4156

9/22/16

team composition assignment
due tonight!
who doesn't have a team yet
(or has a team that is
not exactly 4 students?)

---

use cases vs. user stories

recall that user stories are very
short & simple

" as a <user role>, I want to
do (or I want the software to do)
<some feature>, so that
I gain <some value>"

a user story has been assigned
a priority (by the customer) &
an estimate (by the developers)

its been added to the team's
task board & assigned to
a developer (or pair of developers)

4156

9/22/16

now what?

it depends on whether this is one of the first/earliest iterations, where the sw doesn't do much (or anything) yet vs. adding yet another feature

let's assume for now the case of adding yet another feature (we'll consider initial development in a later lecture)

even after adding "conditions of satisfaction" (which probably don't fit on the index card - need to record in an issue tracker), we still may not understand how the user story needs to operate

this is why user stories are often expanded to "use cases"

use cases describe the step by step details of how the user role interacts with the software to obtain the value of a feature (or achieve a goal)

also try to capture everything that can go wrong during process

4156

9/22/16

Use case elements

name - verb/noun or
actor/verb/noun
communicates scope of use case
(also use label to match
with corresponding user story)

description - brief paragraph
describing the scope of the use case
(this could be a copy of the user story)

actors - types of users who
engage in activities described
in use case

preconditions - anything that should
be true before use case begins
- specify whether can assume true
or need to check

basic flow (happy path) - set
of steps actor takes to
accomplish goal, with clear
description of what system
does in response at each step

4156 trigger

9/22/16

alternate flows - less common user/system interactions, special cases

exception flows - things that can happen that prevent user from achieving their goal
error cases

post conditions - any things that must be true when use case is complete

use cases should capture <u>what</u> the SW does, not <u>how</u>

you do not need UML use case diagrams or any other UML diagrams for use cases

however, wireframing diagrams - mockups of user interface - are often useful (will cover later on in course)

workflow diagrams also useful

in theory, all this extra detail should provide enough information for developer

continue here

but sometimes still not enough

this is one motivation for par programming

need good workspace w/ lots of room

show par programming images

many companies employ par programming for interviews (auditions), even if don't use for regular development
- be prepared!

par programming applies to all phases of development, not just coding

2 people sitting at same PC (or using remote desktop sharing)
- take turns "driving" vs. "navigating"
- switch every 25 minutes or so
- continuous code review

studies show higher quality but mixed results on productivity (does it take twice as long?)

9/22/16

organizations that use pair programming typically switch pairs often
- collective code ownership
- ~~more~~ maximize truck factor

other benefits of pair programming

stay focused & on task
less likely to read email, surf web, etc.
less likely to be interrupted

each expects other to follow best practices

two people can solve problems that one couldn't do alone & have better quality solutions

pool knowledge resources

"over the shoulder" defect prevention & removal
reduces (but doesn't eliminate) need for formal code inspection
(we will cover code inspection later in course)