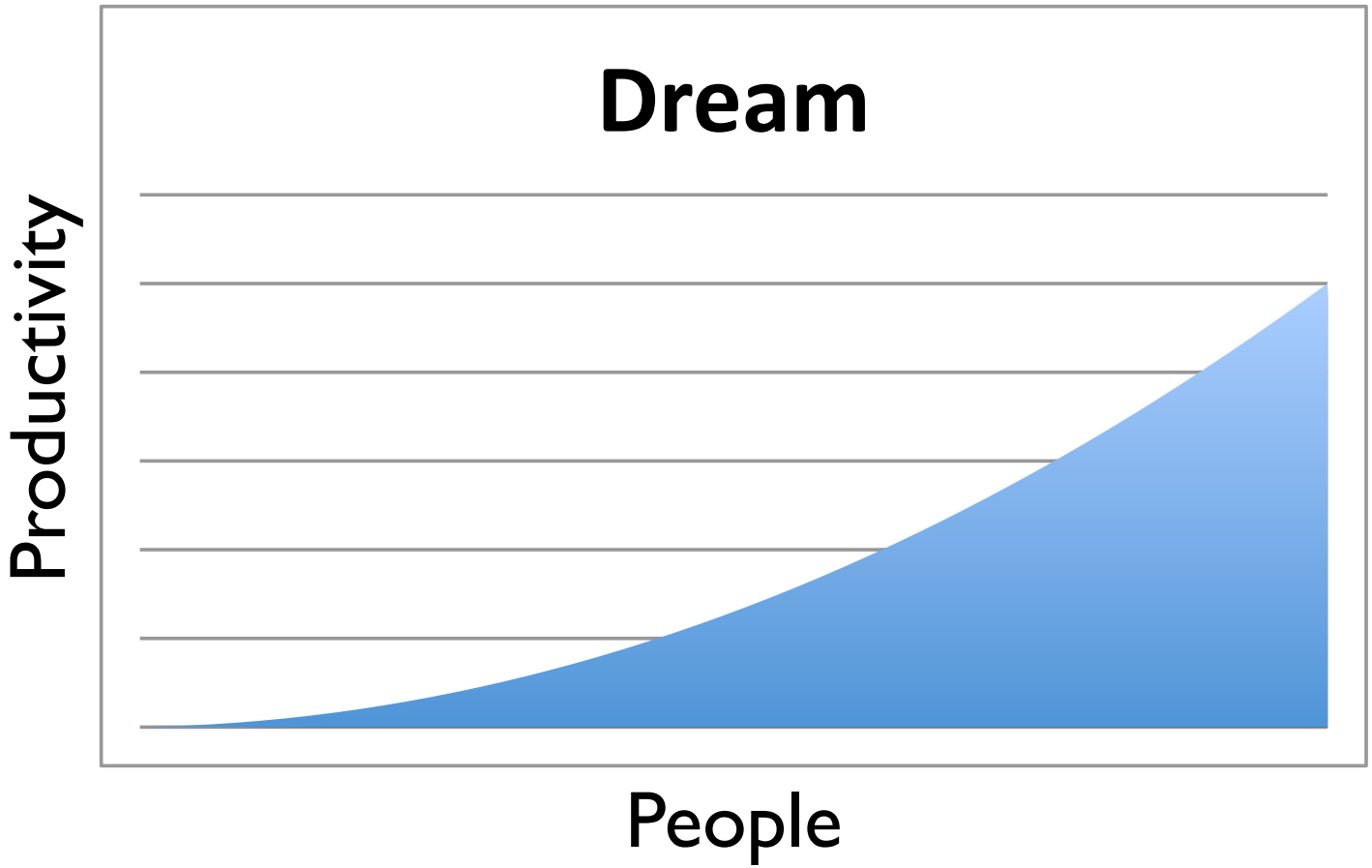
People vs. Productivity



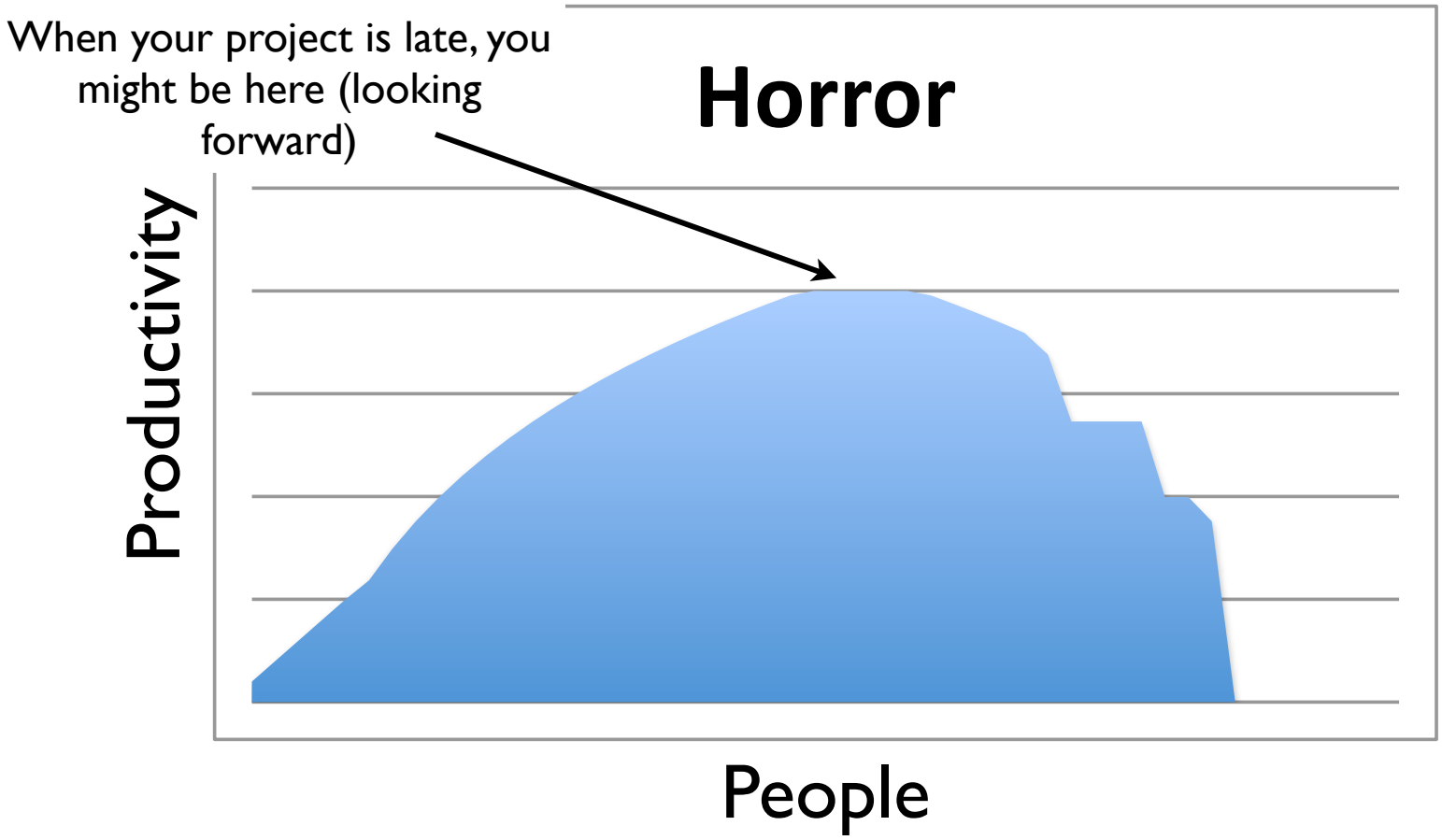
People vs. Productivity



People vs. Productivity



People vs. Productivity



Brooks' law

Brooks's law is a principle in software development which says that "adding people to a late software project makes it later"

Why does adding more people fail?

- Takes resources to recruit (impeding real work)
- Communication costs go up with more people (potential pairwise communications are O(n^2), groups are O(e^n))
- Takes resources to train
- Re-planning takes time
- Disruption of organization, new roles get assigned

Why does adding more people fail?

- Need to communicate new plan
- Chaos and confusing
- Feedback to rest of company requires bottom-up, top-down, bottom-up, top-down communication (more time)
- New plan not any better than the old plan
- People stop working waiting for the re-plan
- Worse, people leave during the re-plan
- Need to account for people leaving, so more re-planning
- Worse case: You enter the "death spiral"

Could adding more people work?

- Strategic hiring practice in place
- Contractors with specific skills could help
- Engineering process that supports adding new people
- Overall functioning organization

Cut a little

Cut a little

- Might work
- Pulling out partial features is work, risk
- Dependency assessment
- When are you going to get this work done?
- Did you just push your problems to the next release?

Slip a little

Slip a little

- Lots of pressure
- Notoriously underestimate how much additional time you need
- Still need to re-plan
- Disturbs the established pace/harmonic
- Screws up next releases
- Doesn’t solve the problem?

Other alternatives

- Choices: Cut a lot of features, slip the date by a lot, go back to the "drawing board"
- Still need to re-plan
- Still need to estimate (are you going to do a better job?)
- Jeopardizes strategic external commitments
- Is all this stuff really necessary?

