Tuesday 10/28/14 4156

based on feedback, decided to spend more time on software process - which is the implicit topic of nearly all the Head First Software Development book

assume we already have initial requirements - which will almost certainly change

how do we plan the rest of the project?

for each user story - or use case - estimate how long it will take anyone (or any pair) on your team to do it, including writing + executing test cases

1/2 day, 1 day, 2 days, etc.
15 days too long

add them all up to see how long whole project will take - will almost certainly be far too long

need customer or customer rep to prioritize what will go in version (or milestone) 1.0 & will fit, assuming 20 working days per month & 3 months max to first milestone delivery

Tuesday 10/28/14 4156

more details on estimation process

for 1st iteration use days - or "points" only if there's similar projects so can convert points to days

later iterations, can use points as relative time units & then convert based on this project - need to decide 1st what was an exemplar point from 1st iteration, considering how long user stories really took

everyone on team (or every pair) does own estimates independently
then compare & discuss to reach consensus

3 C's - consensus, convergence, confidence

"planning poker" - everyone bids on estimate for a user story
be wary of outliers & wide spreads
may be problems w/ assumptions

see if user stories too long or too short
combine less than 1/2 day

Tuesday 10/28/14 4156

break up if more than 15 days
or even 3-5 days if 1-week iterations
"AND" rule - also "OR"

use spike solutions to improve estimates

for use cases, can treat alternate flows
as separate for planning purposes
(to make small enough)

redo estimates for upcoming user
stories (or use cases) at beginning of
each iteration

refine into tasks

define iteration as set of user stories
that will fit in one month (or one week)
based on project velocity

project velocity = how much productive
time, on average, we really get
start with 0.70 (70%)
+ adjust up or down
each iteration based on
experience

for 1st few iterations, all user stories
likely to be high priority (baseline)
so how to prioritize w/in them?

Tuesday 10/28/14

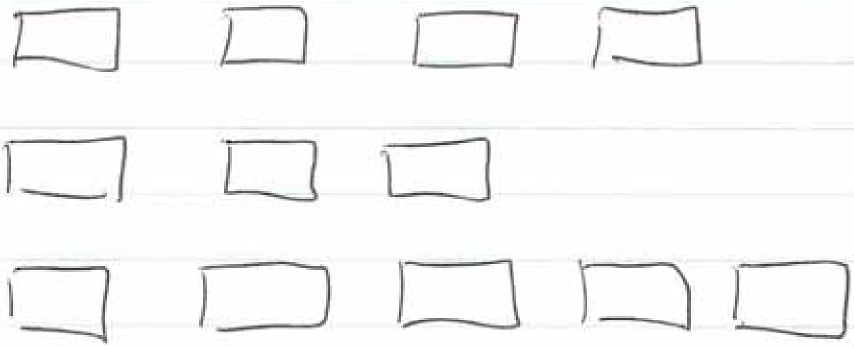
4156

buckets 10, 20, 30, 40, 50
maybe intermediate 25, 35
place equally critical user stories
in same bucket (but not all
in bucket 10 - highest)

