The Math (and Psychology) of scheduling, bug tracking, and triage

apenwarr@gmail.com  August 2018
Unusual Applications of Queuing Theory

apenwarr@gmail.com  August 2018
Advice, Obvious in Retrospect

that seems profound
when accompanied by fancy charts

apenwarr@gmail.com  August 2018
1. Project estimation
Why Goal Setting Doesn't Work

Goal setting may actually be counter productive if not a waste of time

Posted Jul 11, 2014

We have all heard this advice: Set goals if you want to accomplish anything substantial. That advice comes from personal coaches, self-help gurus, management consultants, managers and executives and is deeply imbedded in leadership practices.

In organizations, “stretch goals,” or “hairy audacious goals,” as a management motivational and performance strategy, is widely practiced. Yet, there is evidence that goal setting may actually be counter productive if not a waste of time.

-- Psychology Today
“When we used milestones, and you knew that you had two weeks to complete a step, the two weeks were yours. I, as project leader, couldn't do much to push you to finish earlier.

Moreover, if I came after one week and started to press, even inquired, you would have reacted as if I were out of line. ‘There is still a week to go, what do you want?’”

-- Critical Chain
The schedule is not the place to play psychological games.

-- Joel on Software

The schedule is absolutely the place to play psychological games. But you have to play to win!

-- me
Story points are **ratios**

- People are weirdly good at ratios
- **NOT** absolute days or hours
  - ⇒ Student Syndrome and sandbagging
- **NOT** small/medium/large
  - ⇒ How many smalls make a large?
- Pseudo-fibonacci sequence works well
What is a “story” anyway?

- A small bit of useful functionality *delivered to a customer*.

- The customer must *actually be impacted* (though maybe they won’t notice, like when reducing downtime)

  "*You can’t tell a story without the main character.*"
  
  (the customer)

- Not the same as a bug, task, or ticket
Strict prioritization

“When the facts change, I change my mind. What do you do, sir?”
-- possibly J. M. Keynes
How to do project planning

1. List the **sequence** and **relative size** of stories in a spreadsheet.

2. **DO THE STORIES IN THAT ORDER.**

   Tools are a distraction. Stop using them. Sorry.
Restricted multitasking (Kanban)

- 1 feature/launch; 5 completed
  aggregate value delivered = 4308

- 2 features/launch; 4 completed
  aggregate value delivered = 3590

- 5 features/launch; 5 completed
  aggregate value delivered = 865

- 8 features/launch; 0 completed
  aggregate value delivered = 0

Discussion
2. Stories are not bugs
Stories are not bugs

Stories:

- Slow (~weeks)
- Infrequent
- Controversial (PMs, execs)
- Can be tracked on index cards, spreadsheets

Bugs:

- Fast (~hours)
- Numerous
- Boring (engineers)
- Need automated tracking
Every bug is, on average, the same size

Central Limit Theorem

Stories are lumpy...

...but bugs average out.
Every bug is, on average, the same size
Every bug is, on average, the same size.
Every bug is, on average, the same size
Bug scheduling tips

● Use **stories** for **long term** prioritization and estimation

● Use automatic **bug burndown** charts for **short term** prediction

● **Quality is part of every story:** tag all relevant bugs
  ○ Don't work on bugs that aren't in the current story

● "Regressions and emergencies" is its own top-priority story
3. Triage
Bug bankruptcy doesn’t work (ingress > egress)
Ingress is always > egress; that’s why we prioritize!

(Despair)
Making “too many bugs” manageable

Fixing bugs:
- Slow
- Requires expertise
- You can’t ever fix them all
- Fix it right the first time
- You need a system to handle the inevitable backlog.

Triaging bugs:
- 100x faster
- Easy to parallelize
- You can triage them all
- Expect occasional re-triage
- No big deal if you do a little each day.
But... clutter!

People worry they’ll lose track of bugs if they keep too many open.
Bug tracking: You’re holding it wrong

- Never look at all the bugs in a popular component. You’ll drown. Components are **only needed for triage**.

- Don’t create too many components. You only need about one component per **triage team**.

- Instead, use hotlists (tags, labels, whatever) to track:
  - Triage status
  - “Needs Discussion” status
  - Release / Sprint / Milestone sequence
  - Feature backlogs (overlapping; one per major feature area)

- Bugs don't need to be assigned to someone
Re-triage: when it's already too late

“*It took us two years to get into this mess.*
*It’s okay if it takes us two years to get back out.*”

- Once you have a triage methodology, you don’t have to triage all your old bugs right away.

- Instead, create a “rolling query” that just pops up a few old bugs each day:

  componentid:999123+ -hotlistid:888424 -created:730d

- Re-triaging **a few at a time** is surprisingly rewarding:
  old bugs are much easier to assess than new ones.
Questions

Super overkill extended director's cut slides and text:

[apenwarr.ca/log/?m=201712#13](apenwarr.ca/log/?m=201712#13)

or search for "apenwarr epic treatise"