4156 endearly - Nipun proposal 11/24/15

ending an iteration / starting the next
ch 9 ~~Kanban - remotely repetitively~~
book assumes you do certain things as part of iteration!

customer-driven functionality
continuous integration
solid test coverage
reliable progress tracking
pacing adapts to team

& discusses other topics as if they're only done when there is time left over

- but for many organizations, some or all of these are built into iterations, NOT "extra"

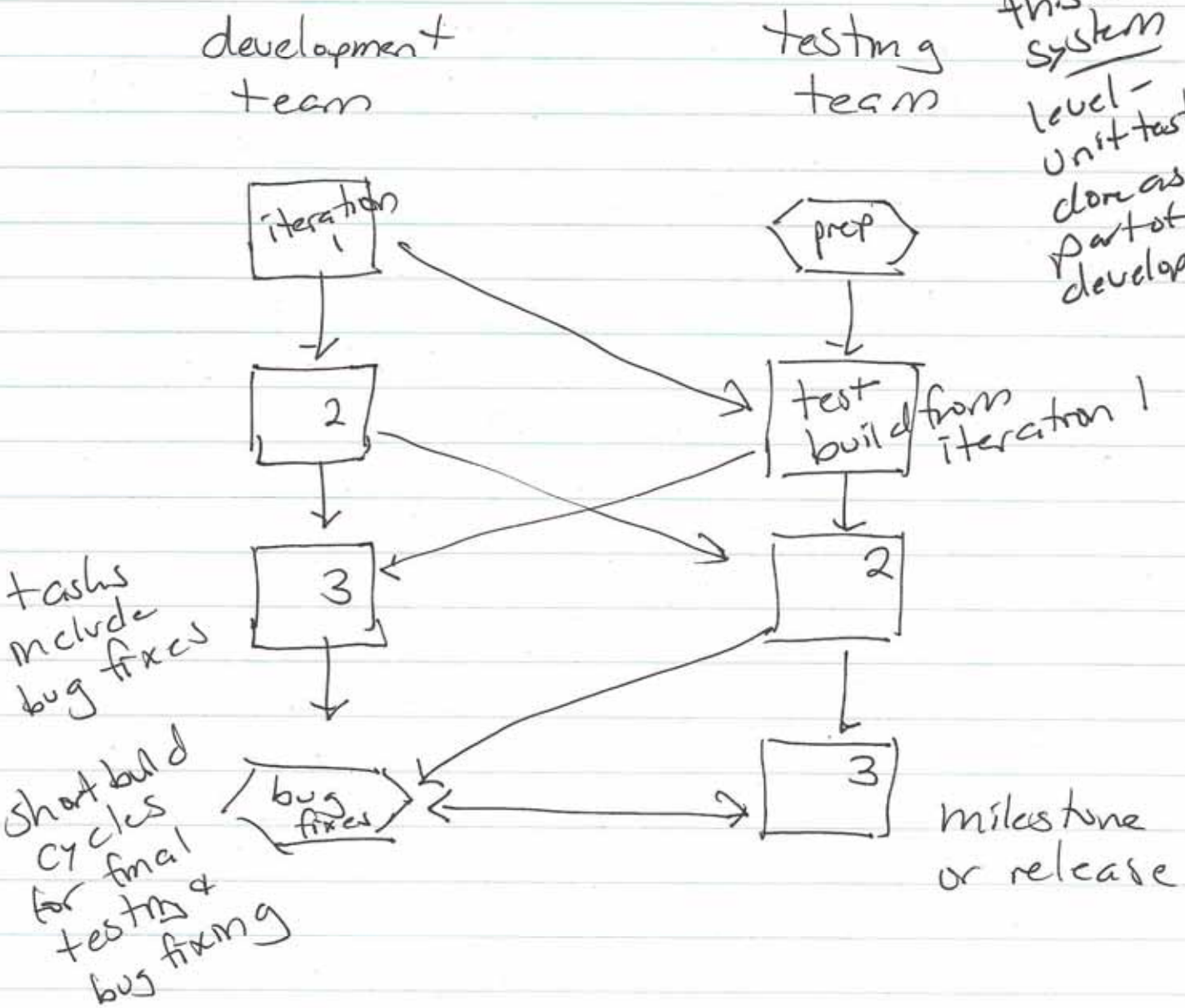
system-level testing
refactoring & code cleanup
documentation updates
process improvement

development environment updates
~~process improvement~~
R&D on new tech
training & personal development time