iteration 1

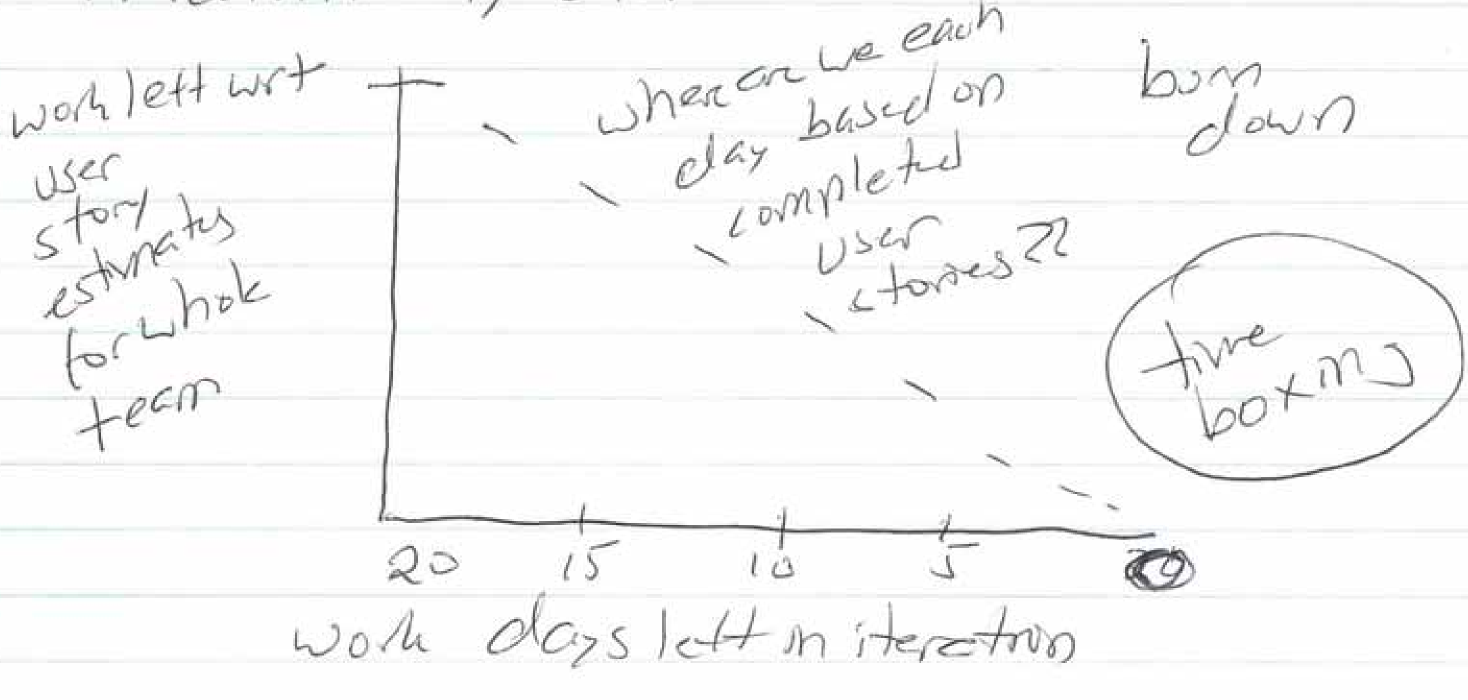
iteration 2

iteration 3



release 1 ~~plans for [illegible] user stories~~

iteration 4, etc.



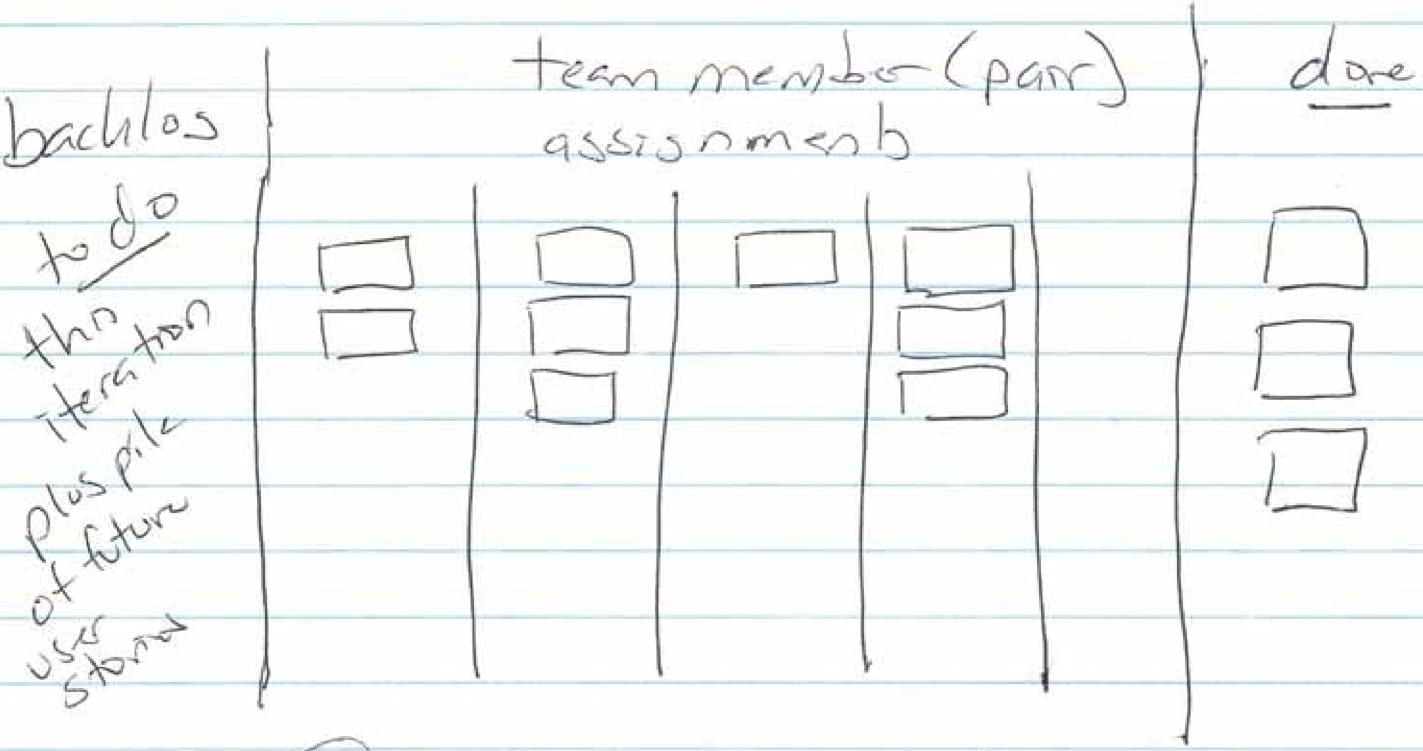
where are we each day based on completed user stories??

