4156 10/20/16

review for first exam

Head First ch 1 - 6½
Ron Patton ch. 4

Ron Patton ch. 4
"examining the specification"

specification = requirements
(user stories & use cases)

"testing" the spec enables finding bugs before any code is written

black box	vs.	white box
don't look at code		do look at code

static	vs.	dynamic testing
examine & review but don't run		run the sw

testing spec is static black box

.pretend to be customer
quality = "meeting the customer's needs"

4156 10/20/16

check existing standards & guidelines

review + test similar sw

specification attributes checklist

- complete
- accurate
- precise, unambiguous, clear
- consistent
- relevant
- feasible
- code-free
- testable

consider whole product & individual features

specification terminology checklist
problem words to look for

- always, never
- obviously
- sometimes, usually
- etc., such as
- good, fast, cheap
- handled, processed rejected
- if ... then ... missing else

415 6 10/20/16

Head First ch 1
"pleasing your customer"

deliver sw that is needed,
on time + on budget

big bang, going dark often means
wrong sw delivered to customer

iteration solves problem (process)

each iteration should produce
working sw
mini-project, quality sw
customer changes mind
about features, priorities
integrate new features into series
of iterations

4156 10/20/16

Head First ch. 2
"knowing what the customer wants"

user stories = title + description
single thing the sw needs to do

talk to customer, ask questions
"bluesky" brainstorming w/ stakeholders

role playing - pretend to be sw
observation - watch what do now

requirements must be customer-oriented
no technical terms
no design decisions

time estimates by developers
how long will it take
include design, test, code & deliver

total almost certainly too long add up for individual user stories to get estimate for each iteration & for full project

(risks)
what assumptions are developers making when determining estimates?
clarify w/ customer & each other

415 6

10/20/16

planning poker - convergence
larger difference ⟹ lesser confidence

calendar month = 20 working days

15-day rule - all estimates should
be ≤ 15 days (why not 20?)

AND rule - look for "AND" in user story
to break up

4156 10/20/16

Head First ch. 3
"planning for success"

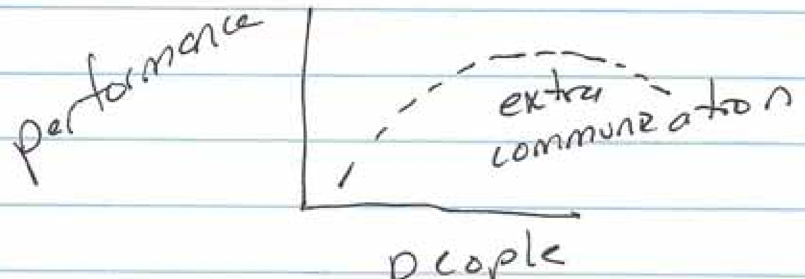
estimates add up to too long, so
customer has to prioritize

milestone 1.0 = 1st major release
deliver r.t. demo

focus on baseline functionality

milestone vs. version, milestone vs. iteration

adding more people ⟹ less productive
diminishing returns



customer prioritizes within high priority
what to do in next iteration
10, 20, 30, 40, 50 buckets