4156

11/24/15

full system testing:
good system testing requires
TWO iteration cycles



emphasize this is system level - unit testing done as part of development

dual iterations means lots more communication - maybe test team rep attends dev team standup meetings

4156

11/24/15

try to keep both iteration sequences in sync - time box

- note testing iteration 1 might take shorter or longer than developing iteration 2
- so ... ?

sometimes continued development depends on fixing bugs

one option is to start incorporating bug fixes as tasks as soon as bug reports available

could do test build w/in iteration 1 (as opposed to end) if full system available to test earlier

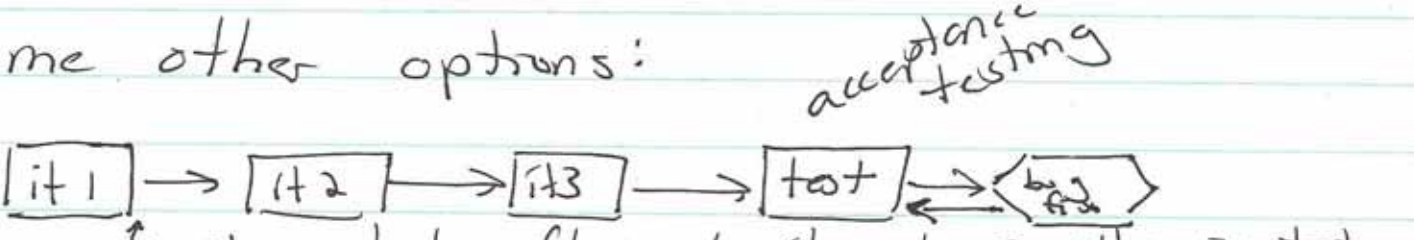
or carve off portion of time - not included in developer days for iteration - where development team works on bug fixes, e.g. 20% (one day per week)

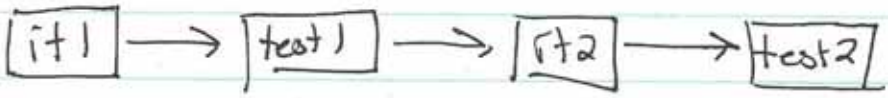
but may not have separate team available

4156

11/24/15

some other options:





50% of time spent testing

how to know when tested "enough"?

establish clear success criteria
e.g., zero-bug-bounce
(no outstanding bugs -
altho more may be found later)

"bugs" can include ambiguities, missing features, problems with website style

1. tester finds bug
2. files bug report
need reproduction steps
what tried to do, what happened, what expected to happen, any error messages

use bug tracker

Severity? likelihood?

4156

story is just work to do, doesn't have to be app feature

11/24/15

3. create story or task to fix bug
   prioritize w/ customer (remember these are system-level)
   severity? likelihood?
   maybe generic "bug fix" story
   w/ lots of tasks, keep adding
   but hard to estimate time
   - guess, use past experience — how long took to fix a "similar" bug

4. fix bug
   first write failing test that exposes the bug, then fix bug so test passes

5. tester verifies fix
   marks as closed

6. update bug report

metrics — new-bug submission rate
   going up or down?
   many bugs from same area of code?
   # left to fix & priorities

more contents for bug report
   need details — specifics
   version, platform, location
   what exactly does developer need to know to reproduce

4156 11/24/15

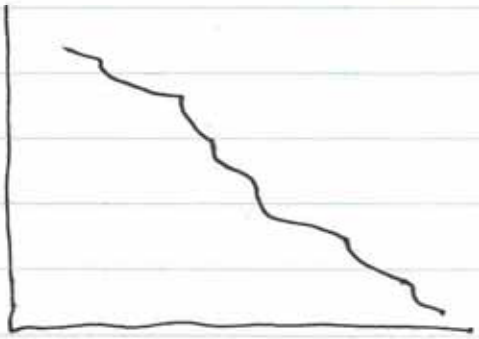
process improvement!

let's consider some possible burn-down graphs, what do you think happened?