burn down

time boxing

Tuesday 10/28/14 4156

big board


backlog
to do this iteration plus pile of future user stories
team member (pair) assignments
done

Volunteer

might be assigned,
might pick from available
- finish, pick another

how does testing fit into iterations?

TDD of units part of developing those units, possibly also TDD at acceptance level (need infrastructure)

but there's other testing (a ~~and~~ document reviews)

Tuesday 10/24/14

before deciding a user story/use case is "done" need to integrate w/ repository & run all regression tests (all tests for previously checked in code) – "continuous integration"

all that has to be considered part of time estimate!

→ alternative nightly build runs regression tests – then need to consider separate scheduling of bug fixes (also when your checkin causes bugs in other code)

if process includes document reviews or inspections, need to include as part of declaring user story "done" or scheduled separately

one approach

→ it1 → it2 → it3 → test & debug → release

very late in process, & makes iteration longer (4 mos vs. 3)

Tuesday 10/28/14 4156

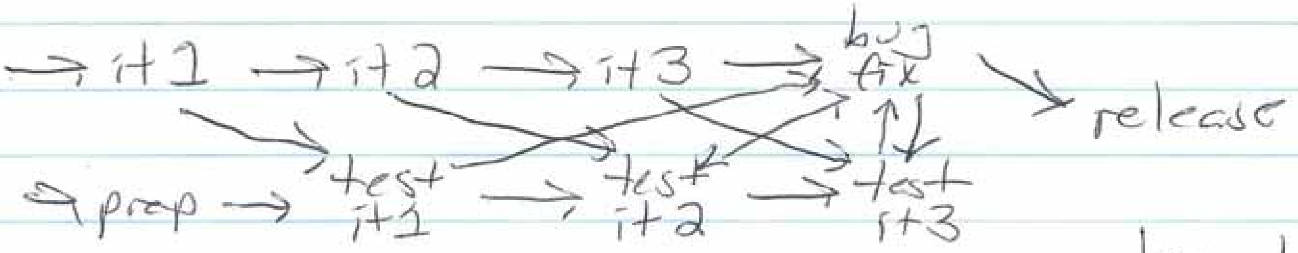
another approach

→ it1 → test1 → it2 → test2
→ it3 → test3 → release

doubles length of time until release
(6 mos vs. 3)