All choices require (re-)planning

- Re-planning takes time
- Which must be taken into account
- Wasted time during re-plan?
- Approval/Communication is top down, bottom up, top down, bottom up, top down...
- More caution -- CYA -- adds more inefficiency

Working harder, longer, smarter

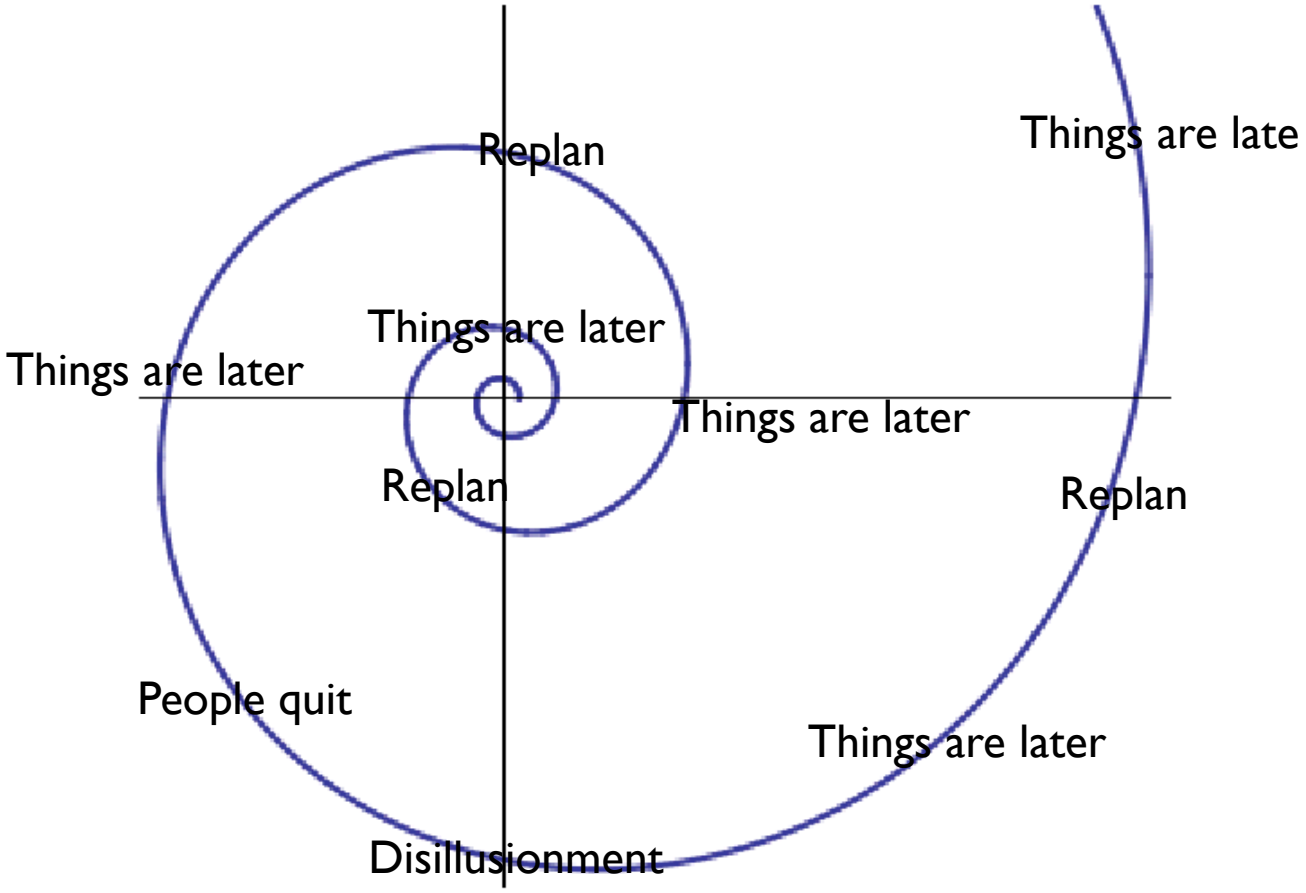
- Might work
- Not long term sustainable
- Motivational issues
- Cynicism is contagious
- Should management's failure to plan and lead constitute an emergency on the engineering team?

Choices for engineers

- Be positive and proactive
- Stay out of the way of the "ball of blame?"
- Be cynical (bad)
- Quit (what are the consequences?)

Software death spiral

Software death spiral



How does this relate to trains and continuous processes?

- Trains: reset expectations on which trains features will go out
- Continuous: reset expectations when features will come out
- Tactical re-planning less?
- Is there less accountability?

Take aways

- Uncertainty and change are parts of the "normal" process; being late/behind is a common side effect
- Optimism is a key reason for being late
- Brooks' law: Adding new people to a late software project make it later
- Solution: Add people, small cuts, big cuts, small slips, big slips...but all have issues
- Work harder, longer, smarter: Long term issues too

What does this mean for your project?