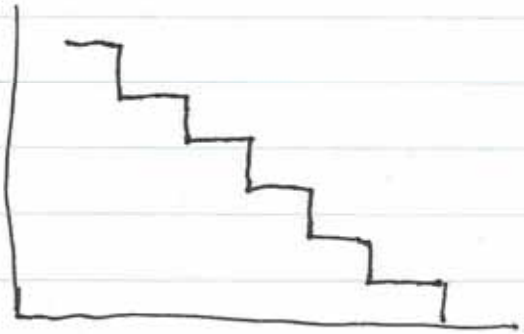
work remaining keeps increasing lots of unplanned tasks or bad estimates

drop off at end - probably cut out some things as deadline approached

how to prevent in future?



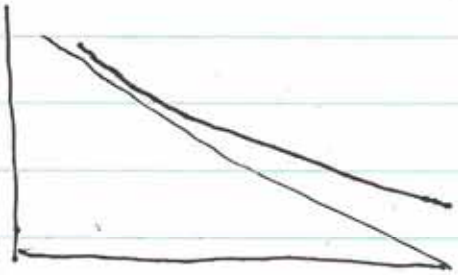
perfect! although not realistic



doesn't look perfect but it is realistic given user story completion in discrete units not continuous

4156

11/24/15



work keeps drifting to right of ideal
probably an estimate/velocity problem, plus didn't drop things at end to finish on time

can help future estimates to track how long between task moves to "in progress" and to "completed"

- if longer than the time estimate for the task, why?

System testing - as close to real-world as possible

- dedicated testers or beta release to real users

- if developers must do system testing, don't test own code (?)

4156

11/24/15

iteration review
- need to include time at end of iteration (or begin of next)

prepare for review mtg
be forward-looking
calculate metrics
velocity, coverage
checklist of topics

example questions
is everyone happy w/ quality of work? documentation? testing?
how feel about pace - frantic, reasonable, boring?
everyone comfortable w/ work assignments, area of system
are tools helping or hurting productivity? what new tools should be considered?
is process effective? should we change?
specific code to revise, refactor, rewrite?
performance problems?
big bug problems?
is testing effective? enough coverage?

4156 11/24/15

finally, is deployment under control + repeatable?

~~to~~ make "stories" for any process changes - although this would not work very well if what you want to change is the story-driven process

first two iterations often "bad" people underestimate how long tasks will take & then overestimate later on

make sure to record bigboard at end of each iteration, before wiping clean for new iteration

ch. 10 largely repetitive of each chapters BUT now the tasks are to integrate someone else's code

user stories again, whether you are writing code or reusing

- create class diagrams if don't already exist (there are tools to do this)

4156

11/24/15

and it already exist, check accuracy

need to be able to trust the 3rd party code
- test it first, before any integration with your code

ch 11 to-dos for integrating 3rd party source code

Create place in bug tracker for issues + use it

Organize source code into src, test, docs, etc. folders

high priority! do before changing anything

Write build script + get buildable

put code in your repository

integrate code into your CI config

Write tests simulating how you need to use the 3rd party code

other things to do eventually

figure out dependencies w/in codebase
how to package compiled version
document code

4156

1/24/15

run coverage report
Inc cant of code
Security audit
reverse engineer to create UML

focus on the functionality you need from the 3rd party code

how to estimate how long to get the 3rd party code working

spike testing - spend a fixed period of time, ex. 1 week, trying to fix code & extrapolate from that

pick random sampling of failing tests (do not pick just easy or just hard)

calculate bug fix rate

$$\frac{\text{bugs fixed}}{\text{\# days developer time}} = \text{daily bug fix rate}$$

$$\text{bug fix rate} \times \text{\# remaining bugs} = \text{estimate of how long to fix}$$

4156

11/24/17

better estimate might include
"confidence factor"
say it 70%

$$\frac{\text{estimate from spike testing}}{70\%\ (.7)} = \text{new estimate to fix reported bugs}$$

Ch 12

there are many successful sw dev processes

- develop iteratively
- evaluate & assess process itself
- best practices

ex. bigboard
user stories
version control
continuous integration
(test-driven development)
test coverage

we shipped