iterations 1, 2, 3 ⟹ 1st milestone
90 calendar days

4156 10/20/16

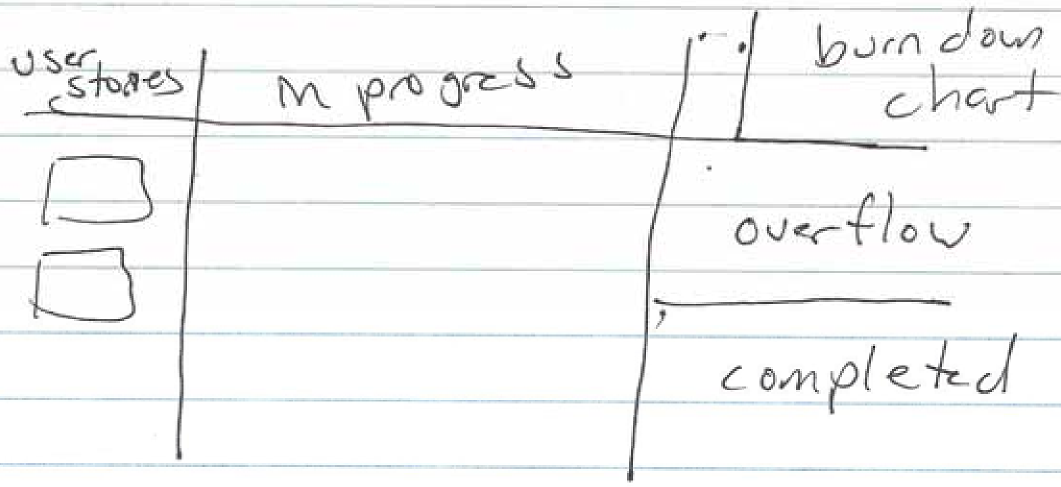
continuously building & always runnable
for customer feedback

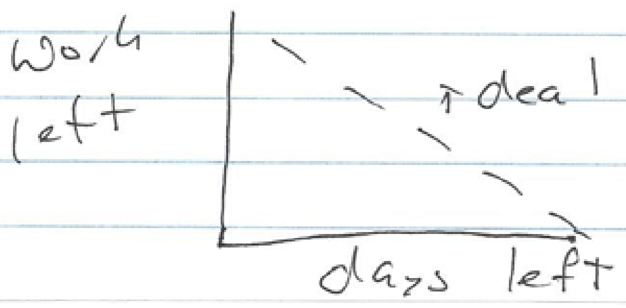
☆ project velocity — ignore for now,
will be covered later (and exam)

overflow work goes to a later iteration
adding more iterations at
end if need be

big board = development dashboard =
task board

what work is in pipeline, what's
in progress, what's done





4156 10/20/16

Head First ch. 4
"getting to the real work"

work is more granular than user stories
need to break down into tasks

same idea as user stories
title, description, estimate
but now in developer terms
not customer

1/2 day to 5 days

estimate for tasks should add up
to estimate for user story

use tasks instead of user stories
on task board

completed tasks vs. completed
user stories

assign tasks to developers
r.t. entire stories

one story at a time

4156 10/20/16

what are in tasks - classes & methods, UI screens, DB schemas, SQL scripts, etc.

keep task board accurate - daily standup meetings

work on strongly related tasks at same time

→ track progress
update burndown rate
update tasks
what happened yesterday & what's going to happen today
bring up any issues
5-15 minutes

ideally morning 1st thing

some examples with class diagrams & sequence diagrams

☆ you need to know class diagrams
you do not need to know (for test) sequence diagrams or other UML

some discussion of refactoring - also later (2nd test)

4156

10/20/16

how to handle unplanned tasks
e.g., customer asks for extra demo

add to task board like
other tasks

needs it own estimate

often results in putting off other
tasks to next iteration

10/20/16

Head First ch. 5
"getting it done with great design"

introduces problem with a particular feature (behavior) spread over multiple classes - ripple effect

breaks single responsibility principle

each object should have only one reason to change (cohesion)

SRP analysis

The classname methodname itself

for every method in class

do sentences make sense?
if not, method may belong in another class

when the method takes a parameter that is an object of another class

then ~~The class~~
The classname methodname and parameter ~~name~~ type itself

4156

10/20/16

not just SRP, also DRY
don't repeat yourself

avoid duplicate code by abstracting
out commonality to another
single location

mon on unplanned tasks

4156 10/20/16

Head First ch. 6
"defensive development"

use version control

details on how to use subversion (svn)
but we're using git

checkout / checkin - commit messages

Merge conflicts

tagged versions + branches/trunk

tracks who changed what when

roll back changes when needed

4156

10/20/16

Head First ch. 6 1/2
"insert tab a into slot b"

use a build tool & build scripts

mostly about Ant

everyone on team needs to use same

mentions running test cases