maybe test & debug is one week not one month,
but still takes longer

if separate testing team available, can
do in parallel



boxes testing time to match
iteration time but still
probably 4 mos vs. 3

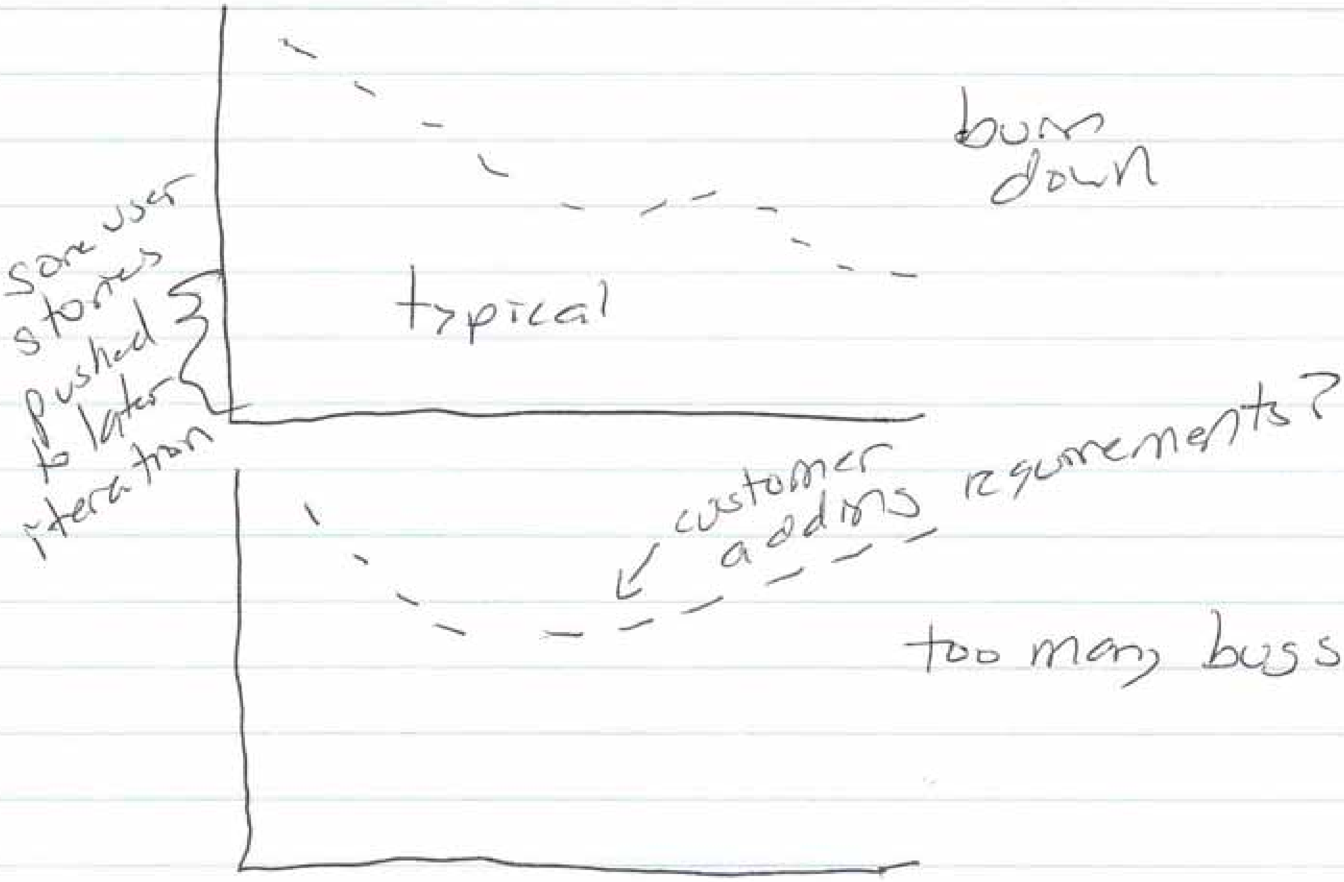
how to
get back
to 90
days?

testing iterations include
integration & stress testing,
not just unit & acceptance tests
(~~a later lecture~~) → on Thursday

Tuesday 10/28/14 4156

ending an iteration - post mortem
iteration review, process
improvement

~~recalculate project velocity~~



only consider new/changed requirements between iterations if possible

revise estimates for all remaining stories in current milestone

Tuesday 10/28/14 4156

recalculate velocity

total # days work accomplished based on original estimates / # actual working days (~20)

↑

x # developers = new velocity

calculate work days available

developers x ~20 x new velocity

↑

= # work days

fill board with work for next iteration remember to consider left overs from previous iteration – both user stories & bug fixing

get customer's approval – priorities might have changed

how long does bug fixing take?

Tuesday 10/28/14 4156

spike debugging

pick random cross-section of bugs & fix them

bugs fixed / # work days spent
= # bugs per day

or predetermine # of days +
keep picking new bugs until
times up

consider "confidence" N %

bug fix rate × # bugs × $\frac{1}{N\%}$
= # days to fix

need to include regression testing
(during checkin or nightly build)
in debugging time

doesn't count??
or does it??

Tuesday 10/28/14 4156

Scrum - "hot" version of agile
(supplanted XP)

an iteration is called a "sprint" & may be only 1-2 weeks, ~~most~~ ideally shippable

"product owner" = customer representative or go-between

scrum development team - might or might not involve pairs
"team room" 4-9 people, self-organizing
(like all agile)

Scrum Master - facilitator,
(same as "administrator" from Harlan Mills Surgical Team, 1971)
manages backlog (priority decisions made by product owner)
tries to avoid "technical debt" - doing it the wrong way
removes impediments

may b w/ other scrummasters but chief surgeon did thm in Mills team

may or may not use TDD but always uses continuous integration (push to test)

Tuesday 10/28/14 4:56

task board = agile big board

Same burn down chart

meetings - Sprint planning daily
daily standup "Scrum"
product feedback - Sprint review
process improvement - Sprint retrospective